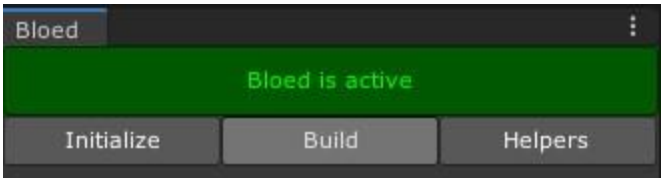
Bloed is a level editor for Unity for quick&dirty prototype levels. It is available on the Asset Store (https://u3d.as/2LR1) and on itch (https://ratking.itch.io/bloed).

Contact

For bug reports or other inquiries please contact the developer via the itch.io forum or via https://ratking.de/blog/contact/. The current version of Bloed was created with Unity 2019.4 LTS and tested with 2020 LTS, 2021 LTS and 2022 LTS.

The Editor: Bloed

To open the editor, choose **Window** → **Bloxels** → **Bloxels Editor** in the menu bar. The editor has three tabs - **Initialize**, **Build**, and **Helpers**. When Bloed is **active**, there's also a **tool box** inside the Scene window, on the left side.



Initialize

- In a scene with no bloxels yet you have to initialize a **BloxelLevel**. A box appears with the title “Create New BloxelLevel”, with an “Initialize!” button to actually generate a level in the scene.
    - **Settings:** Insert an asset of type **BloxelLevelSettings** here, which lets you define the standard bloxel(s) used in this level, e.g. if it’s CUBE or AIR. (So you either ‘carve out’ the geometry, or you build it.) BloxelLevelSettings also define standard layers, tags and other things for the bloxel geometry chunks.
    - **Chunk Size:** Standard is 8, so a chunk can contain 8*8*8 bloxels each. You cannot change this after the BloxelLevel was created.
- You can create as many BloxelLevels inside a scene as you like.
- If you created new BloxelTypes inside the *Resources/BloxelTypes* folder, or changed existing ones, click on **”Update ALL Types & Textures”**.
- If you changed or added some BloxelTextures only, choose **“Update Textures Only”**.
- Afterwards it can be useful to recreate all meshes in all scenes, you can do that with the button **“Recreate ALL Existing Meshes in ALL Scenes & Prefabs”**. The current scene must be saved in advance, as this function iterates through all prefabs and scenes in the project. So it might also take some time before it finishes.
- The **Display Settings** let you change several colors of the various grids and markers in bloed.
- The settings in the **Other Settings** let you change things like the default chunk size, the raycast distance for picking and placing bloxels, or the extent of the grid.

Build

- **Grid X / Y / Z:** Activates a grid that will be used for placing/removing any bloxels. You can activate/deactivate the grid with the hotkey G. To change the orientation of the grid’s plane, use the hotkey Shift+G.
  Hint: the grid is not visible in the Texer tool or the Joister tool.
- **Lock Cursor To Grid:** Locking the camera pivot (called the 3D cursor) to the current grid position is very useful so you don’t move the grid around accidentally. If locked, you can move the grid into the positive direction with Space, and into the negative direction with Shift+Space.

- **Types:** Displays the current shelf of bloxel types you can choose from. Change the shelf by clicking (with LMB) on the Types button, or use the mouse wheel while hovering it with the mouse cursor. By right-clicking on the button, the respective BloxelType in the Project will be selected.
  If you press 1 inside the Scene, you choose the AIR bloxels. With 2 you switch to the CUBE bloxel. If you press 1 or 2 again, it changes back to the previous bloxel type.
- **Textures:** Displays the current shelf of bloxel textures you can choose from. Change the shelf by left-clicking on the Textures button, or use the mouse wheel while hovering it with the mouse cursor. Hold Shift and use the mouse wheel over a texture to increase or decrease the displayed size of all texture buttons. By right-clicking on the button, the respective BloxelTexture in the Project will be selected.

Helpers

The helpers are mostly needed for optimizing the level data. But the **first** three buttons actually let you remove and recreate meshes. You can either “Affect ALL Bloxel Levels” or change the currently active BloxelLevel only. It’s also possible to “Affect Joists Too” or ignore them. (Joists are cuboids that use Bloed’s textures - see the tools below.)

- **Remove Meshes:** Will remove all of the meshes of the chunks (and joists), but not the actual bloxel data. This is useful if you want to keep your scene’s file size small. You can create the meshes again anytime with the Recreate Chunk Meshes button, or activate “Recreate On Start” in a BloxelLevel.
- **Recreate Meshes:** Will regenerate the meshes of the chunks according to the current bloxel data.
- **Generate Secondary UVs:** These UVs are needed for lightmaps. As soon as you edit a chunk of a BloxelLevel its secondary UVs become invalid, so this button generates them.

The **second** box with buttons visualizes certain bloxels and bloxel sides. This only applies to the currently active BloxelLevel. To see the marked bloxels, Bloed has to be set as active.

- **Mark HIDDEN Bloxels:** Bloxels are “hidden” if they have a texture different from the standard texture, and if they’re CUBEs/full, and if they’re surrounded by other bloxels so they don’t actually have any vertex data in the mesh. This means they are (usually) unnecessary data.
- **MISSING Bloxels:** It might happen that there are bloxels of types not available anymore. This button displays them.
- **Mark ALL ExtraSides:** The Texer tool lets you add additional side data to a bloxel. This button marks all side data created this way.

- **INVISIBLE only:** With this button you see side data that is invisible. For example, if you applied side data to bloxel sides with no vertex data anymore (because you placed another bloxel adjacent to it), or removed the bloxel (by replacing it with AIR), the side data is invisible. It is unnecessary data that should be removed.

The **third** box with the four following buttons might change the bloxel data, but do **not** generate an **undo** step. *Use with care!*

- **Remove All HIDDEN Bloxels:** Lets you remove all bloxels that are hidden - see the definition above.

  If your game allows you to destroy bloxels, so you can “dig” them out, then you probably should *not* use this function, otherwise data will be lost.
- **Clean Up MISSING Bloxels:** Missing bloxels will be removed, and replaced by the standard bloxels.
- **Remove All INVISIBLE ExtraSideDatas:** Lets you remove all side datas that are invisible - see the definition above.
- **OPTIMIZE Bloxels With ExtraSideData:** Used to optimize the bloxel data. It mostly tries to find out if a bloxel with side data would be better off if its texture actually gets changed to the side data’s texture and the side data removed.

Tools

Change the current tool by using the **<** or **>** button on the left side of the Scene view. The tools are only visible if Bloed is **active**. Each tool’s action can be undone with the hotkey Z, and redone with Y (without Ctrl).

BLOXER

The **standard** tool of Bloed. You place bloxels by pressing the left mouse button. To actually have an effect you either need to mouse-hover an already existing bloxel, or have the grid activated and hover it.

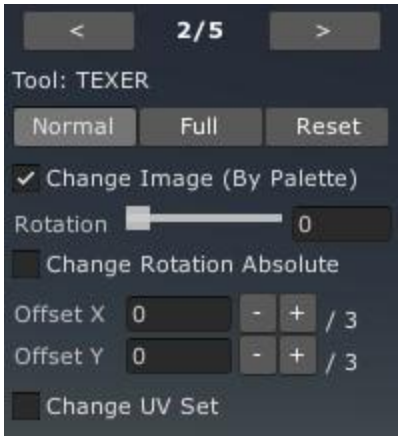
- This tool lets you change the chosen template (i.e. rotation) of the current bloxel type. Use Tab and Shift+Tab to quickly go through the different templates.
- You can **pick** an existing bloxel’s template by pressing Ctrl+LMB.
- You can **pick** an existing bloxel’s texture by pressing Ctrl+RMB.
- If you want to place a bloxel at the position of the 3D cursor (camera pivot), keep Shift+RMB pressed.

You can also place or remove **several bloxels** at once, in a cuboid with the size of up to 9x9x9 bloxels.

- **Centered:** If true, the cuboid will be placed so the currently picked position is at its center. If false, the current picked normal will be used to place the cuboid.
- **Hollow:** If true, the bloxels inside of the cuboid will not be modified.
- **X/Y/Z:** Change the size of the cuboid.
- **Tex Padding:** The surrounding bloxels will also change their textures to the one chosen in Bloed. This is useful if you want to create caves/rooms with the AIR template, and let them instantly have a certain wall texture.
- **Change Tex Only:** What it says. Instead of changing the bloxels' templates and textures, only change the latter. Useful for "painting" the environment quickly.

TEXER

Lets you change the appearance of individual **bloxel sides** by pressing the left mouse button. To actually have an effect you need to hover a bloxel's side or inner mesh. This tool has three modes: Normal, Full and Reset. With **Full** you actually change the texture of the full bloxel, so no new side data is generated. With **Reset** you just remove all side data of a bloxel. The **Normal** mode has the following options:



- **Change Image (By Palette):** Change a bloxel's side to the texture currently selected in Bloed. If deactivated, the bloxel's current texture won't be changed.
- **Rotation:** Choose the rotation of the texture, from 0 to 3.
- **Change Rotation Absolutely:** If this is not activated, Rotation changes the rotation on each click. Otherwise it sets the rotation absolutely. If this option is enabled, use Tab and Shift+Tab to quickly change the Rotation.
- **Offset X/Y:** Change the offset of the bloxel's texture. This only makes sense if the texture type has a size bigger than 1x1.
- **Change UV Set:** Only applicable for the inner mesh data of a bloxel, and only if the texture actually has more than one UV set. Check the "BloxelType" topic under "Assets and Folders" to learn more about UV sets.
- You can pick an existing bloxel's texture by pressing Ctrl+RMB.

CUBOIDER

With this tool you can place or remove **several bloxels** at once, in a cuboid of arbitrary size. Create the selection cuboid by dragging the mouse while holding the Ctrl key and LMB. Having the grid activated (via G) is recommended for this operation. Use the buttons on the bottom of the tool to fill/clear/texturize the cuboid. You can also move the selection by clicking LMB.

- **Fill Hollow:** If true, the inner bloxels are not affected. You can then also define the Thickness of the walls of this hollow cuboid.
- **Texture Padding:** The surrounding bloxels will also change their textures to the one chosen in Bloed. This is useful if you want to create caves/rooms with the AIR template, and let them instantly have a certain wall texture.

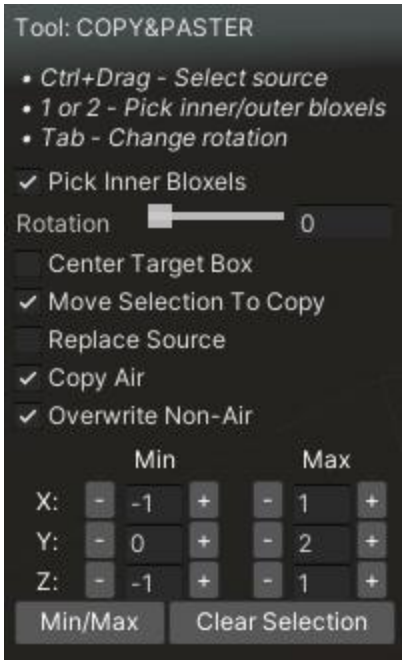
Only visible if there's a selection box:

- **New Box Height:** This value is used when creating a new selection cuboid on the grid.
- **Negative Height:** If true, the new and the current box height will be interpreted as depth.
- **Center Box:** If true, using LMB to move the box centers it at the mouse cursor. Also "rotating" the current selection box via Tab will do it around the center instead of the Pos.
- **Move Box With Grid:** If true, the current selection box will always be anchored to the grid.
- **X/Y/Z:** The position and size of the current selection box can be changed directly here.
- **Pos/Size / Min/Max:** Toggle between position and size display or minimum and maximum display of the cuboid.
- **Clear Selection:** Remove the current selection cuboid.
- **Fill (*Template*/*Texture*):** This button will fill the selection box with the currently selected bloxel template and texture, as chosen in the Build menu.

- **Fill With Air:** Shortcut for directly filling the box with AIR bloxels.
- **Fill With Default Bloxels:** Shortcut for directly filling the box with standard bloxels.
- **Texturize (*Texture*):** Only changes the texture of the bloxels inside the selection box and not the templates.

COPY&PASTER

Copy or move a big cuboid of bloxels around. Create the selection (source) cuboid by using the Ctrl key. The Copy&Paster does **not** copy side data (created by the TEXER) currently, only the raw bloxel templates and textures.



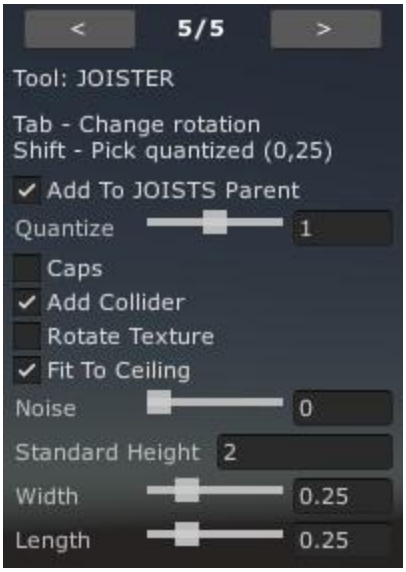
- **Pick Inner Bloxels:** Either pick the inside of a bloxel, or the outside, when creating the selection cuboid. You can use the 1 or 2 key to switch the picking mode.
- **Rotation:** Rotate the created copy (the target) around the Y axis. Use Tab and Shift+Tab to quickly change the current Rotation.
- **Move Selection To Copy:** If activated, the selection cuboid will be moved to the created copy.
- **Replace Source**: If activated, the source bloxels will be replaced by bloxels of the currently selected bloxel type and texture. This is like cut&paste instead of copy&paste.
- **Copy Air:** Either copy all AIR bloxels to the target, or don't.
- **Overwrite Non-Air:** Either replace all target bloxels that are not AIR with the source, or don't.

Only visible if there's a selection box:

- **X/Y/Z:** The position and size of the current selection box can be changed directly here.
- **Pos/Size / Min/Max:** Toggle between position and size display or minimum and maximum display of the box.
- **Clear Selection:** Remove the current selection box.

JOISTER

Create joists, beams, pillars of various lengths by pressing the left mouse button while hovering bloxel geometry. The joists use bloxel textures and are generated dynamically. After creation, you can manipulate and delete joists just like any other game object. (For that you need to deactivate Bloed temporarily.) Be aware though that a joist is belonging to a BloxelLevel, so you shouldn't change its place in the hierarchy.



- **Quantize:** Switch between a snap setting of 0.125 (**0**), 0.25 (**1**) and 0.5 (**2**) when placing a joist with Shift pressed.
- **Caps:** If activated, quads get added at both ends of a joist.
- **Add Collider:** If activated, the joist will automatically have a `MeshCollider` attached.
- **Rotate Texture:** If activated, the texture applied to the joist will be rotated by 90 degrees. Use Tab to quickly toggle this option.
- **Fit To Ceiling:** If activated, a joist added to the scene will try to adjust its length so it fits between floor and ceiling. If no ceiling is found, or this option is deactivated, then the standard height will be used.
- **Noise:** Adds a noise factor to the joist's vertices, so it looks crooked.
- **Randomize Noise Seed:** (only visible if Noise is > 0) If activated, the noise factor's seed value is set to a random value instead of just 0.

- **Width:** The width of the joist, between 0 and 1.
- **Length:** The length of the joist, between 0 and 1.

The 3D Cursor

The cyan 3D cursor is always at the same place as the camera pivot of the Scene view. If you press Alt+LMB while dragging the mouse the camera orbits around this pivot. This is often a more desirable way to rotate the editor camera than just using RMB. Bloed (as long as it is active) expands the F hotkey to quickly move the 3D cursor (and thus the camera) to the bloxel that is currently hovered by your mouse cursor, without having any GameObject selected.

BloxelLevel

BloxelLevels in the Hierarchy have a green (active) or yellow background. If you select a yellow BloxelLevel, you can make it the active one by pressing **Set Active** in the Inspector. For quickly switching between BloxelLevels, you can activate each by pressing Shift + left mouse button while hovering its bloxel geometry. (Only if Bloed is active.)

- **Recreate On Start:** Recreates the geometry of a BloxelLevel when the game starts.
- **Settings:** The LevelSettings of this BloxelLevel - you can change them, but expect strange results if the Standard Bloxels are different.

BloxelProjectSettings

In order to use Bloed you need to have an asset of type BloxelProjectSettings, for the whole project. It mostly defines the TextureAtlasSettings to be used in an array called **Tex Atlases**. You *must* have one BloxelProjectSettings asset in the folder *Bloxels/Resources/Settings* with the name ProjectSettings. It should be there already, but if it's not in your project, create it with right click inside the Project window and then choose Create → Bloxels → BloxelProjectSettings.

For debugging purposes there is a toggle **Show Debug Info/Settings**. It will display some internals in the BloxelLevel and BloxelChunk components, and also add a few more buttons in the Helpers tab, which are especially useful to check and fix inconsistencies in the bloxel data.

BloxelLevelSettings

You also need at least one BloxelLevelSettings asset. It defines the standard bloxel types and textures of a BloxelLevel. If there's no BloxelLevelSettings asset, create one with right click inside the Project window and then choose Create → Bloxels → BloxelLevelSettings. Use the file by assigning it to the **Settings** field of Bloed. It lets you define the following options:

- The BloxelLevelSettings asset needs at least one entry in the **Standard Bloxels** list. Each entry defines a texture (can be None) and if it's AIR (empty) or CUBE (full).
- The **Standard Method Type** defines how to use the Standard Bloxels. This can reduce the amount of bloxel data per scene, especially if you create a more open level; and is useful for levels that can be dynamically changed by the player.
    - **Single Bloxel:** Only entry 0 of the list is used, for all bloxels.
    - **Second Is Floor:** Entry 0 is used for all bloxels above the floor (i.e. bloxels with the Y coordinate being 1 or greater), entry 1 is used for the floor and below.
    - **Custom:** Define your own standard method. This needs to be in a script implementing the `Bloxels.IBloxelCustomStandardMethodProvider` interface. Have a look in the *Bloxels/Examples* folder to see how to use this.
- Be aware that floors and such won't be visible by itself, only if the chunks are generated. Even when "Second Is Floor" is set, the scene is empty at first. Only if you place a bloxel at e.g. 0/0/0, then you will see the generated floor. (This might be improved in a later version.)

You can also change several settings for each generated BloxelChunks:

- **Chunk Layer**
- **Tag**
- **Editor Flags**
- **Shadow Casting Mode**

TextureAtlasSettings

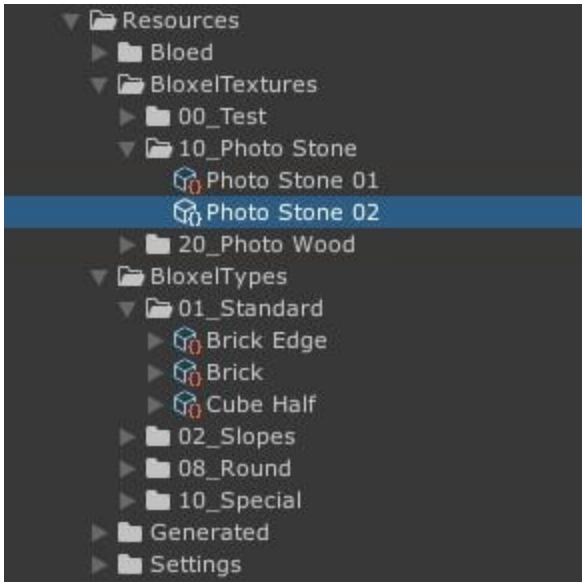
You need assets of type TextureAtlasSettings in order to define texture atlases that will hold all the bloxel textures. Create it with a right click inside the Project window and then choose Create → Bloxels → TextureAtlasSettings. Add each of your TextureAtlasSettings assets to the BloxelProjectSettings. It lets you define the following options:

- **ID:** This will be used to name the generated textures.

- **Material:** The Unity Material to be used. It is important that each TextureAtlasSettings asset will be assigned an individual Material, not used by other TextureAtlasSettings.
- **Properties:** A list of names based on the properties of the Material. "_MainTex" is the standard property, which is the diffuse texture usually (for the built-in render pipeline - for URP you will have to change this to "_BaseMap"). If you want to add normal maps, add a "_BumpMap" property to the list and change the shader of the material accordingly. The amount of properties defines how many textures a BloxelTexture can take.
- **Atlas Compression / Atlas Filter Mode:** How the generated atlas textures will be compressed and which filter mode they will use.
- **As Tex Array / Tex Array Size:** (new in v1.2) Instead of a texture atlas a texture array will be created. While bloed doesn't utilize one of the main advantages of texture arrays (being able to tile the texture layers), using a texture array can help with ugly artifacts in the distance. Be aware that textures inside a texture array always have the same dimensions. This is why BloxelTextures have an additional setting of how textures are cut (if they're divided into several blocks).

Assets and Folders

You need to have two folders inside one (or several) **Resources** folder(s): ***BloxelTypes*** and ***BloxelTextures***. If they don't already exist, create them. Both should have at least one subfolder, but usually several - those will be the categories (called **"shelves"**) of each bloxel type and texture. For example, the folder *Resources/BloxelTextures/Stone* could contain all your stone-y textures, and will be inside a shelf "Stone" in Bloed. To sort your shelves, use an underscore in the folder's name, like this: *10_Stone* - the name of the shelf will still be *Stone*.



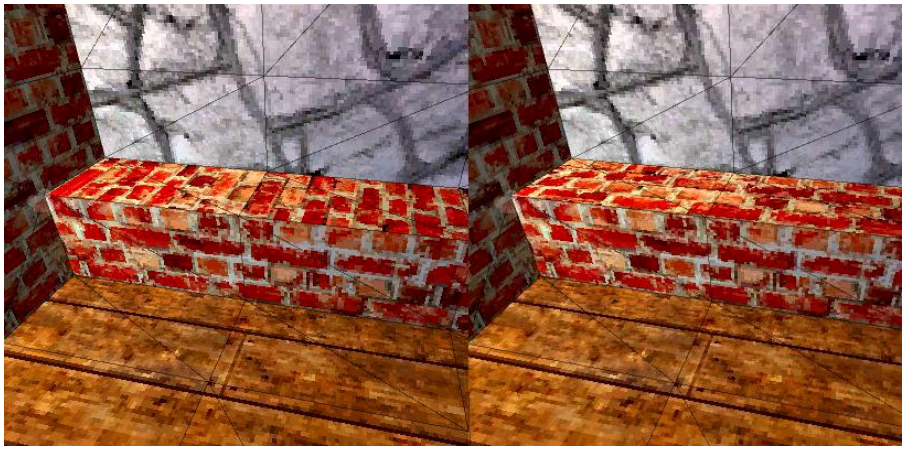
BloxelType

Create them by right clicking inside the Project window and then choosing Create → Bloxels → BloxelType. A BloxelType contains info about a type of bloxel. You do not need to define the standard AIR (empty) and CUBE (full) bloxels, they will be generated automatically. After creating a new BloxelType you have to use the “Init Bloxels” button under the “Initialize” tab of Bloed.

- **ID:** Give it a unique ID. This will be used to identify the type for serialization, so you can move it around without breaking the project.
- **Short Desc:** If it’s not empty, this will be visible in the Bloed type list.
- **Shelf:** Generated by using the folder’s name.
- **Mesh:** A mesh of size 1x1x1 (check the importer’s scale settings!) and the pivot in the center.
- **Collider Type:** Use this if you want to change the bloxel type’s collider. For example, stairs use the slope collider to make the geometry less complex, and to make a character controller’s movement smoother when walking up- or downward..
- **Possible Rotations:** The rotations possible for this bloxel type. Each row represents a “direction”. The checkboxes are for 0°, 90°, 180°, 270° bloxels rotated around the world Y axis of a certain direction. Note that mirrored bloxel types are currently not supported. Imagine the letter “R”. It can have 24 distinctive rotations, for example by turning it on its head and rotating it 90° around the Y axis, or by rolling it 90° clockwise, and so on. The letter O has much less distinctive rotations, because it’s symmetrical on several axes,

and thus needs less rotations generated. Each bloxel type's rotation is called a **bloxel template**.

- **Inner UV Dir Handlings:** All bloxel types apart from AIR and CUBE have inner mesh data which has to be uv mapped in a certain way; you can choose which one. *Box Mapping* works well with bloxel types with faces only looking directly into the cardinal directions; but as soon as you have slopes it gets a bit more complicated and you have to choose a *Preferred Direction*. For round surfaces it is recommended to use the Type *Keep UVs*, to prevent distorted textures.
  The option **"Add After"** lets you generate a second set of possible UVs (choosable with the Texer tool). This is needed for example for "brick bloxels", which you would box map. But to allow the textures to follow the natural direction, it's a good idea to have a second set of UVs that don't *Rotate With The Template*.



UV set 0 vs. UV set 1 for the "brick bloxels" (brick bloxels have a size of 1.0 x 0.5 x 0.5)

BloxelTexture

Create them by right-clicking inside the Project window and then choosing Create → Bloxels → BloxelTexture. A BloxelTexture contains info about how a texture should be used by the bloxels. After creating a new BloxelTexture you have to use the "Init Bloxels" button under the "Initialize" tab of Bloed.

- **ID:** Give it a unique ID. This will be used to identify the type for serialization, so you can move it around without breaking the project.
- **Shelf:** Generated by using the folder's name.
- **Tex Atlas:** Which texture atlas should be used, as defined in the BloxelProjectSettings. This is interesting for textures using a special shader (e.g. with transparency).

- **Flags / Neighbours:** With this you can define which bloxel should clip its polygons to its neighbour (otherwise, with transparent textures, you'd get visible holes in the geometry).
- **Generate Collider:** If deactivated, no collision geometry for bloxels with this texture will get generated.
- **Transparency:** Only to be used for textures inside a material that supports transparency.
- **Noise Strength / Noise Scale:** The higher Noise Strength is, the more each vertex of a bloxel with this texture will be distorted randomly. While Noise Scale defines the amplitude, Noise Scale defines the frequency. Attention: only the bloxel's texture noise will be used, side data is disregarded. Please refer to the Problems/Limitations section below regarding Noise.
- **Base Texture:** The main texture to be used. The texture **must** be *Read/Write Enabled* - check the importer options. If the texture is in a folder with a path that contains the strings "bloxel" and "texture" (e.g. *Assets/Bloxels/Textures/Bricks/bricks_01*), then it should be imported correctly already. Attention: Textures will not automatically be included in the compiled standalone even though they're referenced in a BloxelTexture. The reference is lost in the standalone build. *This might change in future versions.*
- **<List of Material Properties>:** For each additional property defined in the BloxelTexture's Texture Atlas it is possible to add another texture. The Base Texture refers to the first one, usually _MainTex.
- **New Pixel Size: (**Only available if the texture is NOT part of a texture array.) If you change it to a value different from 0, this will be the size of the texture inside the generated texture atlas; it uses Bilinear Scaling for changing the size. For convenience, the original texture dimensions will be displayed, and the texture size inside the atlas (based on your settings plus the padding applied to the sides).
- **Block Count:** Changes how many bloxels are needed to show the texture fully once. For example, if the block count is set to 4x4, the texture repeats only after four bloxels in each direction. If it's set to 1x1, the texture is mapped directly to a bloxel - and the tiling will be very noticeable.
- **Cut To Singles Textures:** (Only available if the texture is part of a texture array.) If disabled, the texture will only appear once in the array and blocks will stretch it. If enabled, the texture uses several layers inside the array, virtually increasing the pixel size of the complete texture.
- **Composition**: By setting **Top**, **Sides**, **Bottom** and/or **Inner**, the BloxelTexture on the respective bloxel side will automatically be set to a different one. This allows for easy creation of blocks that have, for example, a top side with grass and dirt for everything else.

- **Tags:** Define any amount of tags that can be used in your own scripts. For example, tags could let you define the “material” of a texture, so that footstep sounds are different on ‘wood’ than on ‘metal’.
- **Variables:** Define any amount of variables (strings, integers, floats and bools) that can be used in your own scripts. This is basically a more powerful tag system.

Additional Information

- ***Editor***
  - If you have Unity’s **hand tool** activated (Q), you won’t be able to place/delete any bloxels. Just switch to another tool, e.g. to the Move tool (via W).
  - You should activate the Scene window (by clicking inside) before you use any of Bloed’s **hotkeys**.
  - In order to ensure correct picking in the editor, bloed uses a different **collision mesh** (which ignores all BloxelTextures’ noise values and all BloxelTypes’ ColliderType) when Unity’s play mode is inactive. When the scene starts (in the `OnEnable()` method of a BloxelLevel) the correct collision mesh will be generated, delaying the beginning a bit.
- ***Problems/Limitations***
  - While the BloxelLevels and BloxelJoists work with **prefabs**, there are probably a lot of edge cases where the system might produce errors. Try to keep a flat hierarchy and probably don’t add more than one BloxelLevel to a prefab.
  - You are not allowed to edit prefab instances’ bloxels inside a scene. Either open the prefab in the prefab view, or unpack it.
  - To mitigate problems with broken geometry, situations where vertices sit on edges inside a side (e.g. when the sides of two opposite slopes are adjacent) do not use noise values of BloxelTextures.
- ***Optimization***
  - The **file size** of scenes can get pretty big because all the bloxel data plus the created meshes are saved inside. If you want/need to optimize the file size, remove the mesh data (via the button in *Helpers*) before saving the scene and uploading it to a repository.
  - To optimize **rendering**, you will most likely have to resort to static batching and occlusion culling, because even though bloed uses texture atlases, there usually

are a lot of draw calls (depending on the render method). Other tools for optimization are available too, for example Mesh Combine Studio 2 (paid).

- **Universal Render Pipeline (URP)**
  - Switching to URP is fairly easy as there is only one material (in the folder RatKing/Bloxels/Materials) out of the box, and you can choose any shader you like. The only additional step is changing the first property of your TextureAtlasSettings (in RatKing/Bloxels/Resources/Settings) from **"_MainTex" to "_BaseMap"**.
- ***Upgrading***
  - As Bloed is still in development and features can be added and/or changed in future releases, always make backups before **upgrading**. Also make sure that an upgrade doesn't overwrite your settings, atlases, BloxelTextures and BloxelTypes.
- Texture arrays
  - As Unity doesn't support texture arrays natively you have to use a special shader in order to support them in your bloxel-based level. In the *Examples/Shaders/* folder, there are two shaders that can be used as base for this - one for URP and the other for the built-in render pipeline.

API

You can manipulate bloxels during gameplay by using the following static methods.

- class **`RatKing.Bloxels.BloxelUtility:`**
  - `static BloxelLevel` **`CreateLevel`**`(BloxelLevelSettings settings, int chunkSize)`
    Create a new bloxel level, with the specified level settings and chunk size (standard is 8, which means each chunk consists of 8x8x8 bloxels).
- class **`RatKing.Bloxels.BloxelLevel:`**
  - `Bloxel` **`GetBloxel`**`(Position3 posAbs)`
    Returns the bloxel at `posAbs.`
  - `bool` **`ChangeBloxel`**`(Position3 posAbs, BloxelTemplate template, int textureIndex)`
    Changes a bloxel at `posAbs` to the desired `template` and `textureIndex`. To remove the bloxel, set the `template` to null or the standard template.
  - `bool` **`GetBloxelTexSide`**`(Position3 posAbs, int faceDir, out SideExtraData extraSideData)`

Returns true if there's side data for a bloxel at `posAbs` in the wanted direction. The `faceDir` is a number between 0 and 6 (top, back, left, bottom, front, right, and inner).

- `SideExtraData` **`ChangeBloxelTextureSide`**`(Position3 posAbs, int faceDir, SideExtraData newData)`
  Changes the side data of a bloxel at `posAbs`. Returns the former side data, if changed.
- `SideExtraData` **`RemoveBloxelTextureSideData`**`(Position3 posAbs, int faceDir)`
  Removes the data at the `faceDir` at `posAbs`. Returns the side data that existed previously (if any).
- `bool` **`RemoveBloxelTextureSideData`**`(Position3 posAbs)`
  Removes all side data of the bloxel at `posAbs`. If there was no side data it returns false.
- `void` **`UpdateSeveralChunksStart`**`()`
  `void` **`UpdateSeveralChunksEnd`**`()`
  Use these if you change several bloxels in one go, so the meshes get only (re)generated once.