

Isoliner3D - 3D viewer, bed and block model

Contents

Introduction	1
Installation	2
Quick start	2
The viewer window	2
A raster layer: the settings	3
Bed bodies	5
Vector layers: elevation, boreholes, the section	6
Boreholes	6
The section plane	7
The section drawing on the ribbon	8
Polygons with Z: bodies, outlines, prisms	8
Querying the scene by a click	9
Clipping the scene, markup, views	9
The Bed and block model group	10
1.01 Assemble a bed grid	10
1.02 Bed calculator	11
1.03 Bed grid to a block model	11
1.04 Surfaces to 3D (meshes)	11
1.05 Domains to a bed band	12
1.06 Reserve difference (write-off)	12
1.07 Create sample data (demo)	12
The neighbouring plugins	12
Typical situations and solutions	13
Appendix. The multiband grid convention	13

Introduction

Isoliner3D is a 3D window of its own for QGIS. It shows project rasters as surfaces, multiband grids as watertight bed bodies, boreholes as coloured stems and polygon layers carrying a Z elevation as volumetric bodies. The module does not depend on the built-in QGIS 3D view or on Qt3D: the render runs on pyqtgraph and PyOpenGL bundled with the plugin, nothing has to be installed.

The second part of the module is the **Bed and block model** tool group in the Processing toolbox: bed grid assembly, reserve calculation, the block model, domains, write-off, mesh export. It has a chapter of its own below.

Interpolation, kriging, isolines, relief and cross-sections stay with the Isoliner plugin, of which Isoliner3D is a companion. Isoliner3D reads ready grids, computes volume and reserves from them and shows the result in three dimensions.

Requirements: QGIS 3.16 or newer, Qt5 or Qt6. No external dependencies.

Installation

Plugins - Manage and Install Plugins - Install from ZIP, pick isoliner3d.zip. No QGIS restart is needed.

An **Isoliner3D** toolbar with a single button appears, together with the **Plugins - Isoliner3D - 3D surface viewer...** menu entry. If pyqtgraph or PyOpenGL are unavailable for any reason, the entry is not created and the reason is written to the QGIS message log, section Isoliner3D.

Quick start

1. Open a project that already contains rasters: roof and bottom grids, relief, multiband bed grids.
2. Press the **Isoliner3D** toolbar button.
3. On the **Layers** tab tick the rasters you need and press **Update the scene**.
4. Raise the **Vertical exaggeration** until the structure reads, 3 to 10 is usually enough for gently dipping beds.
5. Spread the surfaces with **Z spacing** so that the pile unfolds into a stack.
6. Save the frame with **PNG snapshot...**

The viewer window

The layer list is on the left, the scene on the right. The scene rotates with the mouse and zooms with the wheel. Large grids are thinned automatically: the vertex budget is counted for the whole scene and shared between the ticked layers, so a single surface gets more detail than a pile of ten.

The list is one, rasters and vectors together, as in the QGIS layer tree. The layer type is marked in the row, because the set of properties depends on it. The first row is a pinned **Scene**: the scene-wide settings are the same kind of list object as a layer.

The tick includes a layer in the scene and acts at once, without a rebuild. The properties open on a double click or with the right button, the properties window is not modal and changes its content when another row is selected. Editing any property rebuilds the scene by itself, and on a heavy scene the automation can be switched off by the **Update automatically** tick in the scene properties, which brings back a manual rebuild button.

A panel of icons sits over the scene at the top left: top view, parallel projection, contour drawing, markup visibility, clip removal, saving the contour as a layer, copying and saving a snapshot.

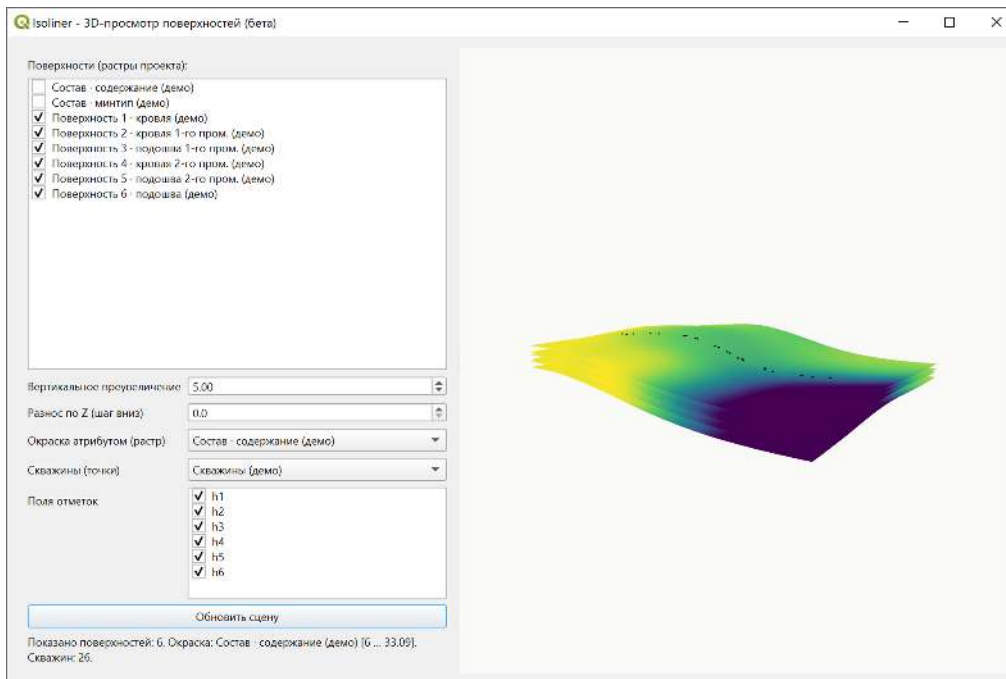


Figure 1: A stack of surfaces coloured by an attribute grid. The scale bar with the range sits under the buttons.

A raster layer: the settings

The list shows all the project rasters. The **Filter layers...** line narrows the list by a substring, the **All** and **None** buttons check and uncheck the rows visible after the filter. A scene set is assembled by hand in seconds even in a project with dozens of rasters. The checks survive a list refresh, so adding a new layer to the project does not reset a scene that is already composed.

Under the list is the **Layer settings** panel for the selected row. The settings are individual per layer and are remembered for the session.

- **Mode:** Auto (a multiband grid is drawn as a body, a singleband one as a surface), Surface (forced, any band used as heights), Bed body.
- **Elevation band (Z)** - a drop-down of this raster's bands with the names taken from the band descriptions.
- **Colouring** - a single list: Palette, Custom colour, then the layer's own bands by name, then the project rasters. Picking an external raster enables the **Attribute band**, and Custom colour enables the swatch to the right of the list. A click on the swatch opens the colour picker, the colour lives in the layer settings.

A texture is set in one of two ways. The **Project map (texture)** entry takes **all the visible layers of the tree** and renders them as a stack: handy when you need exactly what the map shows, but the extra visible layers get onto the texture too. The **Texture: layer name** entries at the end of the list take exactly one layer, and its visibility in the tree does not matter. The second way is more precise and lets different surfaces wear different maps.

Either way QGIS renders the layers over the extent of the surface itself, and the result becomes a texture. Anything will do, an orthophoto, a basemap, a geological map with all its symbology and labels, a hillshade. Reprojection is handled by QGIS, so the coordinate system of the map layers may differ from that of the grid.

The image resolution is set by **Texture side (pixels)** under the scene-wide settings. It does not depend on the mesh density, so the thinning of large grids does not affect the map detail. The

price is video memory: a 4096 by 4096 texture takes 64 MB.

The texture colour is multiplied by the shading from the surface normal. Without it the relief under the map would stop reading, because the light and shade by which the eye recognises form would be gone. The scene grids themselves do not go into the texture: draping a surface onto itself makes no sense.

The colouring priority is this: the texture, then the layer's own band, then the external raster, then the palette. The scale is one per scene, the bar with the range appears under the buttons, no-data cells stay grey.

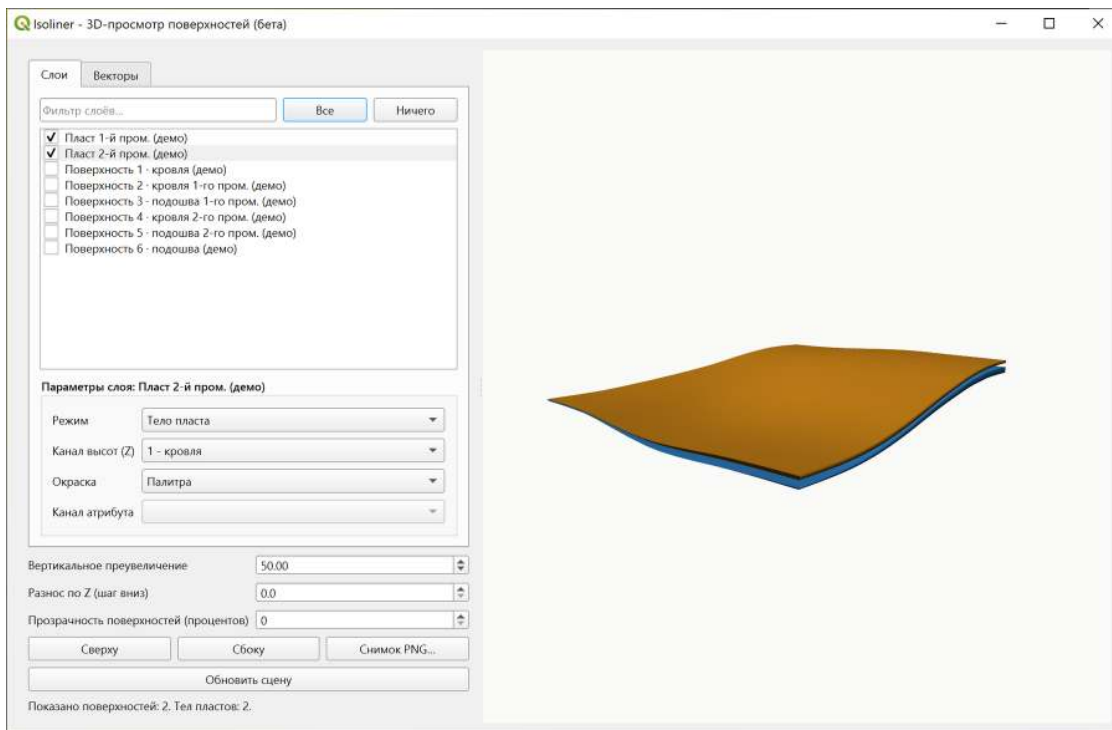


Figure 2: The **Layers** tab: the filter, the **All** and **None** buttons, a set of two bed bodies and the **Layer settings** panel of the selected layer.

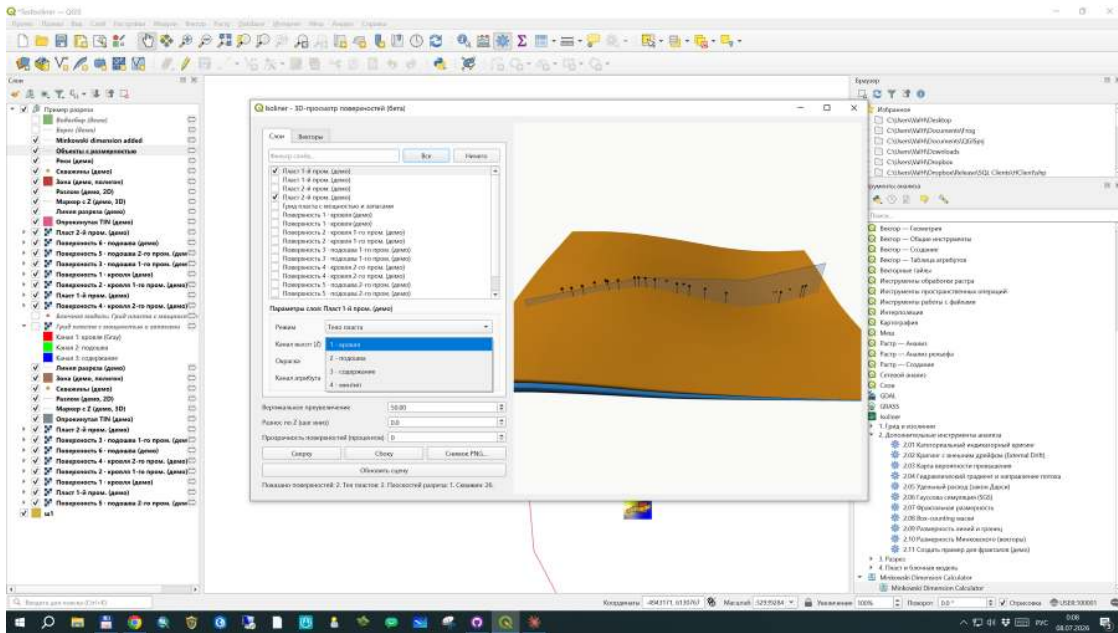


Figure 3: The band lists show the names from the grid descriptions: roof, bottom, content, mineral type.

Bed bodies

In the Auto mode a multiband grid is read as a body by the convention: band 1 is the roof, band 2 is the bottom. The volume is closed by a side skirt along the data boundary, which yields a watertight body that looks correct from any side and is cut correctly by the section plane.

Bed grids assembled by the Isoliner tools show their own band names in the lists: roof, bottom, then the parameter names. Bodies and plain surfaces live in one scene and obey the shared exaggeration and transparency settings.

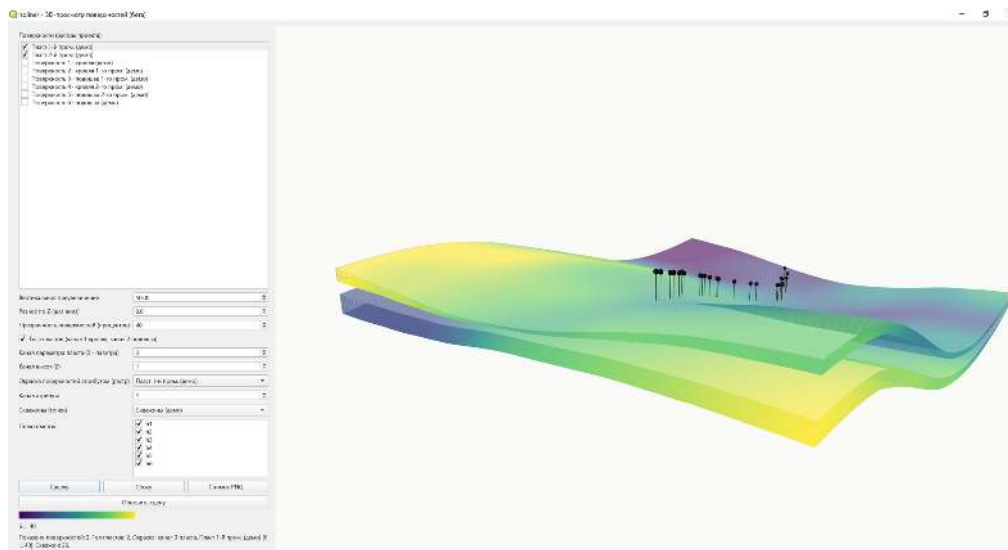


Figure 4: Two bed bodies, each coloured by its own grade band. Boreholes pierce the stack.

Vector layers: elevation, boreholes, the section

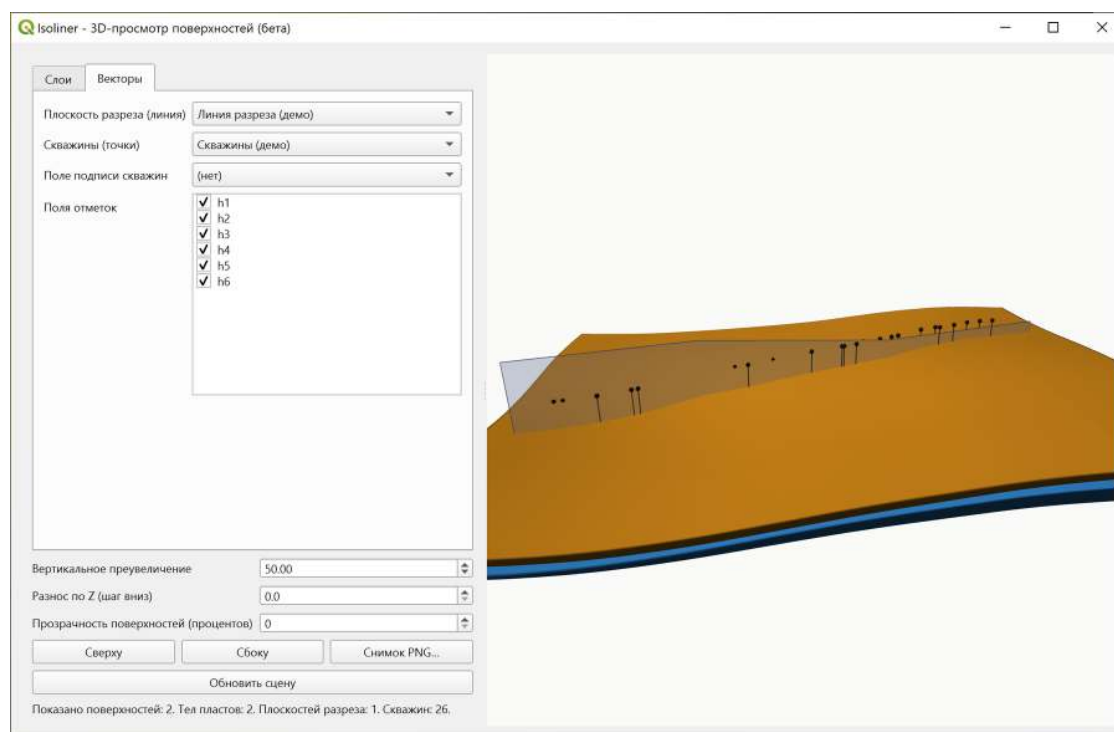


Figure 5: The **Vectors** tab: the section plane, the boreholes, the label field and the elevation fields. The scene shows the section ribbon with boreholes on a bed body.

Boreholes

Boreholes (points) takes a point layer. The list below holds the numeric fields, in which the horizon elevations have to be ticked. Fields named like h1...h6 are ticked automatically.

Every borehole is drawn as a stem of cylindrical segments between neighbouring elevations. The intervals are coloured by stratigraphic position, that is by the order of the ticked fields, so the same horizon reads in one colour across all the boreholes. A mast with a ball is placed above the collar: it lifts the collar above the roof by two percent of the scene span, and the borehole stays visible even where the stem runs inside an opaque body. The collar balls are drawn while there are no more than five hundred boreholes.

Borehole label field adds text above the masts. Fields named like name and well are guessed automatically, the **(none)** entry switches the labels off. The labels are thinned: if a labelled borehole is already nearby, the text is skipped, and a dense well stock stays readable. The cap is 500 labels per scene.

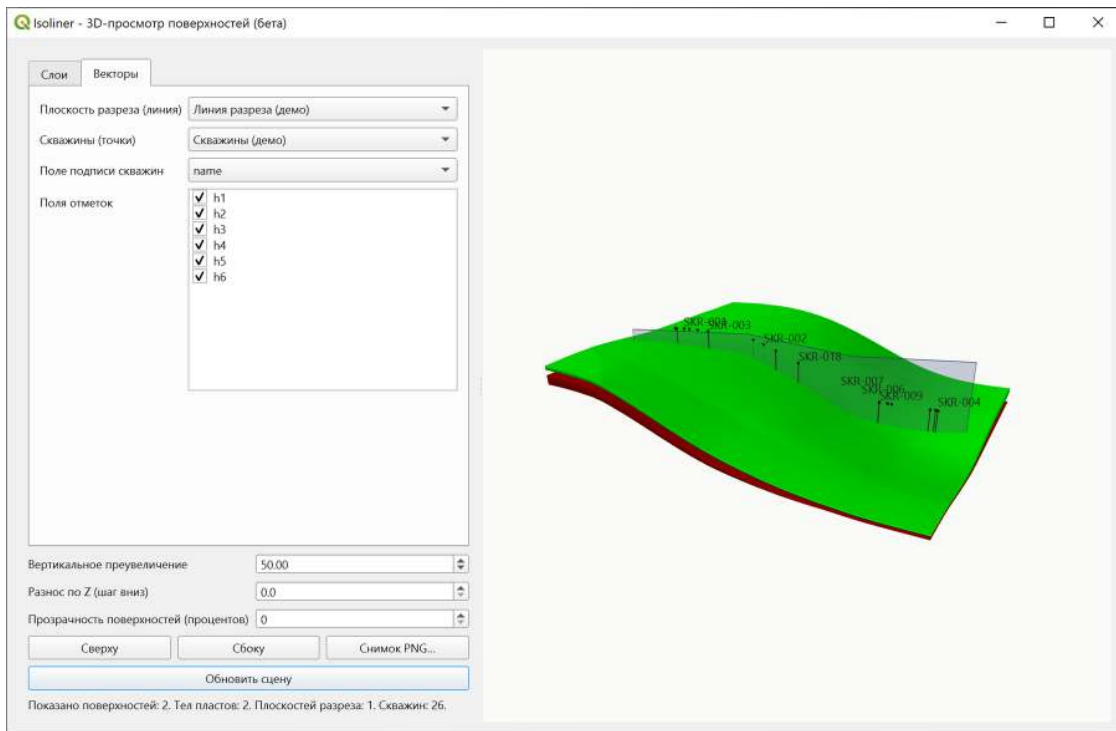


Figure 6: Borehole labels above the masts with automatic thinning. The **Vectors** tab with the label field on the left, the bed bodies coloured with custom colours.

The section plane

Section plane (line) accepts any line layer. The most convenient input is the **Section definition** layer produced by the **Section along a line** tool of the Section group in the Isoliner plugin: the ribbon takes its height range from the **zmin** and **zmax** fields. For an arbitrary line the ribbon is stretched over the scene span with a margin.

Polylines and several lines in one layer are supported, the bends are drawn by the vertices. A bright intersection trace runs along the ribbon over the surfaces, and for the bodies from the **Bodies** tab a section contour is drawn, that is the line where the vertical curtain along the section cuts the body.

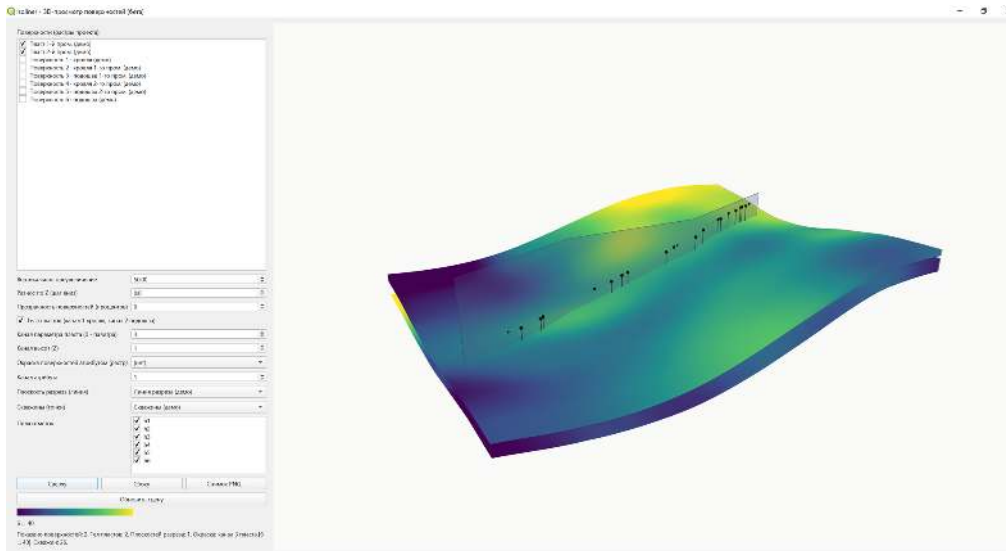


Figure 7: Bed bodies, boreholes and the section plane in one scene: the block model stitched with the section.

The section drawing on the ribbon

The **Section drawing (layer or group)** field dresses the ribbon in a real drawing. The Isoliner section tools build it in «distance along the line by elevation» coordinates, and the ribbon in the scene occupies exactly the same region of space, so the drawing lands where it belongs with no recalculation at all: distance runs along the ribbon, elevation across it.

Either a single layer or a whole group can be picked: a drawing usually has several layers (beds, boreholes, labels), and a group is taken with all its symbology in tree order. Layer visibility does not matter, the drawing lives in its own coordinates and is in the way on the map anyway.

No shading is applied to the drawing: it has to read as a drawing rather than as a lit surface. The resolution is set by the same **Texture side** field.

Several sections are shown at once. The Isoliner tools build the drawings for every line of the layer in one go and lay them out side by side, and every definition line carries its own fields: the section number `sec_id`, the name `sec`, the vertical exaggeration `vex` and the layout offset `ox`, `oy`. That is how the module finds the area of its own section on the drawing layer.

The extent comes from those fields rather than from the bounds of the layer features. A table of distances and azimuths hangs outside the drawing frame, and going by feature bounds would shift the image. What is worse, this would only show up for those who build with the table.

The names of the shown sections are printed in the status line.

When the `ox` and `oy` fields are missing from the definition, the drawing is not draped and the reason goes to the log: the section was built by an older version of Isoliner and only needs rebuilding. When the section number is there but the chosen drawing layers do not contain it, the drawing and the definition come from different builds and the ribbon stays without an image.

Polygons with Z: bodies, outlines, prisms

The **Bodies** tab shows polygon layers carrying a Z elevation as volumetric bodies right in the scene, next to the surfaces and the bed bodies. Polyhedral surfaces, TIN and any MultiPolygon Z layers will do. Tick the layers you want and press **Update the scene**.

The geometry of each feature is broken into triangles as a separate mesh and coloured on its own, so a suite of several beds comes out multi-coloured. The same vertical exaggeration and transparency apply to bodies as to surfaces, so a polyhedron and a stack of horizons read at one scale.

This is the only way to see an overturned fold or an overhang in three dimensions. A grid cannot describe such a form at all, because a grid keeps one elevation per node, while a polyhedron or a TIN keeps a true surface with several Z values above one point of the plan.

Querying the scene by a click

A click on a surface or a body queries the model. A ray is cast from the camera through the cursor, the nearest intersection with the surface is found, and the status line prints the layer name, the point coordinates and the values of all the bands by name, plus the thickness for a bed. The hit is marked with a red ball until the next click or a scene rebuild.

Dragging is separated from querying: rotating the scene with the mouse works as usual, the query fires only on a click without movement.

Clipping the scene, markup, views

The scene can be cut so that only the part you need is left. The clipping contour is set in the scene properties: any polygon or line layer of the project, or one drawn right in the scene. The **The piece** row sets what to show.

For a polygon it is **Keep what is inside** or **Remove what is inside**: a slice of the cake or the cake without the slice. For a line it is **To the left of the line**, **To the right of the line** and **A corridor along the line**. The corridor takes a band of a given half-width on both sides of the profile, and that is usually more useful than a bare section: the data stay next to the line, and it is visible how the structure changes away from the profile. The half-width is entered on the panel over the scene, next to the drawing icons, and is shown only in the corridor mode.

The edge follows the contour rather than the bounding rectangle, and the holes of the contour stay holes. It is not only the surfaces that are cut: lines break where they leave the piece, points and boreholes are selected by their position, bodies by their centre.

Drawing in the scene. The contour icon turns on the markup mode: a click on the surface adds a vertex, a rubber band follows the cursor, the right button or the neighbouring icon removes the last vertex. From there two ways. The icon with a tick closes the contour and clips an area by it. The polyline icon finishes a line and clips a corridor along it, switching to the right mode by itself.

The vertices are taken from the surface by a ray from the camera, so the markup lies on the relief and stays true when the scene is rotated. When the scene has no raster surfaces, say only isolines are shown, a vertex is taken from the level of the middle of the scene. The plan position of the vertices is what the clipping uses. The markup itself is drawn over the model and does not hide under folds.

The eye icon hides and shows the markup without removing the clip. The icon with a crossed-out contour removes the clip entirely and clears the sketches. The icon with a stack of sheets saves the drawn contour as a project layer, after which the Isoliner tools can use it.

Views. The icon of a frame with a crosshair sets the top view and turns on the parallel projection as well, because a plan with perspective is not a plan. The projection is also switched separately by the cube icon: in a parallel projection the scale is the same across the whole frame, and objects at different heights do not shift relative to each other.

Snapshot. The icon with two rectangles puts a frame of the scene on the clipboard, and so does Ctrl+C. The camera icon saves the frame to a PNG file. The snapshot size equals the size of the

scene window, so it is worth maximising the window before shooting.

The Bed and block model group

Besides the 3D window the module installs an **Isoliner3D** provider into the **Processing** toolbox, with a single group - **Bed and block model**. Its seven tools compute on NumPy and GDAL, they need neither kriging nor isoline building, so they work without the main Isoliner plugin.

The pipeline runs like this. A roof and a bottom are assembled into one multiband bed grid (**1.01**), the calculator derives thickness and reserves from it (**1.02**), the block model unfolds the grid into centroid points (**1.03**), and the difference of two models gives the write-off (**1.06**). Domains (**1.05**) add an area code band to the grid, the mesh export (**1.04**) hands the surfaces over in the 2DM format, the generator (**1.07**) creates demonstration bodies with Z and a test map for the texture.

An assembled bed grid is read by the 3D window as a body: band 1 roof, band 2 bottom. Compute it and look at it right away.

1.01 Assemble a bed grid

Assembles a multiband bed grid by the module convention: band 1 roof, band 2 bottom, bands 3 and on the parameters (grade, mineral type, any other numeric field).

The roof defines the grid of the result. The bottom and the parameters are resampled onto it bilinearly, so the source grids may differ in cell size and extent. The band names are written into the raster band descriptions: roof, bottom, then the names of the parameter layers. That is why the 3D window lists read “content” instead of “band 3”.

Parameter	What it sets
Roof (raster)	the roof grid, defines the output grid
Bottom (raster)	the bottom grid
Parameters (rasters, band 1 is taken)	any number of parameter grids, the field is optional
Roof band, Bottom band	for multiband sources
Bed grid	the result

One run assembles one bed: one roof, one bottom, one output grid. A pile of three beds means three runs, or a single go through the **Run as batch process** button, where every table row sets its own pair of layers.

About the parameter list. The field is optional. Leave it empty and you get a two-band grid, a roof and a bottom. For viewing as a body, for the calculator and for the block model that is already enough.

The list is for the accompanying values tied to the same area: grade, mineral type, recovery factor, any numeric characteristic. The ticked rasters go into the grid as separate bands starting from the third one, band 1 is taken from each.

The point is that after the assembly a bed is described by a single file. Everything else then works off it: the calculator will ask for the grade band and compute the mean grade and the metal tonnage, the block model will lay every band out into its own attribute field, the 3D window will list the bands under their names in **Colouring**. That is exactly why the names come from the layer names: the lists then read “content” instead of “band 3”. It is worth tidying up the layer names in the project before the assembly.

The parameter grids may have their own cell size and their own extent, everything is resampled onto the roof grid bilinearly. It is the roof that defines the output grid, which is worth keeping in mind when choosing what to feed as the first input.

The **Roof band** and **Bottom band** fields live in the advanced section and are only needed when the source grids are themselves multiband. For ordinary singleband ones they stay at one and need no attention.

1.02 Bed calculator

Computes thickness, volume and ore tonnage via the density from a bed grid. When a grade band is given, the thickness-weighted mean grade and the metal tonnage are added.

The summary is taken over the whole bed area or inside a contour: the polygons of a mining block or a domain. Holes in the polygons are honoured. The result is a bed grid with the thickness and the per-cell ore reserve appended as bands, plus an HTML report with the summary.

Cells with a negative thickness, where the bottom ended up above the roof, are zeroed and counted separately. Their number is worth checking in the report: it means the surfaces intersect and the model needs a fix.

Parameter	What it sets
Bed grid	band 1 roof, band 2 bottom
Grade band	empty - compute without a grade
Ore density, t/m³	to convert volume into tonnage
Calculation contour (polygons)	limits the area
Bed grid with thickness and reserves	the result
Report (HTML)	the summary

1.03 Bed grid to a block model

Turns a bed grid into a block model: one centroid point per valid cell. The attributes are the cell row and column, the coordinates, the top and the bottom, the thickness, the volume, the ore tonnage and every parameter band under its name from the descriptions.

From there the ordinary QGIS vector machinery applies: expression filters, joins with external tables, the field calculator. The model grows new attributes without being rebuilt.

The **Vertical layers** parameter splits every column into N blocks between the roof and the bottom. Each block gets its own top and bottom elevations, a layer number and a share of the volume. The grade is copied into the sub-blocks as is, because it is not drilled out vertically. The reserve sum is preserved by the split, which makes a handy self-check: build the model with one layer and with five, the tonnage sums must match to the last digit.

The density comes from the number in the parameters or, when a **Density band** is given, per cell from the grid. The latter is what you need where the ore density varies across the area.

1.04 Surfaces to 3D (meshes)

Exports surface grids into mesh layers of the 2DM format (MDAL). Such layers are understood by the QGIS profile tool, the mesh calculator, the built-in 3D view and third-party software.

A vertical transform is applied to the elevations: an elevation is multiplied by the scale and shifted by the offset. The scale gives the vertical exaggeration, the offset raises or lowers a horizon. **Z spacing** pushes every next grid down by a constant step and turns a stuck-together pile into a readable stack. Thinning reduces the node count on large grids.

The layers are loaded into the project and get a 3D rendering automatically. Cells without data are skipped.

1.05 Domains to a bed band

Rasterises domain polygons into an extra band of the bed grid: every cell gets the code of the domain it falls into, zero outside the domains. The code comes from a numeric field of the layer or, when no field is given, it is the feature order number starting from one. The bands of the source grid are preserved, the domain band is appended last.

From there a domain behaves like any other parameter: the calculator works over the domain contour, the block model is filtered by an expression on the code. The domain contours must be in the same coordinate system as the grid.

1.06 Reserve difference (write-off)

Computes the difference of two block models over the cells with matching row and column: how much reserve was removed between the “before” and the “after” states. The chosen field is subtracted per cell, the ore tonnage by default. The result is points with the before, the after and the difference values. The total write-off is printed to the log.

This is the direct route to operational write-off: the model before the chambers were mined out minus the model after, and the sum of the difference over a contour gives the written-off tonnage that goes into the statement. Both models must be built from the same grid, otherwise the row and column split will not match.

1.07 Create sample data (demo)

Creates demonstration data, so that you can check the display on your own QGIS build without touching the working layers. The variant is set by the **Example** list.

Bodies with a Z elevation. A bed body is a watertight shell of a roof, a bottom and a side skirt. A suite is a stack of folded beds, each loaded as its own layer with its own colour. There are also a cube and a tetrahedron. The plan position and the size come from the extent, vertically the body runs from the base elevation up to the base plus the thickness.

The geometry type is flat, so the elevation is not visible in the 2D view, the Z range is printed to the log. The body itself is best looked at on the **Bodies** tab of the 3D window. A native PolyhedralSurface Z is available from QGIS 3.40, on older builds the output degrades to MultiPolygon Z. The TIN flag yields a triangulated surface.

Map (raster for a texture). A three-band image to check the draping of a texture. The extent is best set by the **Map: by the extent of a grid** field, then the map lands exactly on the surface bounds.

The image is deliberately a test pattern rather than a pretty map. Draping fails in three typical ways, and this map shows each of them at once. A vertical flip shows up in the differing corner marks, a shift or a skew in the graticule, a stretch along one axis in cells that stop being square. On a pretty map none of this is visible.

The check goes like this. Create a map by the extent of your grid, set the grid colouring to **Texture: Map (demo)** and update the scene. If the corner marks are where they belong and the graticule cells are square, the draping works correctly.

The neighbouring plugins

Isoliner3D does not work on its own. Three plugins cover the way from observation points to a volumetric model, and each does its own part.

Isoliner builds grids and isolines: kriging with variograms, minimum curvature, relief and catchments, sections with drawings, a geological bed model. It is the one that prepares the grids and belts shown here. The catalogue: plugins.qgis.org/plugins/grid_isolines.

Topoliner tidies the contours: dangling nodes, self-intersections, gaps and overlaps between adjacent polygons, thinning that preserves topology. It helps before belts and solids are assembled, because a torn topology on the map turns into a torn shell in three dimensions. The catalogue: plugins.qgis.org/plugins/topoliner.

Isoliner3D shows the result in three dimensions, cuts the model by a contour or a corridor, computes reserves from a block model and exports the scene to GLB.

All three work independently: installing the whole set is not required.

Typical situations and solutions

What you see	Cause	Solution
No menu entry and no button	pyqtgraph or PyOpenGL are unavailable, <code>is_available()</code> returned false.	Check the QGIS log, section Isoliner3D. This usually means a broken installation: reinstall the module from the ZIP as a whole, together with the <code>libs</code> folder.
An empty scene and a request to tick a raster in the status line	No layer is ticked.	Tick something on the Layers or the Bodies tab and press Update the scene .
The message about grids that could not be opened	The files of the ticked rasters are unreachable: moved, on a disconnected drive, or the layer was temporary.	Check the layer source in the properties and rebuild the grid if needed.
The surface looks like a flat pancake	The real elevation span is small compared to the extent of the area.	Raise the Vertical exaggeration .
A bed is drawn as a surface instead of a body	The grid is singleband, or the Mode is set to Surface.	Check the band count and set Auto or Bed body.
The body is inside out, the roof below the bottom	The bands are swapped, the convention wants band 1 roof, band 2 bottom.	Reassemble the bed grid with the bands in the right order.
No boreholes are visible	No elevation fields are ticked, or all the elevations are empty.	Tick the numeric elevation fields in the list on the Vectors tab.
Fewer labels than boreholes	The thinning is at work, the cap is 500 labels.	This is by design. Move the camera closer or feed a thinned stock layer.
The surface looks coarser than the source grid	The automatic thinning to 60 thousand nodes has kicked in.	This is by design. For detail on a fragment, cut the piece of the grid you need and feed it as a separate layer.
The section ribbon runs far beyond the pile	The line was fed without the <code>zmin</code> and <code>zmax</code> fields, so the ribbon is stretched over the scene span.	Feed the section definition layer from Isoliner.

Appendix. The multiband grid convention

A bed body is assembled from a single raster whose bands are laid out like this:

Band	Meaning
1	the bed roof, absolute elevation
2	the bed bottom, absolute elevation
3 and on	parameters: grades, thicknesses, mineral type, any numeric field

The band names are taken from the raster band descriptions, which is why the module lists read “content” instead of “band 3”. Grids assembled by the bed tools in Isoliner set the names themselves. For a third-party GeoTIFF the names can be written by any means that writes band descriptions, for example `gdal_edit.py` or GDAL from Python.

Cells without data must be marked with a nodata value. The module does not fill the gaps: a node without data drops out of the mesh, and the body is closed by a skirt along the boundary of the actual data.