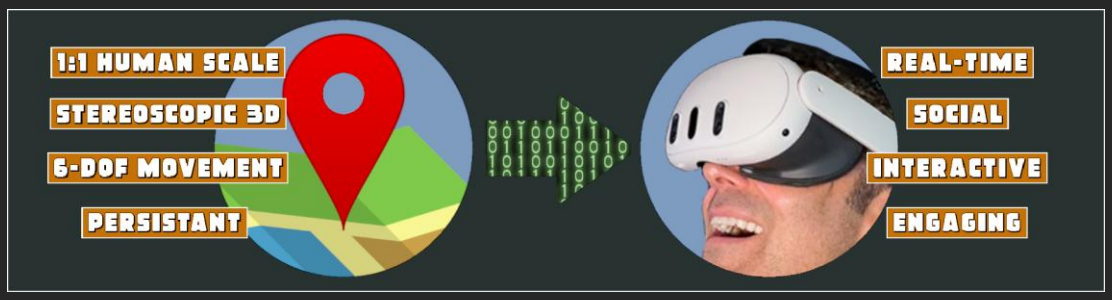
MAKE AN IMMERSIVE MAP OF YOUR WORLD

FAQ's * How to make a map * Part 1 : Give your map a title * Part 2 : Make a first map * Part 2A: The development environment * Part 2B : Make 3D environments with Blender * Part 2C : Make worlds with Unity * Part 3 : Get real-world data * Part 4 : Make 3D models with AI * Appendices

What's an immersive map?

Maps[1] give us an idea of where we're going. Immersive maps[2] show what it's like to be there, and let us share ideas of how we'd like it to be.

Maps come to life with the magic of immersive headsets[3]. Instead of imagining a place in our minds, *we experience with our senses*. Accurately presented environments offer a convincing expression of space-reality that looks + feels like the real thing. We can walk, talk + interact naturally within a digital recreation of buildings, contents, environment + landscape. Form + function can be replicated within reason.



They're free to make, use + share. What people can see + do is up to you. **It's a storytelling tool with which to inform + inspire**.

Why have an immersive map?

The ability to communicate clearly + completely, 'This is my world as I see it', or, 'This is the world as I want it to be' is something new under the sun.



[1] Maps are an evolving cultural tool for understanding the world.
[2] Immersive maps give the illusion that we're somewhere we're not.
[3] An immersive headset is technology that allows us to travel by immersion, eg. the Meta Quest3 (meta.com/quest-3).

There are myriad use-cases. Whether you're planning resilience adaptations for your community, taking bookings for your hotel or addressing the foundational infrastructure of the planet, an immersive map can help.

I've imagined a sustainable future + showcase ideas in a free map[4] where you play with generating a household's electricity mix + install wind turbines to see the effect on CO_2, bills + the view. **The complex problem of communicating change just got a lot easier**. I believe that inviting neighbours, stakeholders and decision-makers to an immersive presentation will help achieve consensus + green-light funding for all sorts of projects.

Sharing your map allows people to see the world from your perspective, to walk in your shoes. You are always in control of who can go where + when.

How might you + your visitors benefit from being inside a working recreation of your location?

How do I get an immersive map?

You can make it yourself (with help if needed), or have one made for you. Follow this step-by-step guide that encourages getting out into the world, recording what's there + making 3D models to arrange in a template map. You'll discover lots about your environment, learn new skills + create something genuinely useful.

Making a map is cost-free (you'll need access to a headset + computer). There's no credit-card required, no billing + no license to pay for, just the cost of powering the tech + your time. Consider a budget for bells + whistles, especially for commercial use.

If you want to collaborate on crafting a map with your unique identity, that's OK. Your map will be stronger with more people involved. Bring friends, family + neighbours together in a shared endeavour that pools the technical wizardry + deep local knowledge of communities + generations.

I can personally support you through + beyond this guide in guided sessions[5] filled with practical lessons, theory + challenges. You'll grasp the reasoning behind each action as well how it's achieved, resulting in accelerated learning + solid understanding.

No technical ability required! Commissions are accepted based on an agreed specification document (Appendix I). The addition of a dedicated digital artist on any development team is recommended.

What's an immersive headset?

An immersive headset is an apex interface for sensory experience of digital data.

It's a head-mounted device that replaces light from the world with light from a computer-generated display. The image *changes with our head position* to show what somewhere looks like *in real-time*, just as the light reflected into our eyes from the world changes as we look around. There's one image for the left-eye + one for the right, achieving familiar stereoscopic-3D depth. The brain intuitively understands what's going on. **We see a construct of a world around us that makes sense**.

The effect is what we call immersion, and it comes free with any headset. The trick, of course, is for the user to become actively engaged with the illusion + feel the pleasing level of sensory synthesis found in everyday life.

What's a Quest3?

[4] We Make Malvern (wemakeworlds/malvern).
[5] wemakeworlds.com/support

The Meta Quest3 (Quest3, Quest3s, Quest Pro) headset is accessible, affordable + available everywhere. It's the runaway success of the last few years in delivering immersive content to the masses. Suitable for older kids and adults, it's sold millions of units[6] and continues to impress with new features + functionality. Expect a Quest4 in 2028 at the very earliest.



The hardware OS[7] is based on Android[8], meaning we can write + run our own software on it.

Consumer-grade virtual-reality hardware has matured over the last decade thanks to advances in optics, processing power and free software. See the intro video[9].

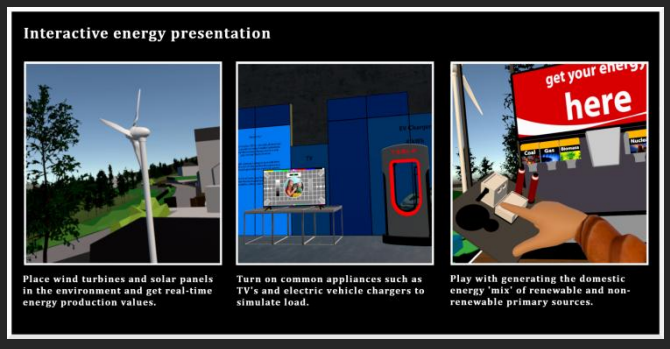
What's We Make Malvern?

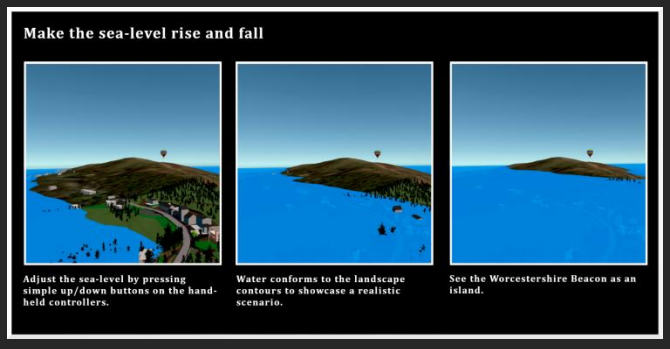
We Make Malvern[10] is a free immersive map of the Malvern Hills, UK, for the Meta Quest3 headset. It places you in a 7km-diameter landscape with a ~300m^2 high-detail interactive environment.

Use you own headset or get in touch to arrange a personal presentation. Let me know if you need assistance. There's an in-app tutorial covering movement + interaction.

Currently in open Beta, so you're free to visit anytime. Please request access from the Discord[11].

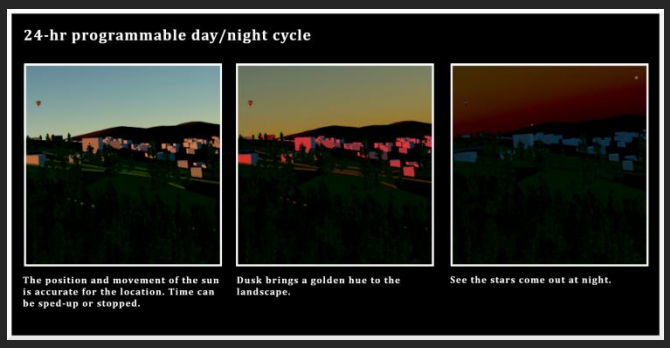


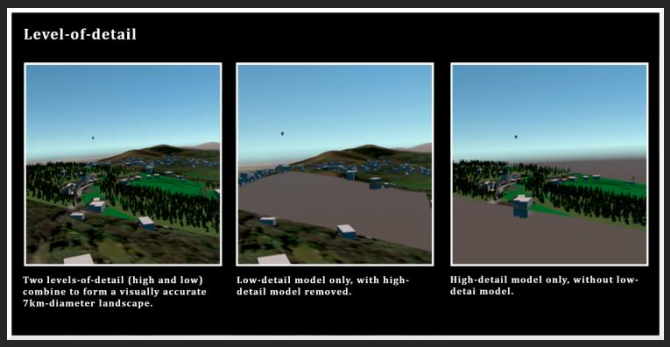


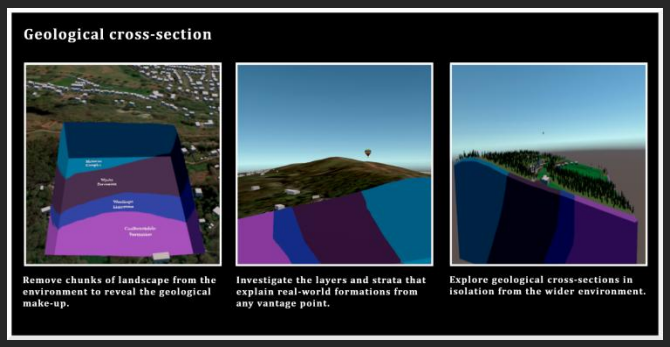


[6] Manufacturer does not post unit figures; a healthy ecosystem sustains sales.
[7] Operating System : software (computer code) that manages the physical hardware. The Quest headset uses Meta Horizon OS.
[8] Android : developed by Google, based on Linux + released as FOSS (free + open-source software). 3.9B+ users.
[9] tinyurl.com/2s3fw37p
[10] wemakeworlds.com/malvern.html
[11] discord.gg/p4tPcagsWX









What's We Make Earth?

We Make Earth[12] is an immersive map of the planet for the Meta Quest3 headset. Maps made with this guide integrate with other maps on a model of the Earth that becomes more detailed as maps are added.

Resources can be created, transported + consumed across maps. The change in resource distribution that achieves global equitable sustainability can be seen *from any individual's perspective*, wherever they are in the world, in real-time.

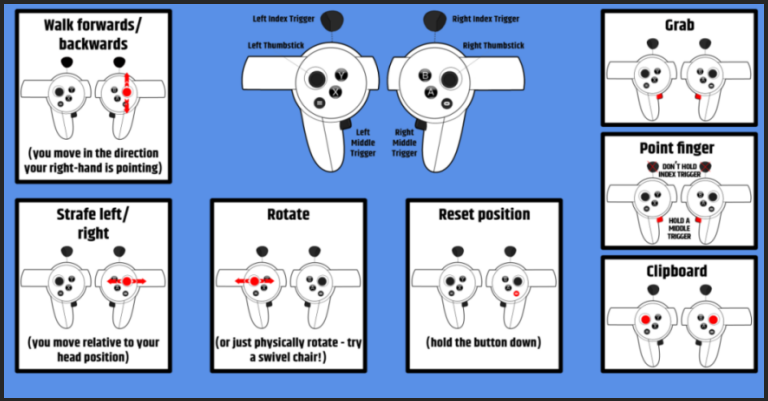
Currently tracking : food (calories) + energy (watts).

Currently in closed Beta. Please get in touch for more information.

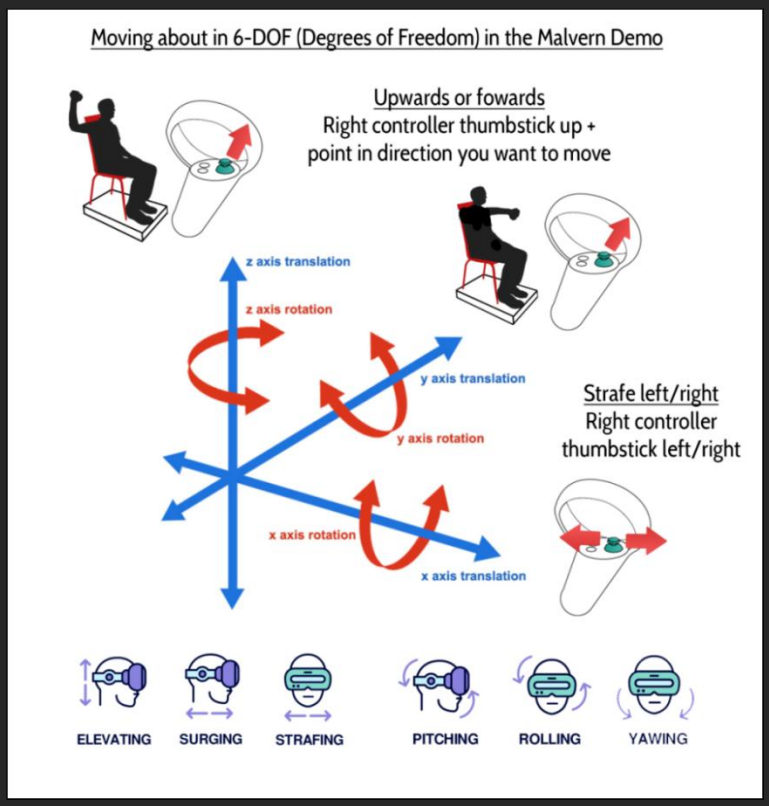
How to move around in an immersive map

Movement is achieved *by physically moving around*, or, as you can't see the real world, sat down holding controllers (a swivel chair works really well). Pushing forward on a thumbstick achieves forward movement *in the direction your hand is pointing*. You move sideways *relative to your head position*. This seems complex on paper but is intuitive in practise.

[12] wemakeworlds.com/earth



Control schema for We Make Malvern



How to make a map

tldr; a complete development environment + workflow for turning raw data into an immersive map.

In Part 1, you define the criteria for your map, but before deciding what *you* need, first understand what *the computer* needs. Part 2 gets you making a first map, proving the technology + skills are in place for Part 3, where you confidently collect the *right* data for *your* map.

Part 1 : Give your map a title

Any good map needs a title, purpose, extent + reference. Come back to this part after completing Part 2 + see if your answers are different.

The title of my map is : _____

The purpose is : _____

The low-detail landscape extent is : _____m diameter (circular disc)

The high-detail environment extent is : _____m^2 area

The reference is: _____ latitude, _____ longitude

Consider creating a full Specifications[13] document for your project.

Title : Keep it simple. Concentrate development around a single idea. Under 20 words.

Purpose : Decide what people will be able to see + do. Who is it for? What does it need to achieve? What are the criteria for successful completion? Under 200 words.

Everything that visitors see + interact with needs to look + feel like the real thing *as much as possible*. Hardware limitations, alas, force a lack of fidelity from real life, so we must consider what places, objects + features are most important. Every decision affects the workflow. For a first map, start small + scale up after.

Extent : The template map has two Level-of-Detail models (LOD's). LOD0 represents what will become your high-detail model with an area of $1km^2$. LOD1 is the low-detail circular landscape model with a ~7km-diameter. Feel free to experiment with LOD sizes depending on your use-case, bearing in mind that a 1000m x 1000m area is $1{,}000{,}000m^2$. To recreate only the ground surface, that's a million square metres of surface feature modelling. As a rule of thumb, the larger the high level-of-detail extent, the more time it'll take to create an accurate representation of the environment.

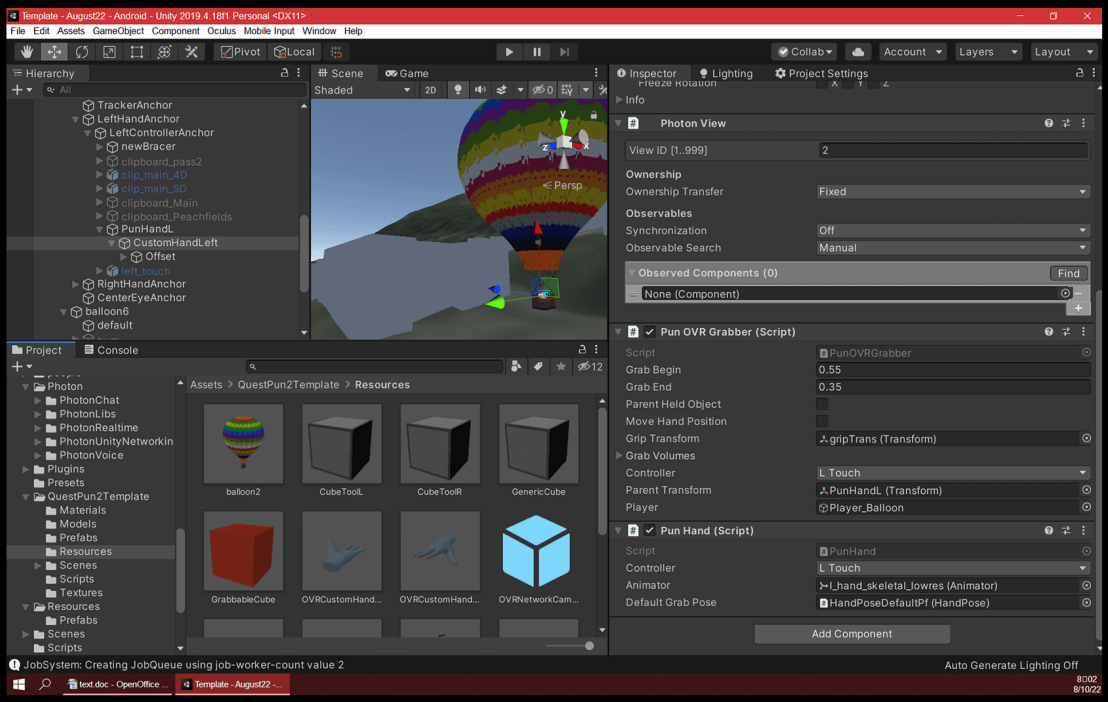
The boundary of a traditional 2D map is referred to in terms of it's breadth (from one side to the other, or the horizontal x-axis) + length (from top to bottom, or the vertical y-axis), giving us the area (breadth x length) of the map. Immersive maps show 3D space, meaning they can show things going on *underground + in the sky*. A third z-axis denotes height/depth. A position in 3D space is referred to by 3 co-ordinate numerals representing distances in metres from a centre point (x=0, y=0, z=0).

Reference : Find the latitude + longitude for the centre of your map (try left-clicking on Google Maps[14] for instant lat/lon to 6 decimal places). This allows it to integrate with other maps on We Make Earth[15] (optional).

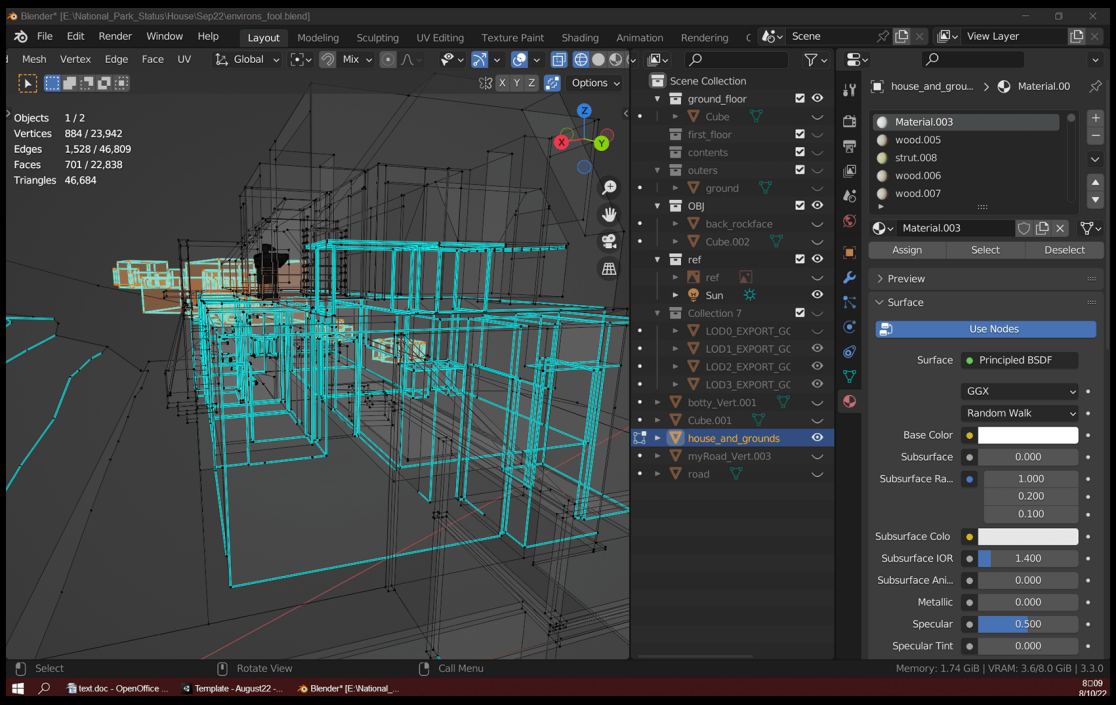
Part 2 : Make a first map

How does the headset (hardware) know about the world we want it to show? We tell it, by making a computer program (software). An immersive map is the combination of hardware + software.

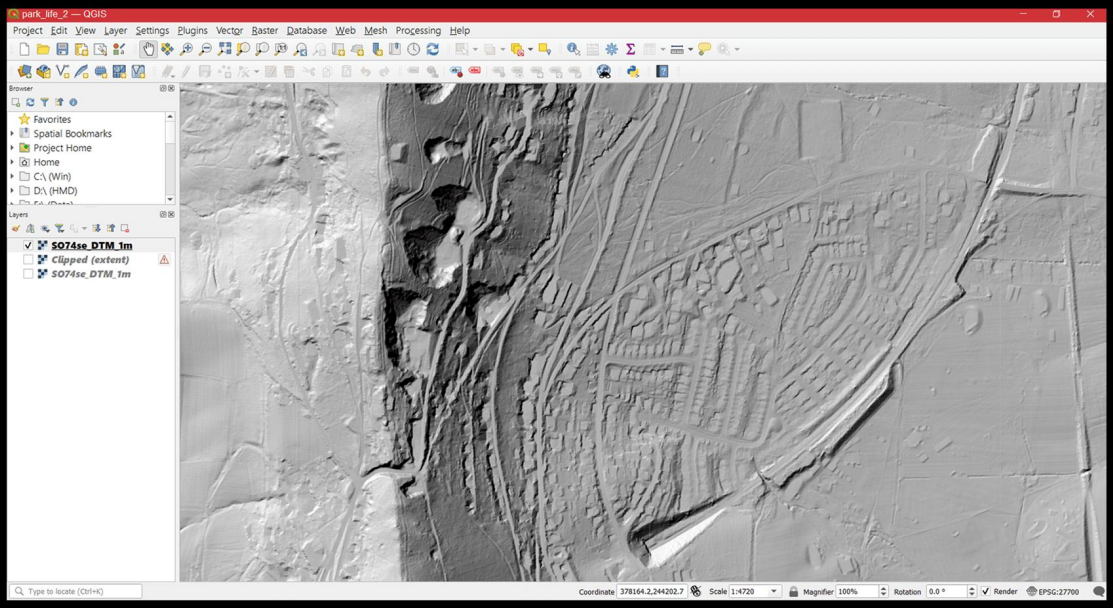
[13] A full Specifications document for We Make Malvern is detailed at the end of Appendix 1.
[14] google.com/maps
[15] wemakeworlds.com/earth



The Malvern demo was built inside the industry-leading Unity development environment. No license is required for projects with revenue or funding less than $100K in the last 12 months.

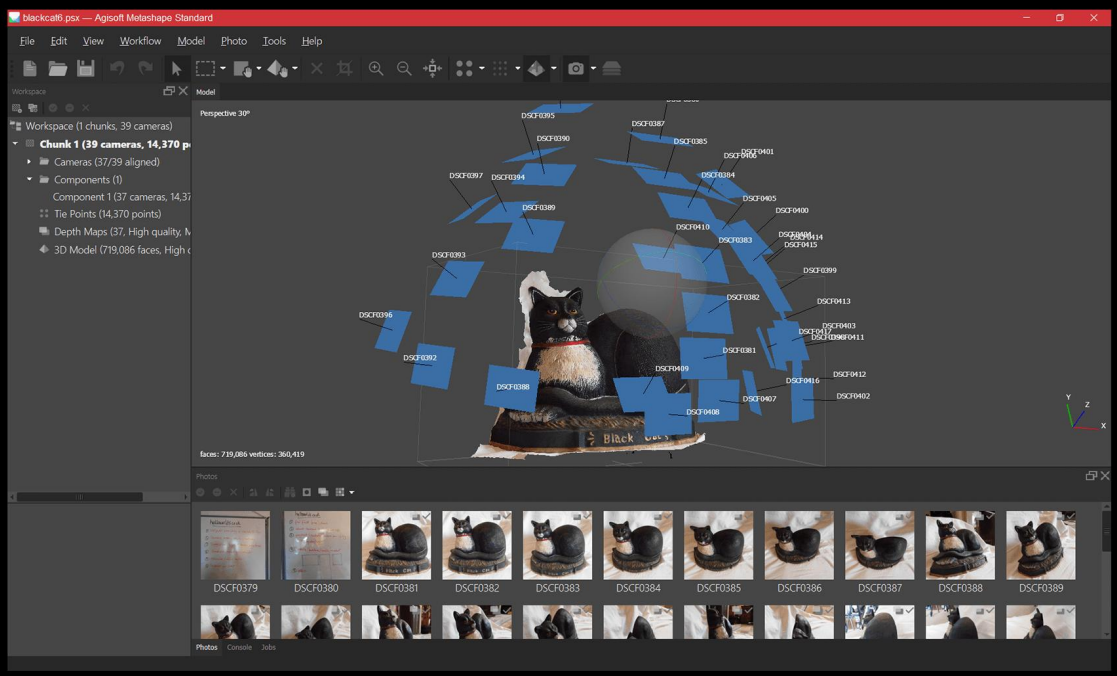


Assets and models were created using the open-source Blender 3D creation suite.



QGIS 3.28 Firenze

Data from a variety of sources is manipulated in QGIS, an open-source geographic information system, to form the basis of 3D landscape and environment models.

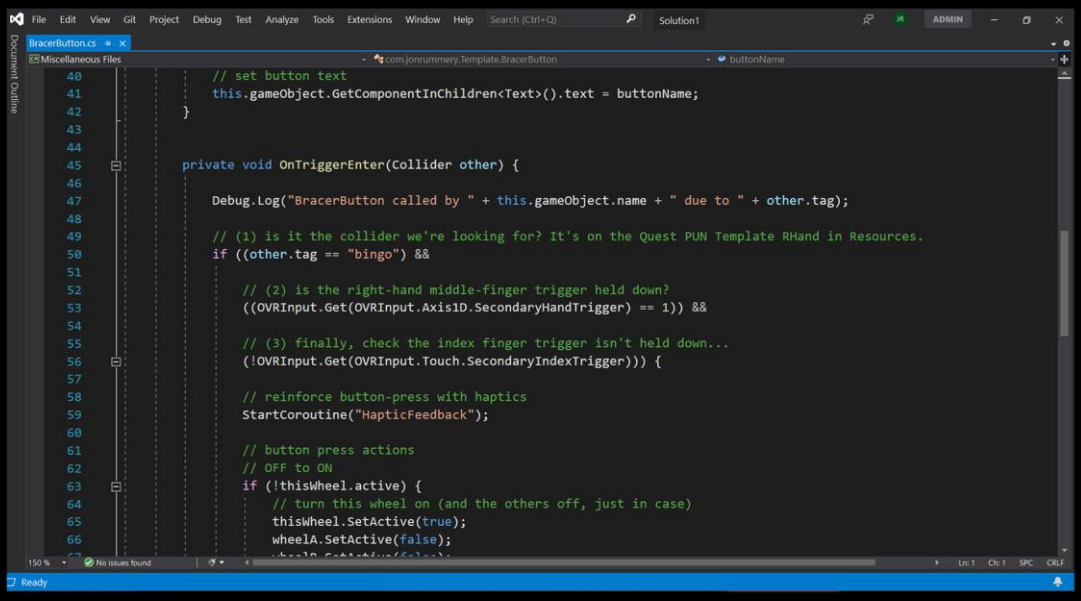


Scaniverse

Metashape

Environments, buildings and contents are recreated using data from photogrammetry, photography and manual measurement.

A LiDAR scanner and DSLR camera work with the Scaniverse app (which recently unlocked Premium features) and the Metashape software package.







Features and functionality are brought to life by scripting in the C# programming language using the VisualStudio integrated development environment.







The entire pipleline leads to your application being findable and downloadable from the Meta Store in seconds.

PART 2A : THE DEVELOPMENT ENVIRONMENT

It’s time to set up the hardware + software of your development environment. No prior experience is necessary other than basic computer skills. There’s a steep learning curve for complete beginners but progress is rooted in

delivering practical results that are measurable + fun to achieve. Remember, the internet is a great resource for tutorials + questions. You can always ask for help.

The hardware doesn't care that we're making a map, or whether it's good or bad. It just does what it's told, giving no quarter for our mistakes. The software, however, is completely under our control. Only we know what we want to achieve. Only we know when it is ready.

The GIGO (Garbage In, Garbage Out) adage stands, more so as the effort required to use an immersive map is greater than clicking a mouse - to travel by immersion only to find it's a glitchy mess will provoke more ire than a website displaying incorrectly.

(info box 1) Minimum hardware specifications.

PC : Any computer that can run a 'AAA' title.

Monitor : 27" (dual-monitors recommended)

CPU : Intel i5-4790 (or equivalent)

GPU : Nvidia 1070 (or equivalent)

Memory : 8GB

Disc : 120GB SSD (Solid State Drive; massive increase in performance over old mechanical drives)

OS : Windows 11 (Meta do not provide drivers for any other OS)

Mouse : 3-button with scroll-wheel

Keyboard : with numeric keypad

Headset : Meta Quest3 (Quest3, Quest3s, QuestPro and future devices). Quest3 recommended due to it's superior optics (30% more pixels + wider field-of-view than the cheaper Quest 3s). The Meta Touch controllers (included) run on 1xAA battery each (rechargeable batteries work).

(i) Set up the Quest headset

Out-of-the-box set-up + on-boarding takes 10 minutes. Just turn it on, put it on + follow the instructions.

You'll need :



(ii) Headset Developer Mode

In a web-browser, go to dashboard.oculus.com and sign in with your Meta account to create a new organization. This gives you access to the Meta Horizon Developer Dashboard. Create a developer team + associate your account

with the organization. Verify the account with 2-factor authentication, then activate Developer Mode in the Meta Horizon OS mobile app (Devices -> Quest3 -> Headset settings -> Developer mode -> On).

Ensure your Meta Quest headset can pair with the mobile app + is in Developer Mode (required to run our homebrew[16] apps).

(iii) Meta Horizon Link

Developing an application is an iterative process. We want to change something on the computer + instantly see it in the headset (witho having to create an output file every time).

Install the Meta Horizon Link app on your computer[17] + follow Meta's detailed instructions on successfully launching Quest Link from the headset.

Open the Link app on your PC and check Settings -> General -> OpenXR Runtime -> Meta Horizon Link.

(iv) Physical environment

The process of development will have you switch from keyboard + mouse to headset + controllers to pen + paper in quick succession, which can be ergonomically challenging. Make sure all six can peacefully co-exist on your work surface.

Consider a swivel-chair, raised to a height so that your forearm, wrist and hands are not held in crooked positions when typing + clicking. Aim to be looking straight ahead at your monitor(s). Create a safe zone for drinks and a stand for reference documents.

PART 2B : MAKE 3D MODELS WITH BLENDER

(i) Open the template models.
(ii) Recreate the template models.
(iii) Landscape.
(iv) Blosm.
(v) Export.

Blender turns raw data into 3D models. It will export every single object + environment in your map to individually named files for import into Unity (which builds it into the application that runs on the headset).

The program makes extensive use of keyboard + mouse shortcuts in it's quest to understand what you want. The function of many keys will become second nature in no time, but for the complete beginner it's necessary to spend some time learning the basics. YouTube videos are invaluable (try making a donut with Blender Guru[18]). Feel free to spend serious time playing before attempting anything ambitious.

The goal is to create + export 2 models, LOD0 + LOD1, that represent your versions of the models found in the template map[19].

[16] Software written by hobbyists for proprietary hardware.
[17] meta.com/en-gb/help/quest
[18] youtube.com/watch?v=z-Xl9tGqH14
[19] tinyurl.com/2s3fw37p

(info box 1) Triangles : vertices, edges + faces.

Digital models, to the computer, are all about triangles. The Quest3+ headset can theoretically shift 1.8M triangles @ 120Hz, but to ensure a smooth frame-rate consider capping at 1M @ 90Hz. The template map has 42 triangles, We Make Malvern has ~800,000.

1 vertex on it's own has no edge or face.

2 vertices can be joined to make 1 edge with 2 vertices + no face.

3 vertices can be joined to make 1 triangle with 3 vertices, 3 edges + 1 face
(this face, or side, is known as the 'normal'. NB The other side doesn't exist!).

The *topology* of a model refers to how surfaces are constructed from vertices, edges + faces. We use triangles because they're *planar* – all 3 vertices always lie in a single plane.

(info box 2) OBJECT mode + EDIT mode.

OBJECT mode is for moving, rotating or scaling models.

EDIT mode lets you play with the vertices, edges + faces of a particular model.

The TAB key switches between OBJECT mode and EDIT mode. Mouse-clicks, menus + key-presses can behave differently in the two modes.

(info box 2.1) 1,2 + 3

In EDIT mode, quickly move between vertex, edge + face mode by pressing 1, 2 + 3 respectively.

(info box 3) There's 11 different workspaces to choose from in Blender, each for a different aspect of 3D modelling. We're only going to use the 'Layout' workspace, which is the one that should be selected by default. Ignorance here is something of a philosophy… 80% of the time we're only going to use 20% of what's on the screen. Train yourself to only see + understand the bits you need (+ try not to get overwhelmed when you press the wrong button). Save often.

(info box 4) Changing the view with the mouse.

Scroll-wheel : zoom in/out
Middle-mouse button : orbit
Shift + middle-mouse button : pan
Ctrl+ middle-mouse button : dolly

(info box 5) We will use 3 active windows for the current workspace :


3D Viewport


Outliner


Properties

Become familiar with how Blender allows you to rapidly resize + repurpose windows.

(info box 6) 3D Cursor + origin

3D Cursor : a reference point in 3D space.

CTRL+middle-mouse button to set cursor.

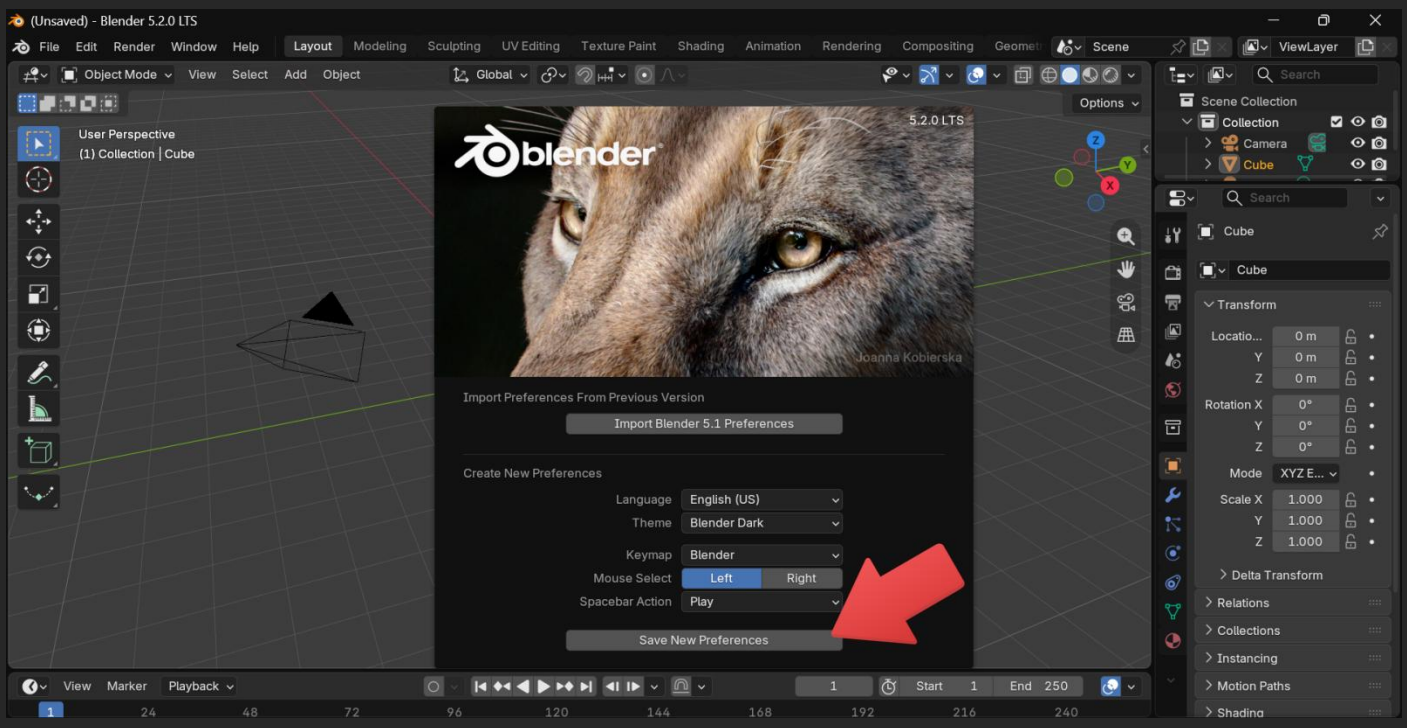


Origin : the location in 3D space of an object. Denoted by a dot with different color depending on state.

(info box 7) Zoom in to selected object with NumLock+.

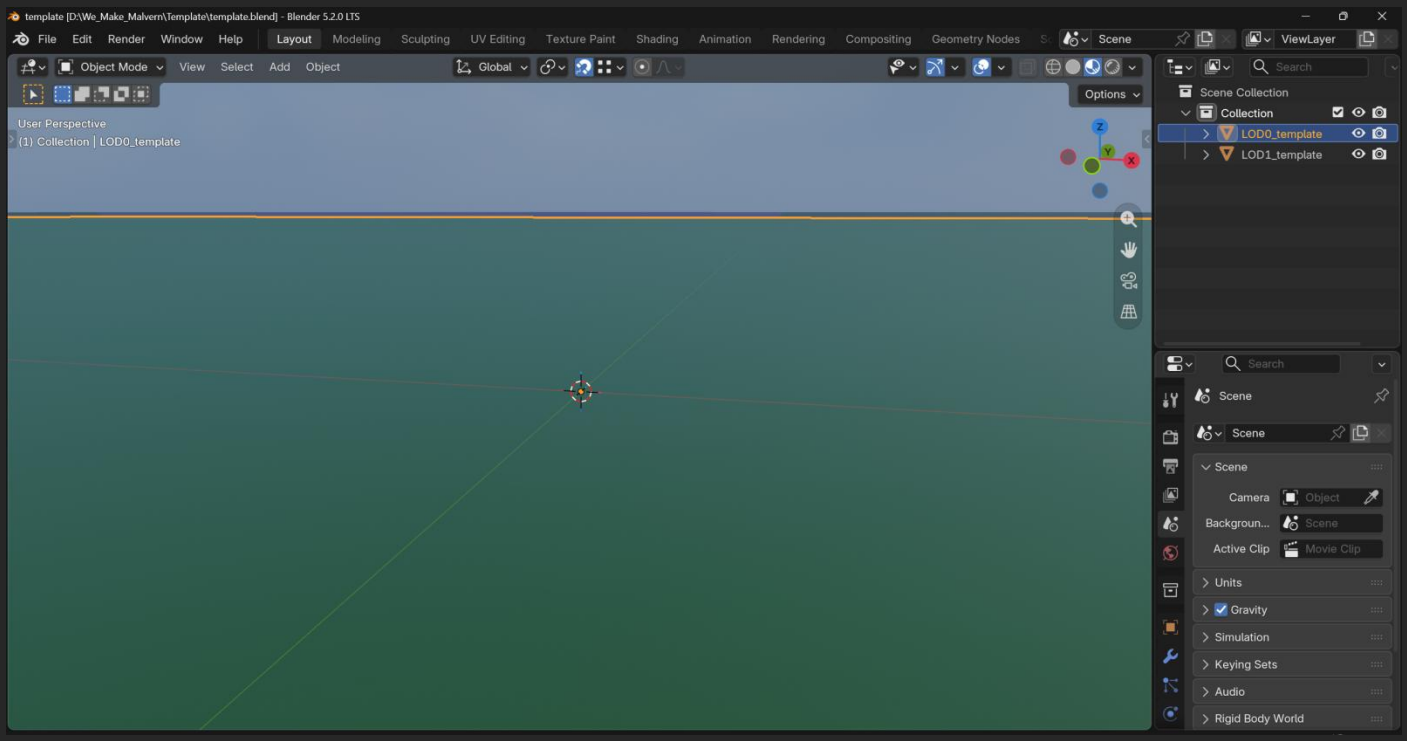
(info box 8) .obj files : simple data containers for 3D models using an ASCII[20] format description of geometry + materials.

(i) Open the template models.
Install Blender 5.2 LTS[21] + click 'Save New Preferences' when ready.

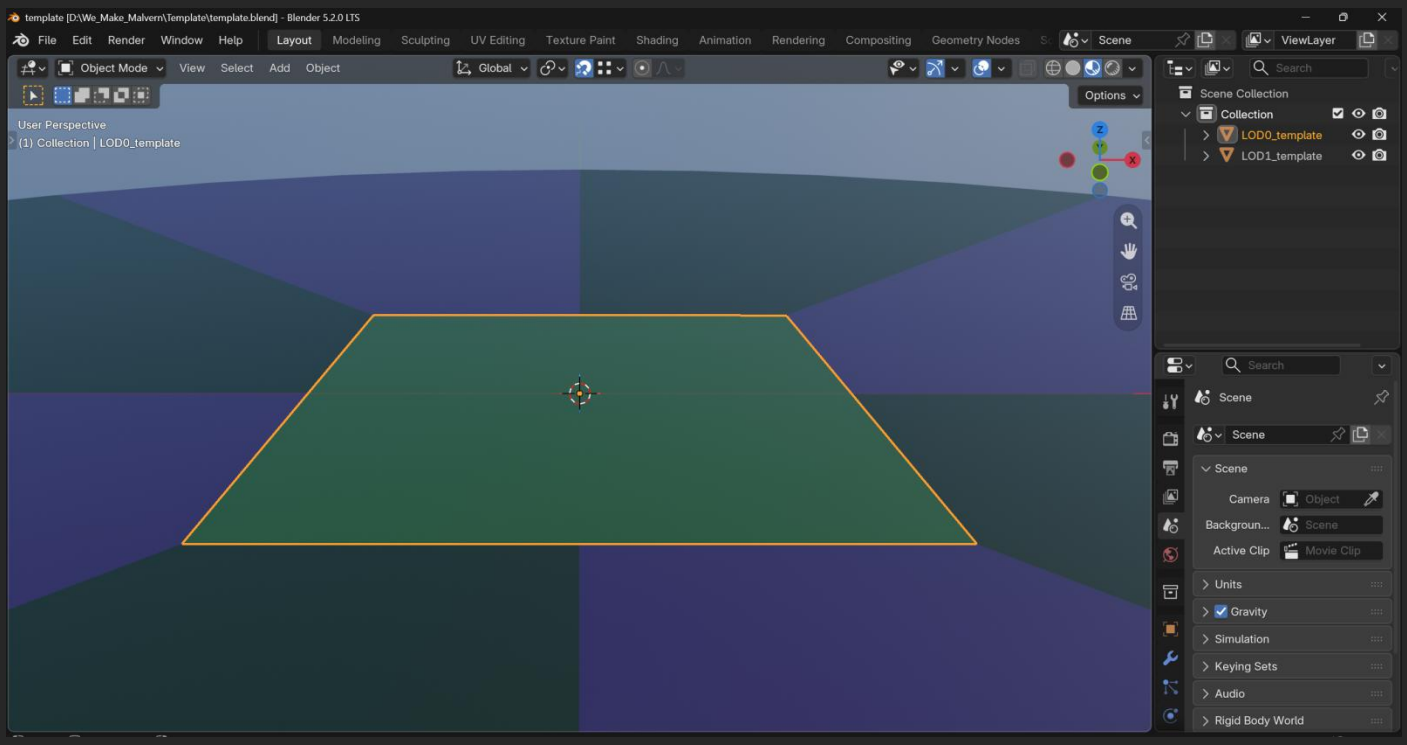


Download the template.blend file[22] + open it.

[20] A character encoding format, human-readable with a text editor.
[21] blender.org
[22] tinyurl.com/2s3fw37p

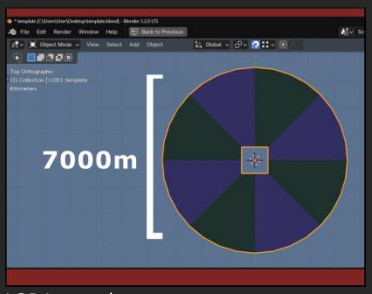


Orbit the view so you can see both LOD's properly.

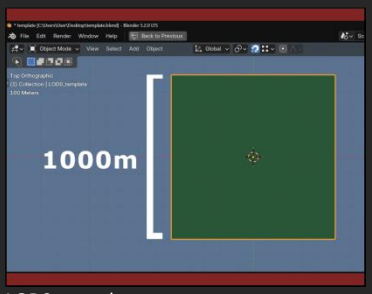


(ii) Recreate the template models.

Before making your own detailed models, fully understand how the template LOD's were made by recreating them from scratch.

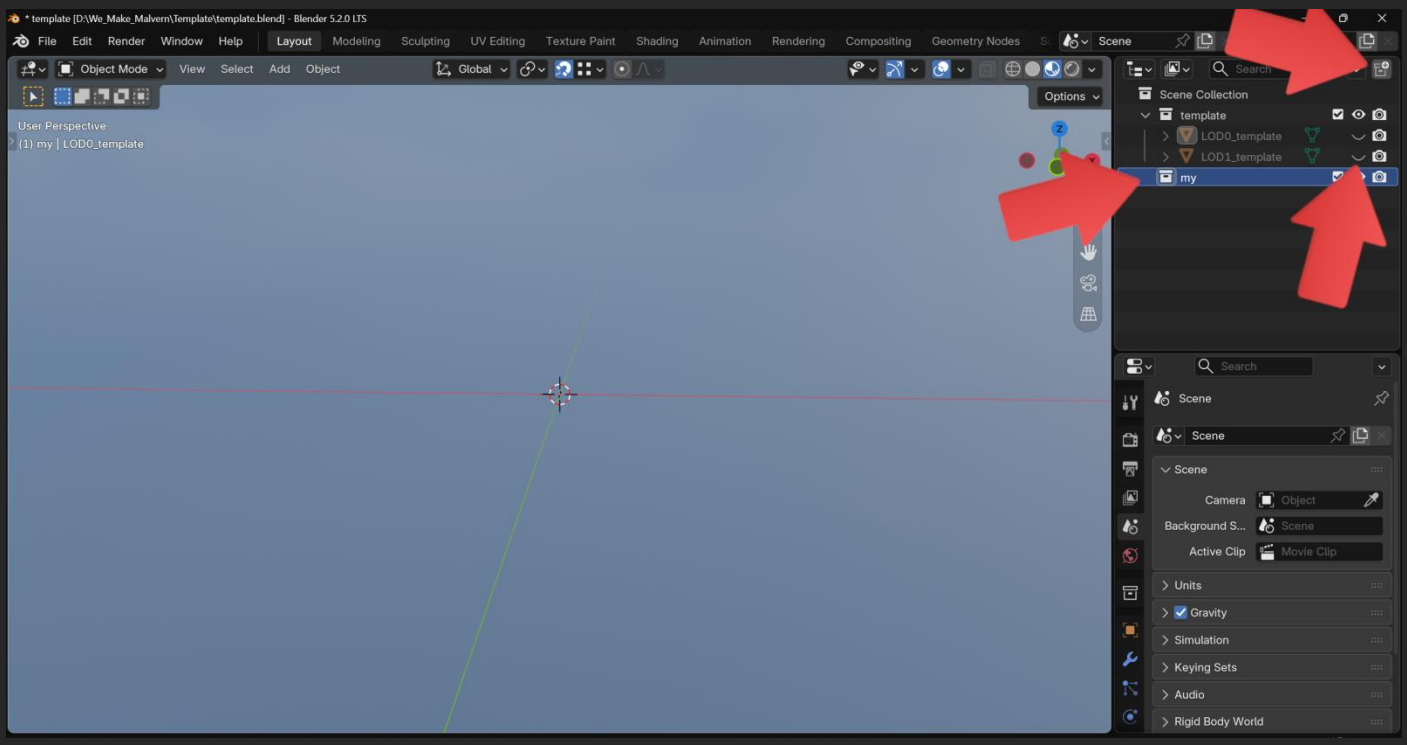


LOD1_template
Low-detail landscape
(40 vertices, 48 edges, 8 faces, 40 triangles)

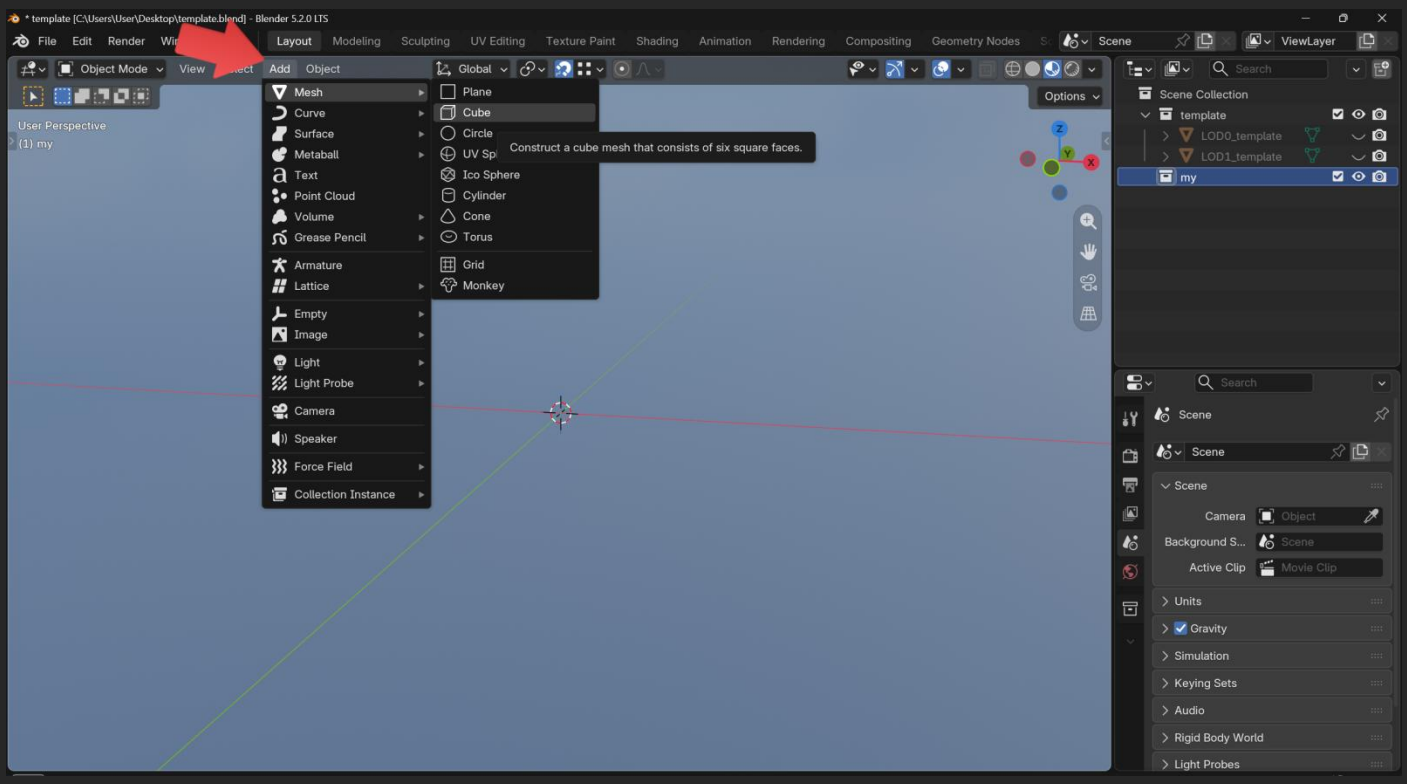


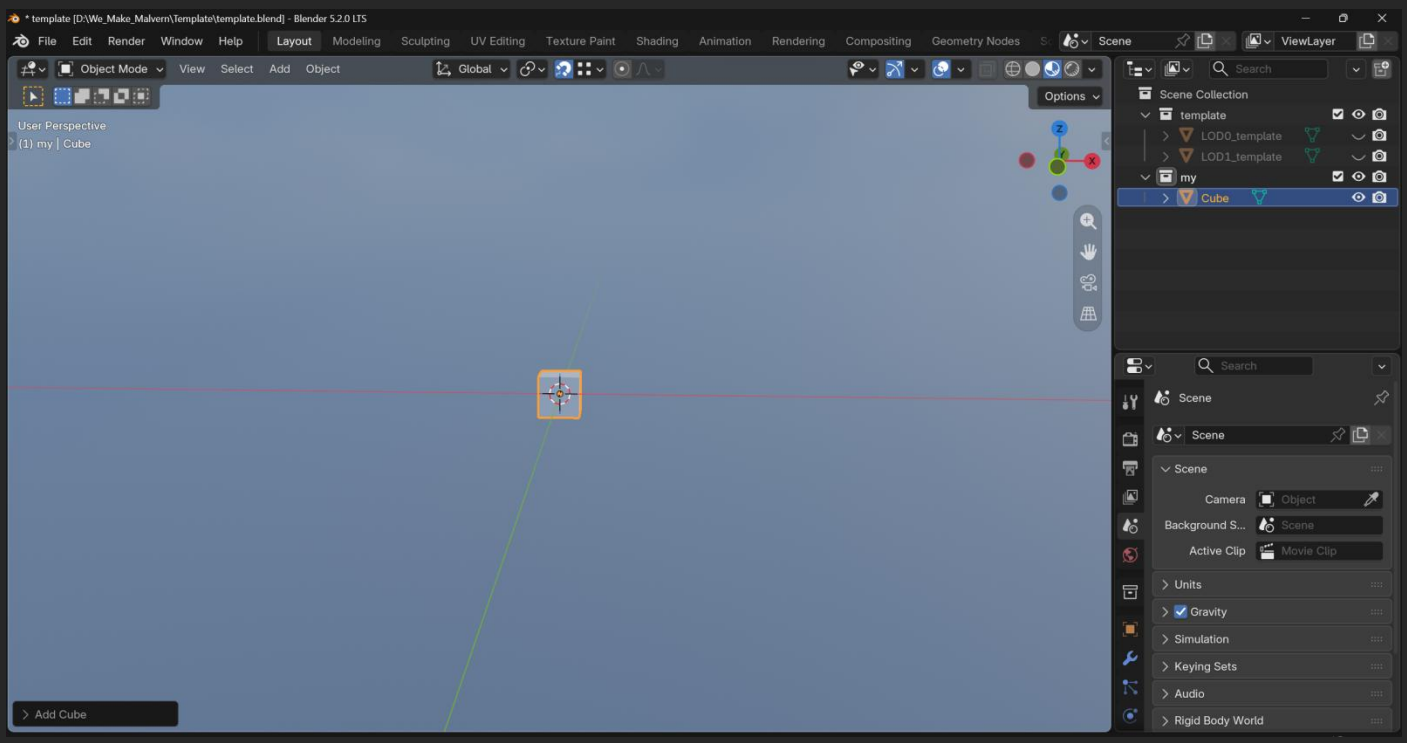
LOD0_template
High-detail environment
(4 vertices, 4 edges, 1 face, 2 triangles)

Hide the template models in the Outliner by clicking on the ‘eye’ icon next to their names. Rename the Collection to template and create a new Collection (resize the Outliner window if you can’t see the icon). Rename this Collection to ‘my’ and select it.

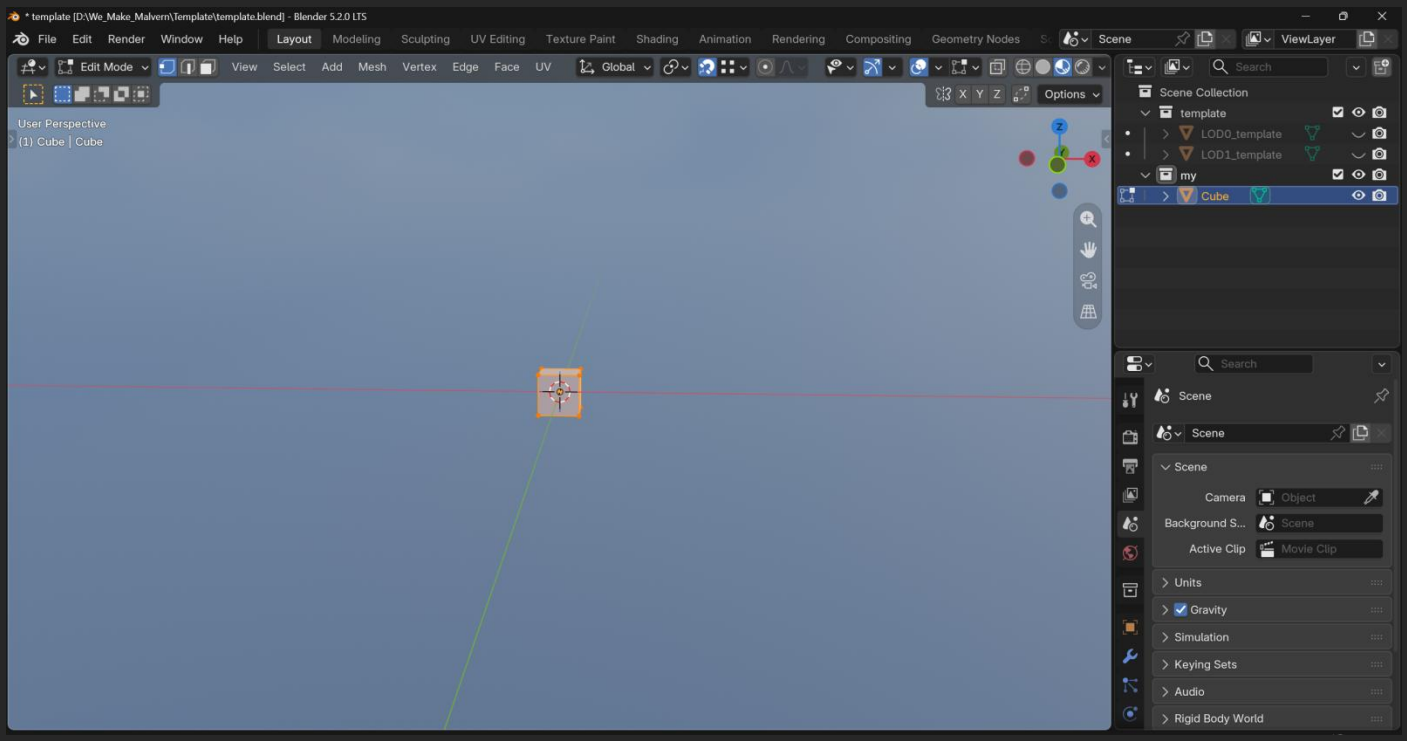


In OBJECT mode, make a vertex by first creating a cube with Add -> Cube. (NB There’s many ways to create a vertex… this is just one way). Challenge : Why are we in OBJECT mode + not EDIT mode? We’re creating (OBJECT mode), not manipulating (EDIT mode).

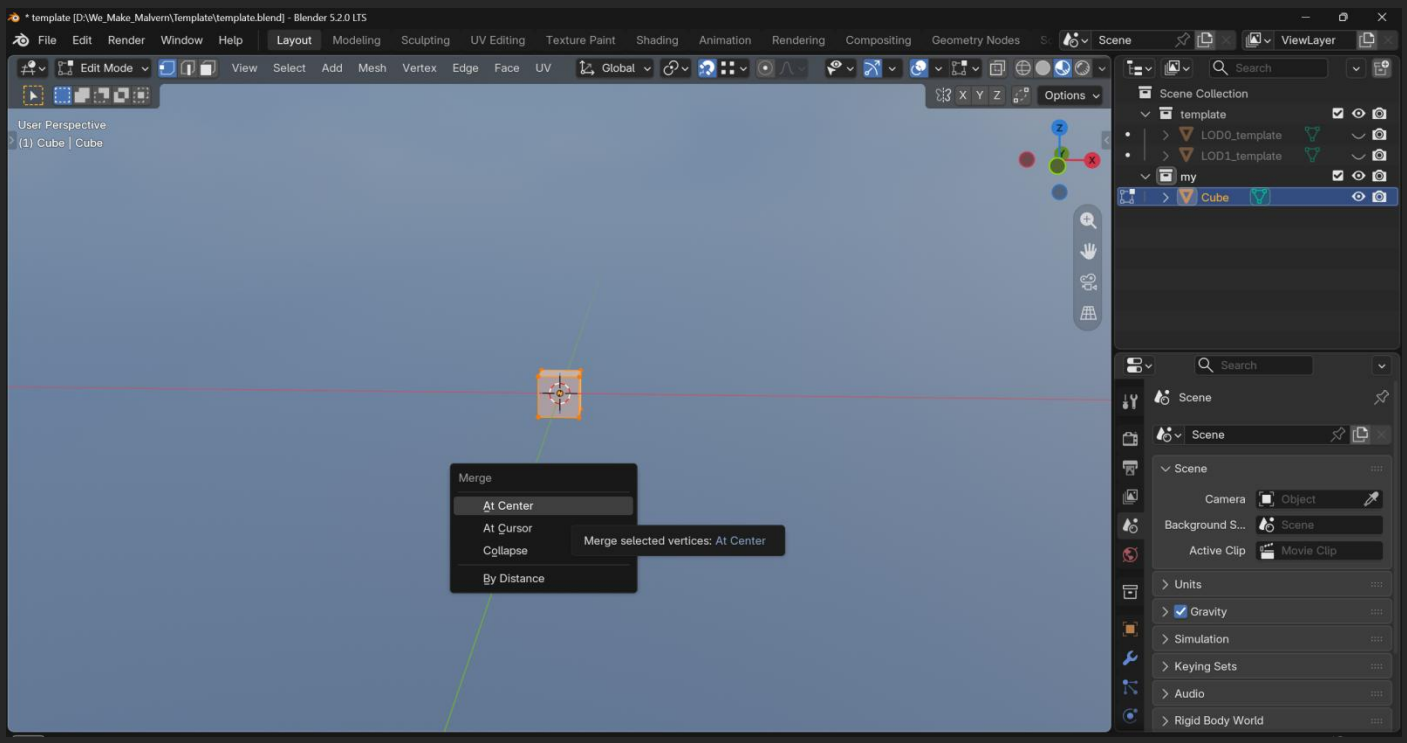




Press TAB to enter EDIT mode.



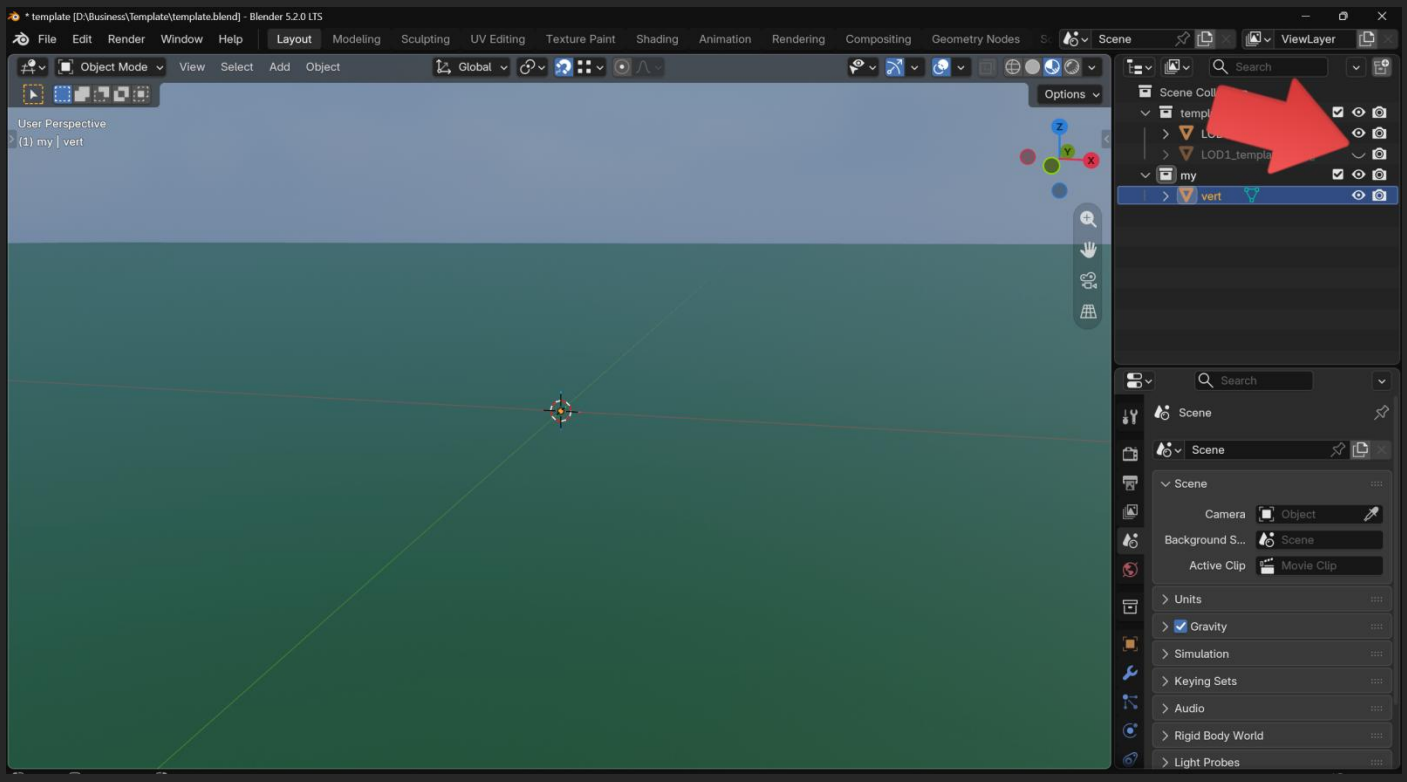
Press 1 to make sure you're in Vertex mode. (Challenge : Where on the screen does it tell you you're in Vertex mode?) Press A to ensure all the vertices are selected (you may have to zoom in/out to see the cube). Press M to open the Merge menu + select At Center.



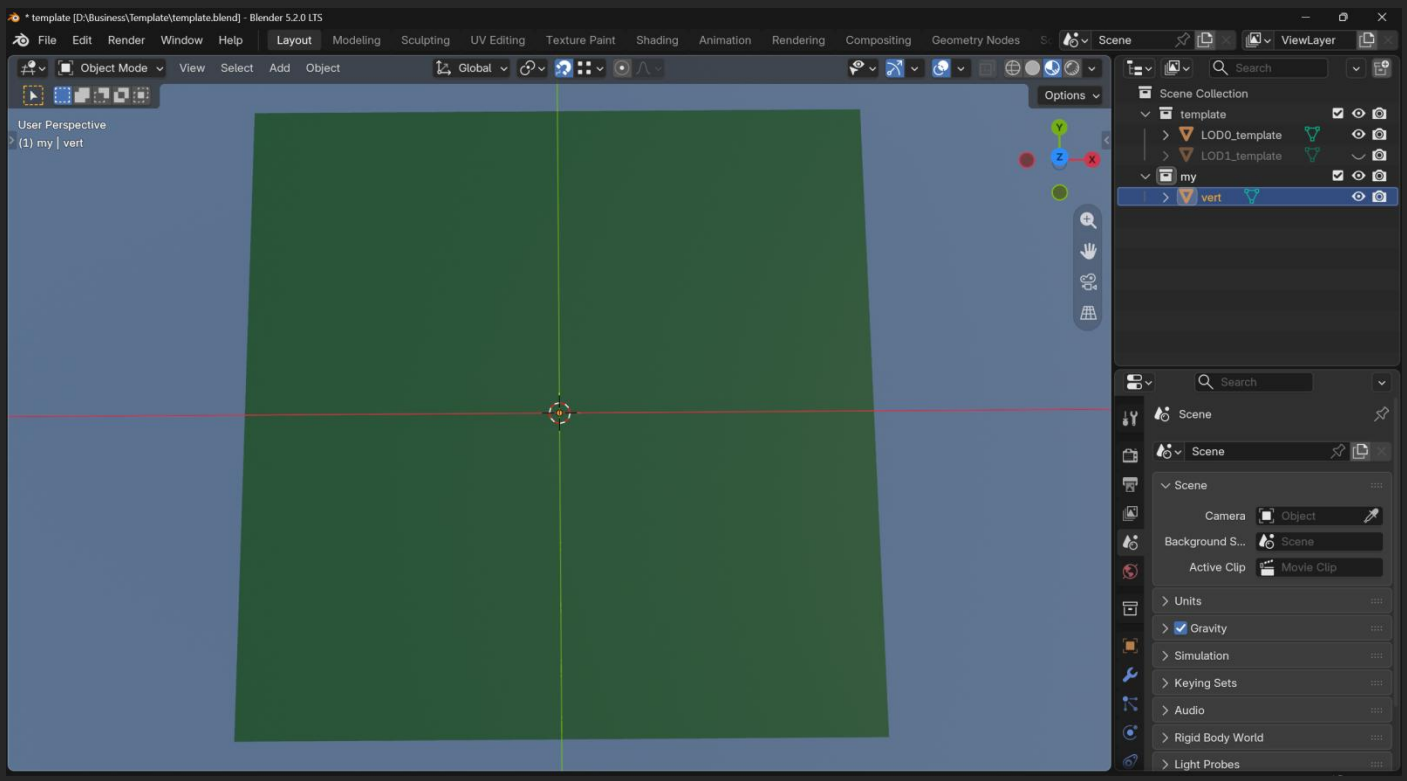
A single vertex! From this tiny point of nothing, our world will take shape. Rename it 'vert' and select it.

(Question) How many ways on the screen can you tell that this model is in EDIT mode? A: 3+.

Click on the 'eye' icon next to LOD0_template to make it visible again, for reference.

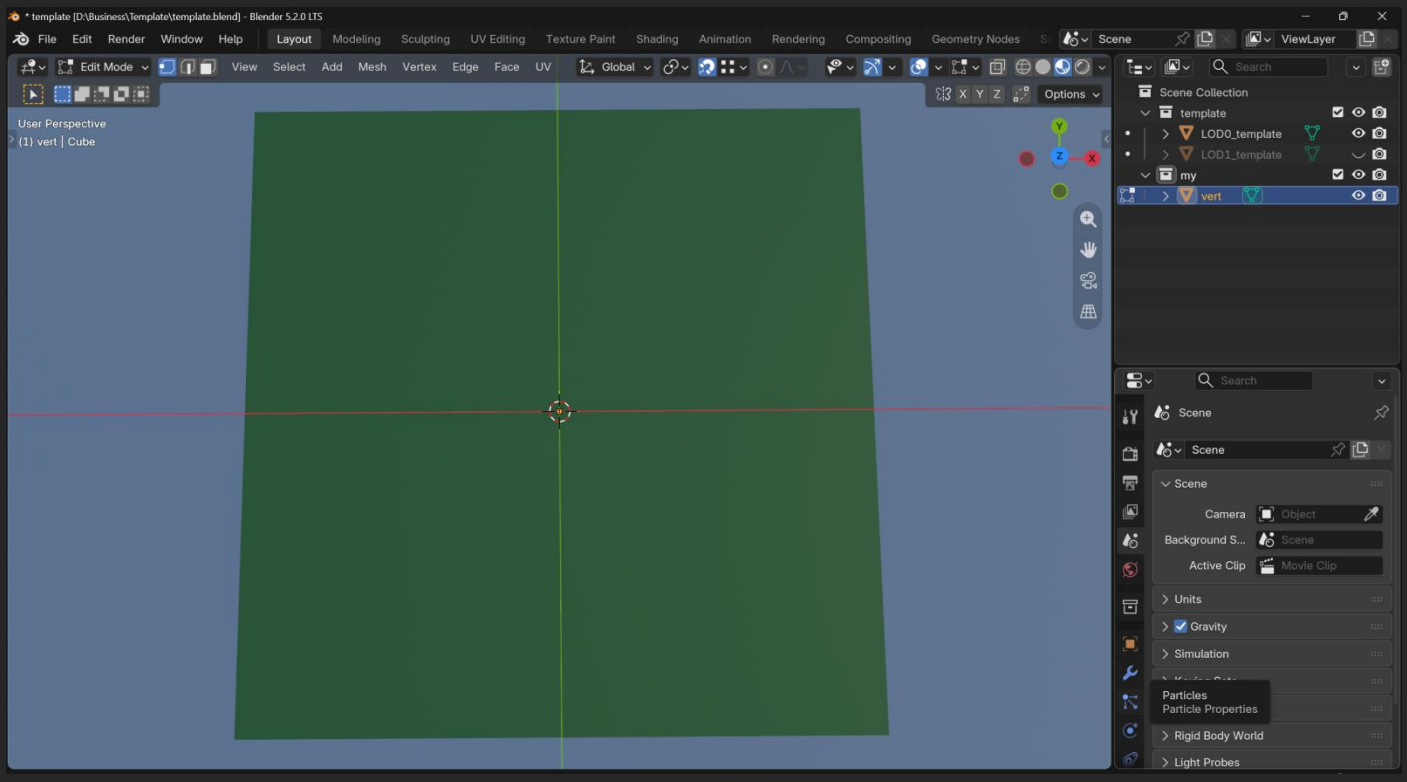


Orbit the view so that you can see the whole of LOD0_template. Notice how the red + green grid lines correspond to the X + Y axis of the gimbal (top-right of the 3D Viewport).



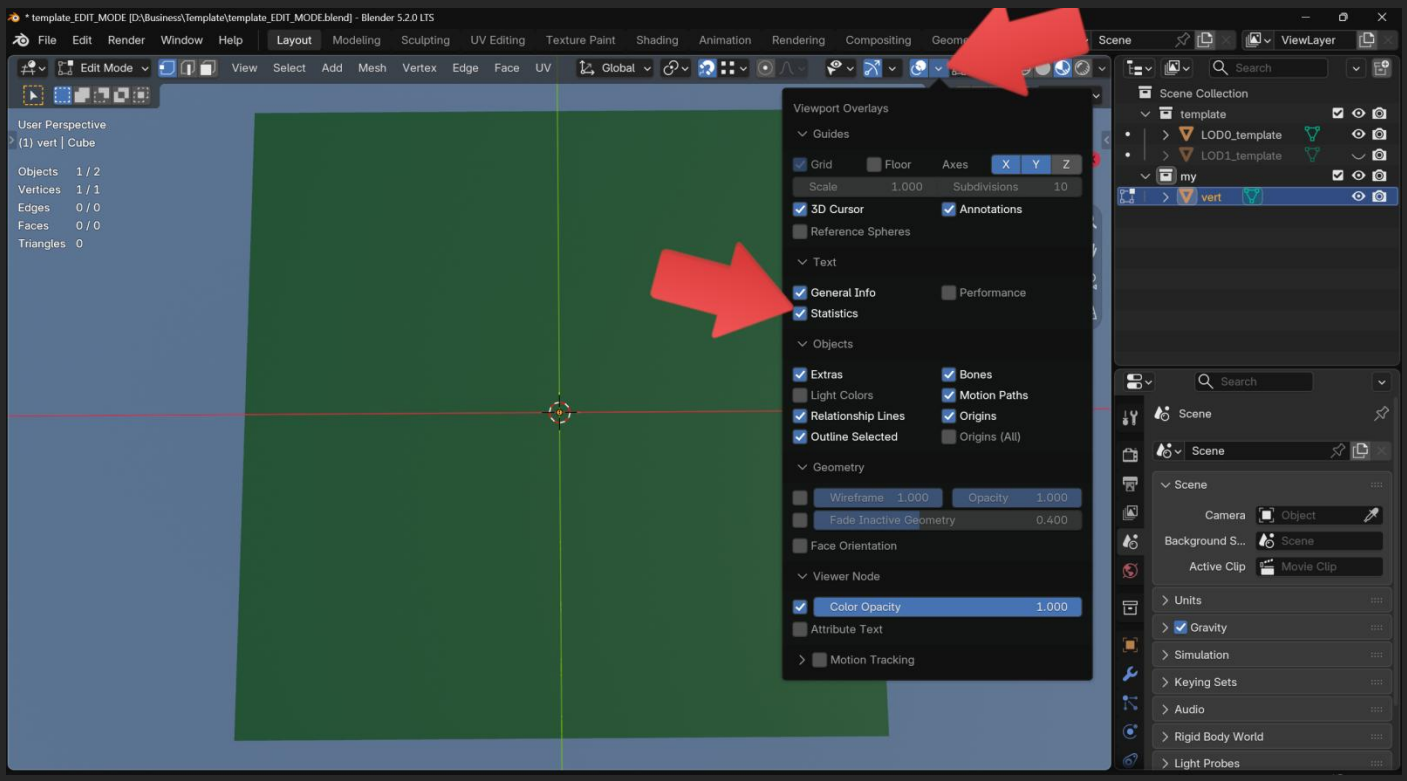
(NB Our vert is still selected in the Outliner. Blender hasn't selected the LOD0_template just because we've made it visible).

Press TAB to enter EDIT mode.



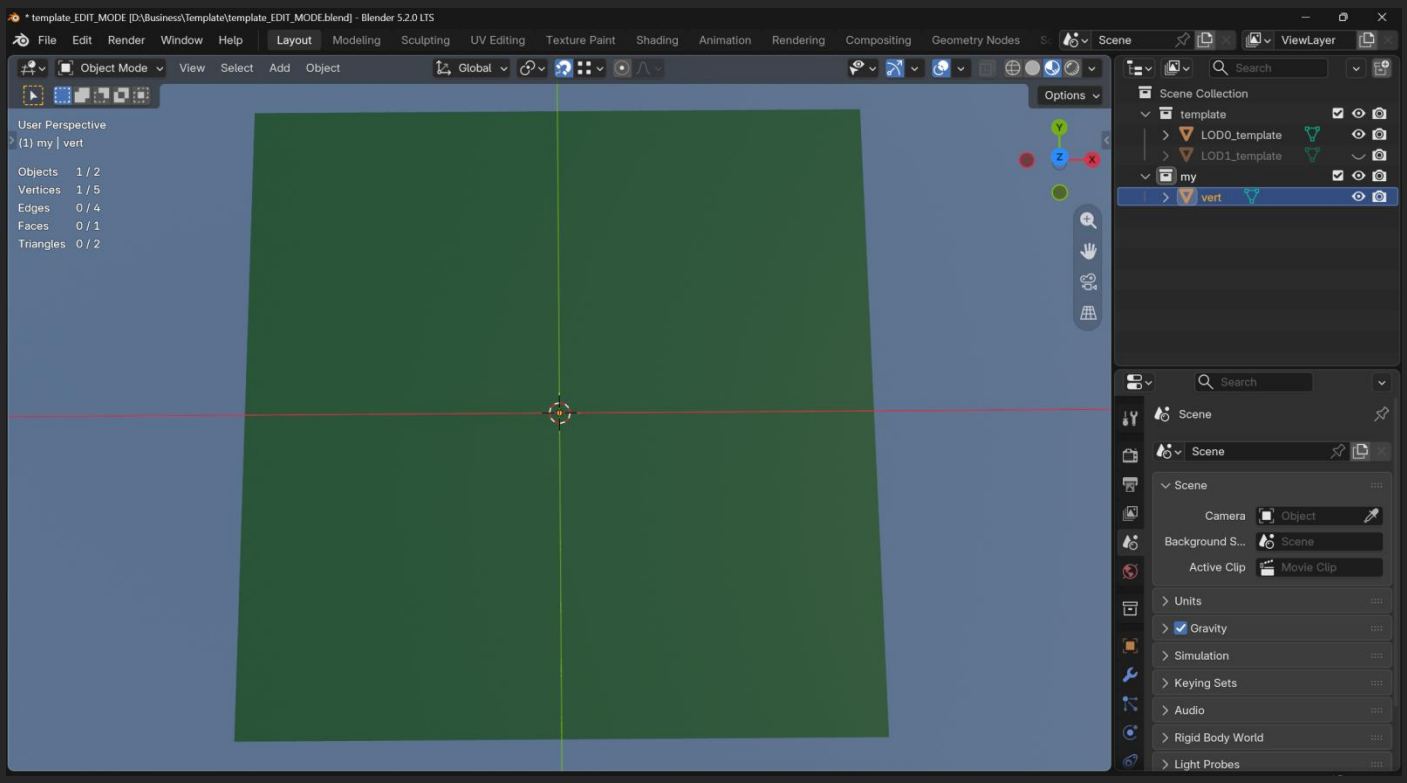
Our single vert is now in EDIT mode. Blender has changed the display to remove options that only exist in OBJECT mode + add options that only exist in EDIT mode.

Click on the Overlays menu + check the tickbox for Statistics to get an authoritative count of the vertices, edges, faces + triangles selected for manipulation, handily placed in the top left of the 3D Viewport.



In EDIT mode, statistics are delivered per selected object(s). We have 1 of 2 objects selected : vert selected, LOD0_template not selected, LOD1_template hidden (try unhiding/hiding LOD1_template to see the change to stats). There's one vertex selected out of a possible 1. No edges or faces exist.

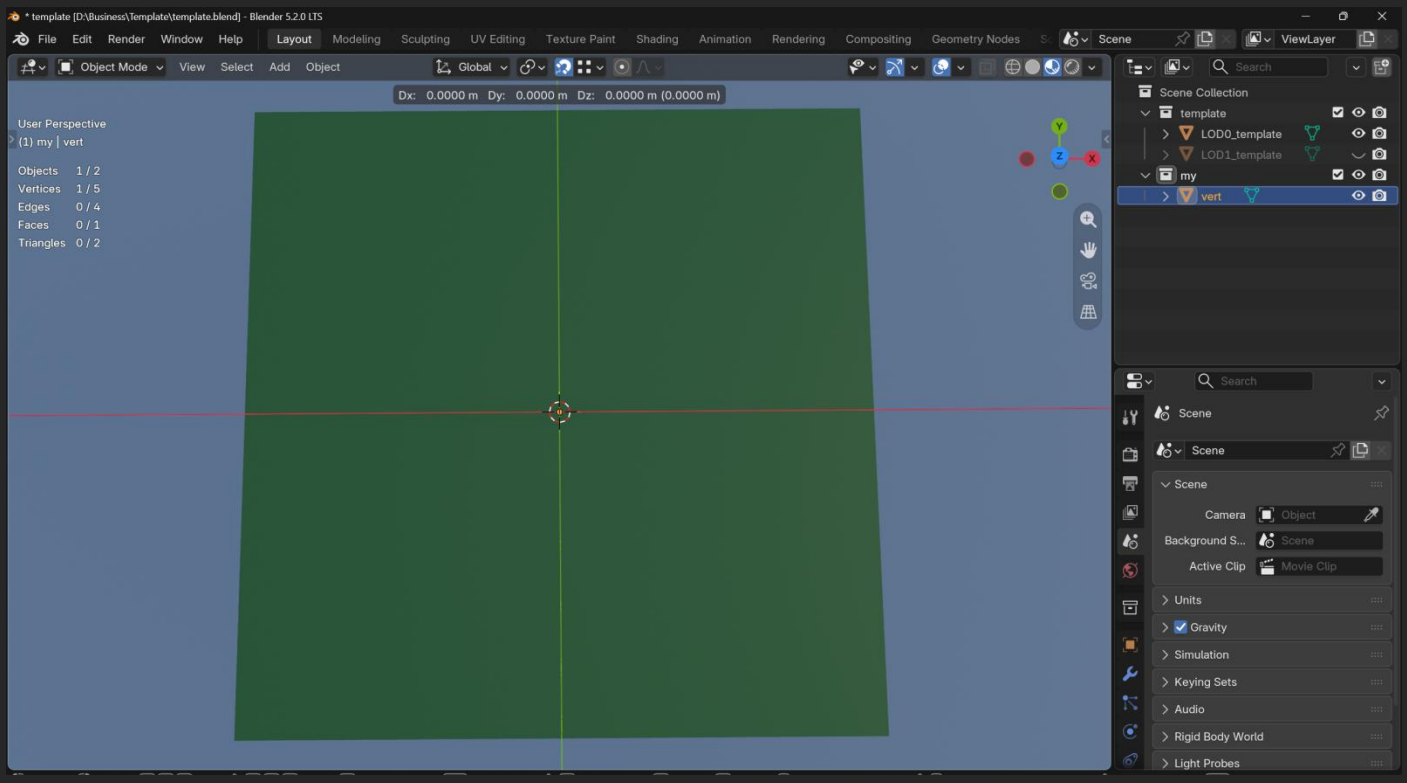
Press TAB to change back to OBJECT mode.



OBJECT mode gives stats for visible objects. We still have 1 of 2 objects selected, but we now have 1 of 5 vertices (1 single vertex + 4 unselected vertices which belong to the visible LOD0_template that didn't show up in EDIT mode).

Play with switching between OBJECT mode + EDIT mode. Hide + unhide objects in the Outliner. Become familiar with reading the stats.

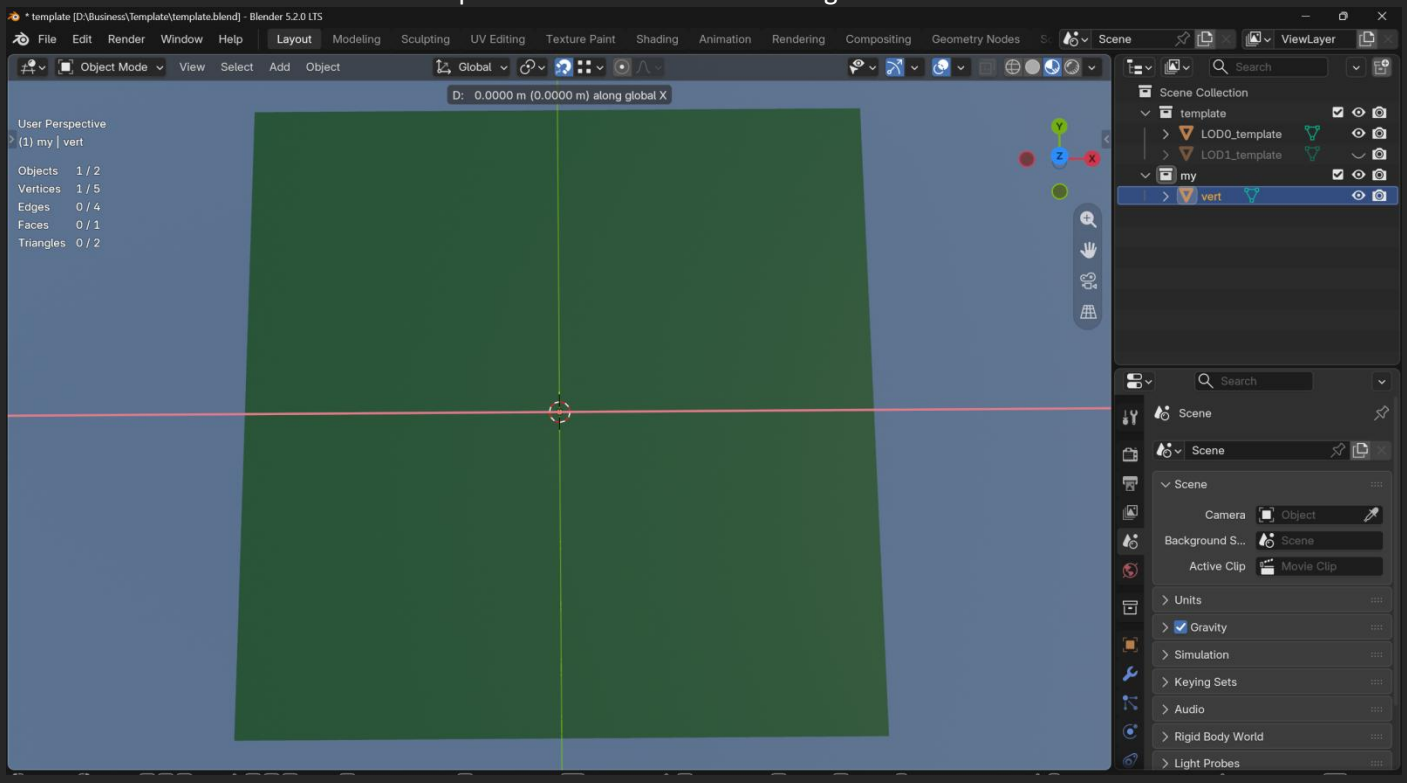
In OBJECT mode, ensure the vert is selected. Press G to start moving the vert.



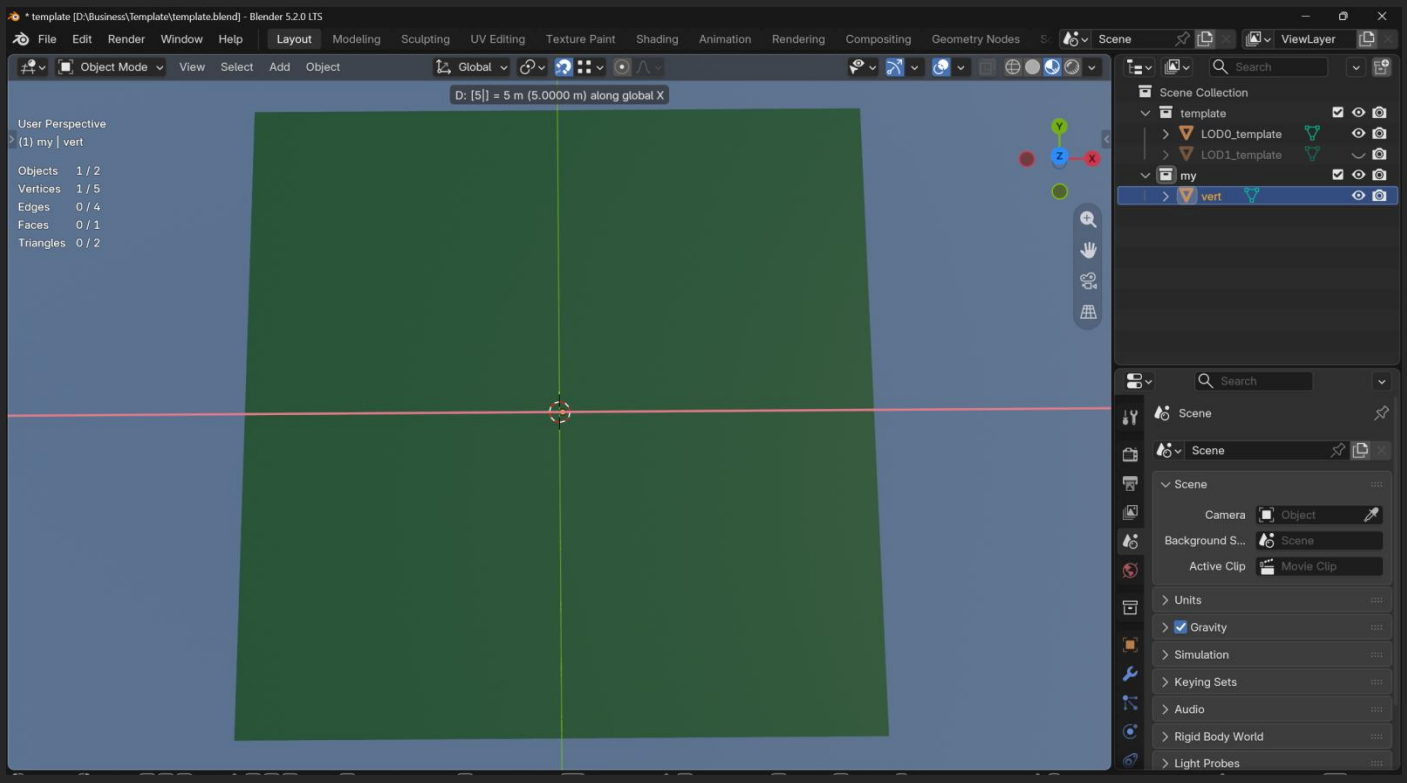
Notice how the top middle of the 3D Viewport is a lot of zero numbers. Blender is telling you that the next input from keyboard or mouse will be used to input distance (D) in the x,y+z dimensions. Remember, we're working in 3D space on a 2D screen. We could wiggle the mouse + hope for the best, but keyboard entry is 100% accurate.

LOD0_template is 1km^2. Our vert is slap in the middle. So we need to move it 500m in the +ve x-axis (from our view, that's to the right).

Press X to tell Blender that our next input is for the x-axis. The x-axis grid-line becomes thicker.

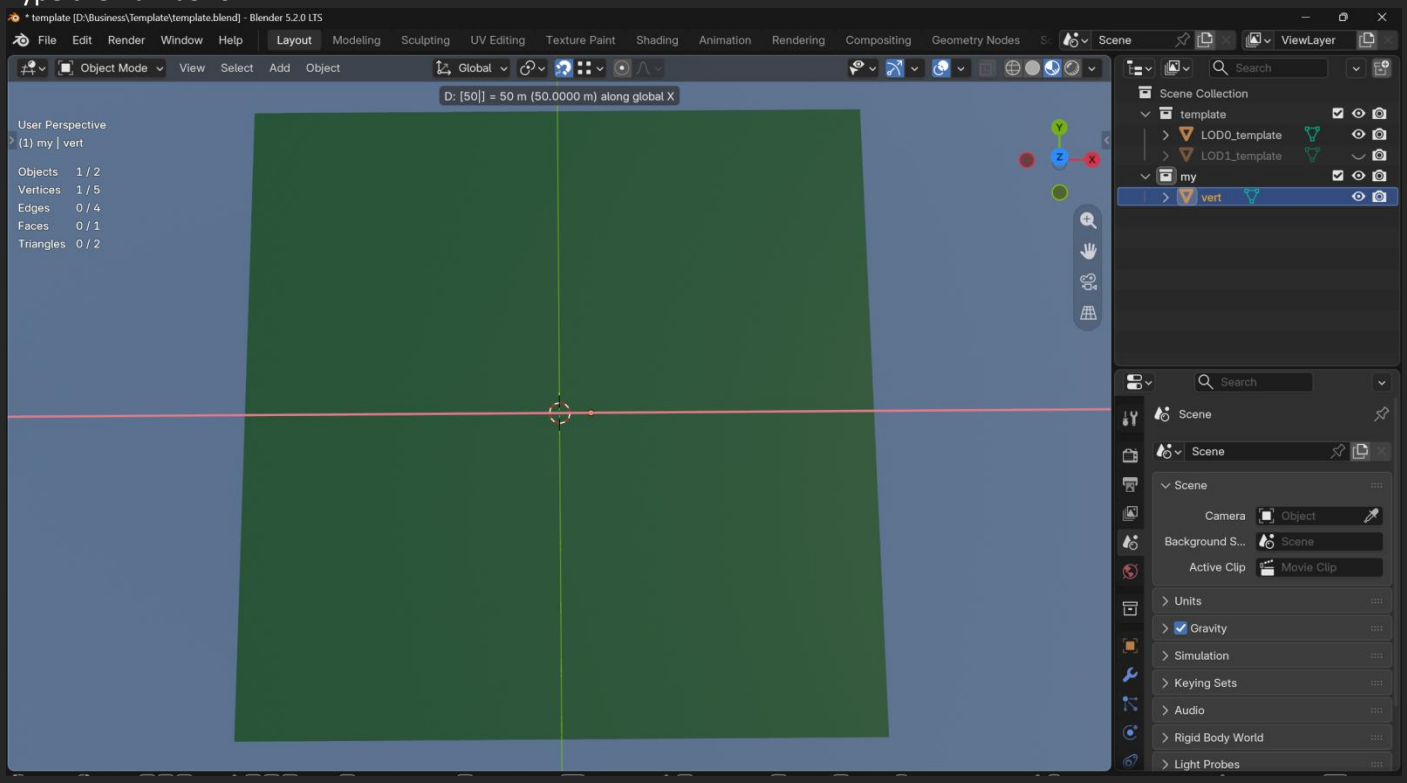


Type the number 5.



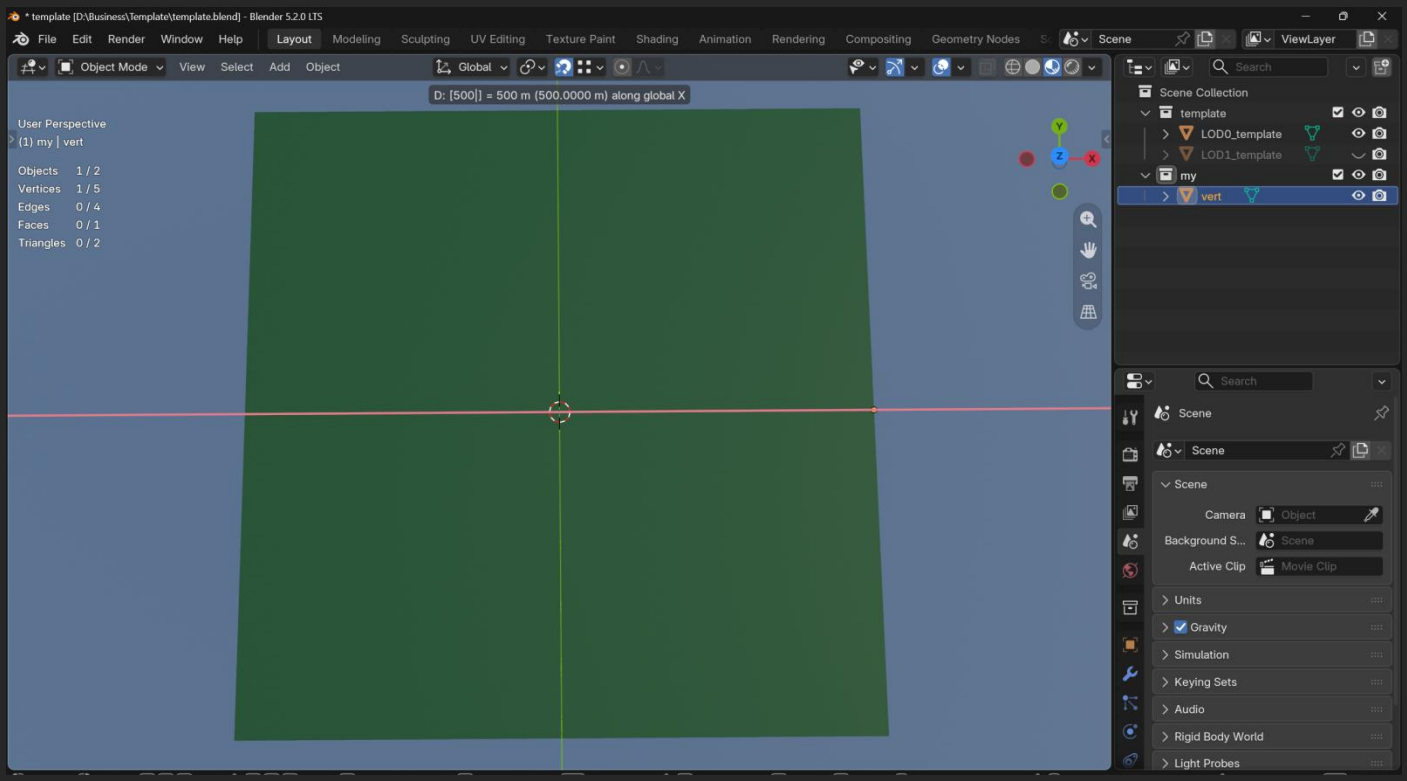
The vert has moved slightly along the x-axis. Your zoom-level might show this differently.

Type the number 0.

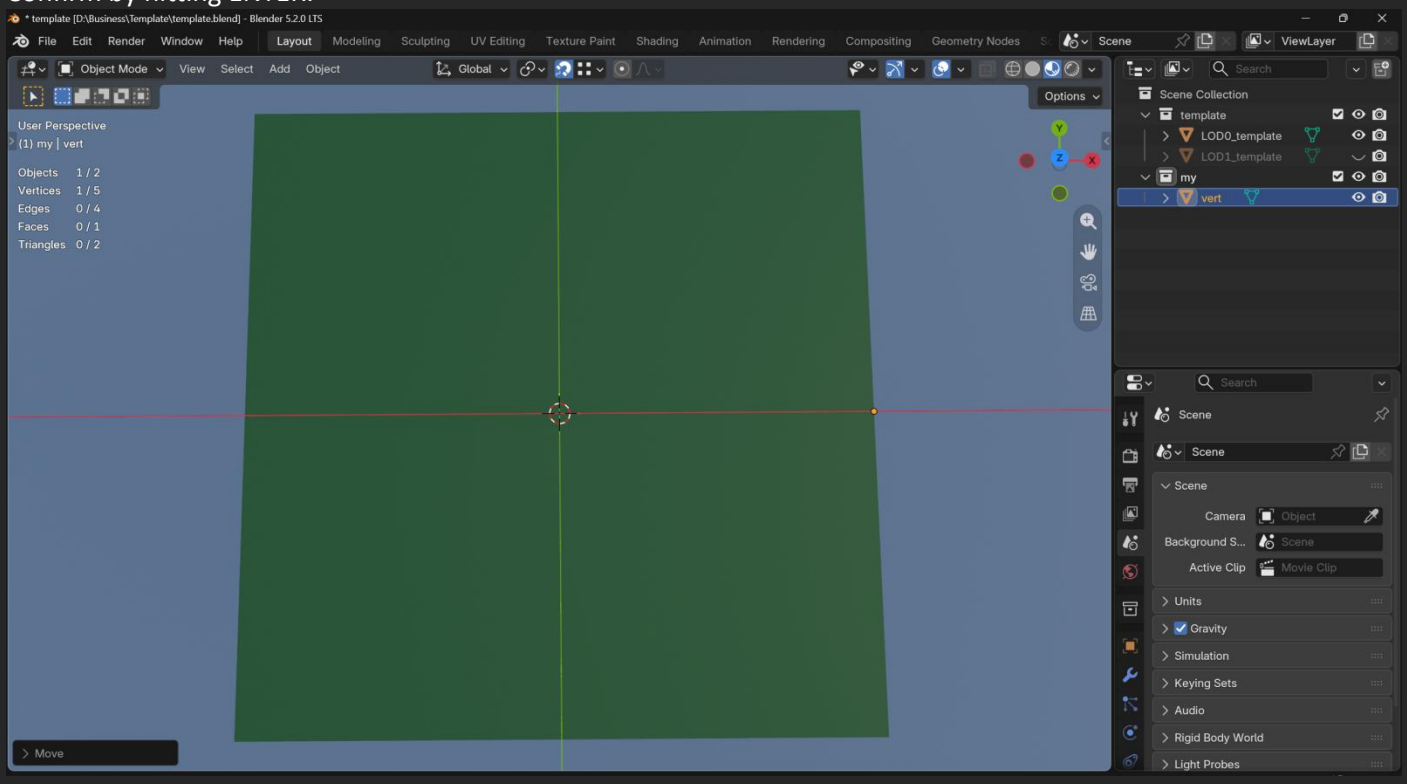


The vert jumps further along the x-axis (45m further across, to be precise). We haven't told Blender that number entry is finished, so the first digit (5) took the vert 5m from where it started + the second digit (0) placed it 50m from origin.

Type another 0 to make the vert move 500m along the x-axis from origin.

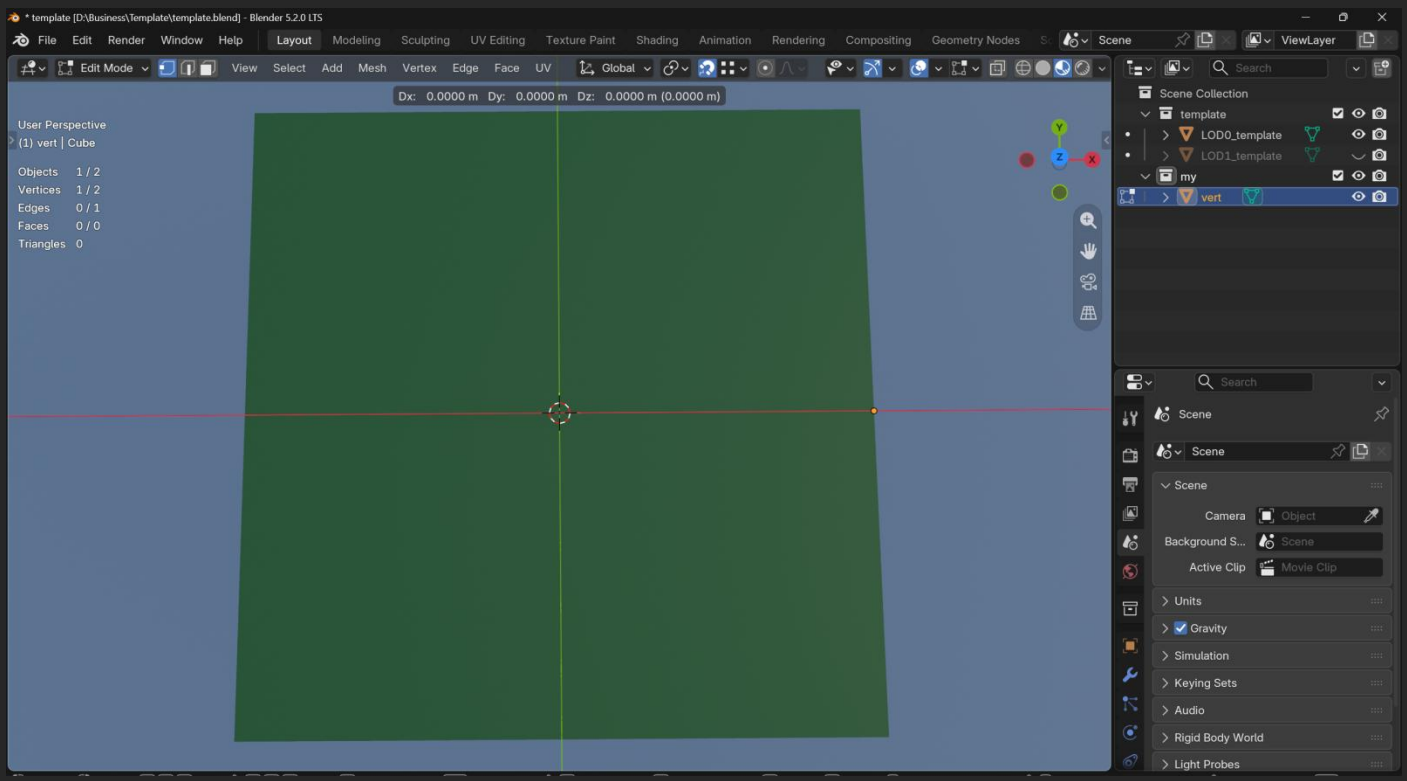


Confirm by hitting ENTER.



That puts the vert nicely in-line with the LOD0_template model.

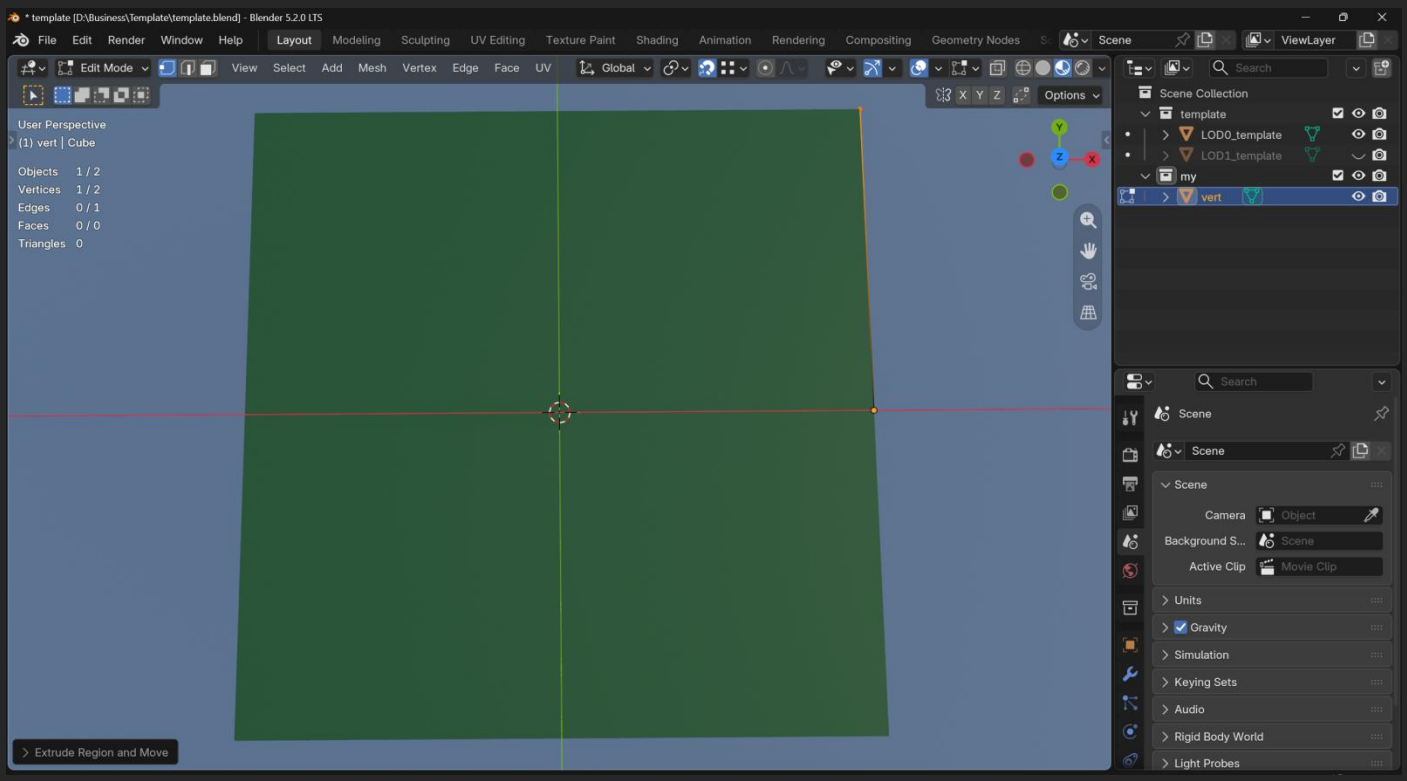
To create form, make sure the vert is selected, TAB into EDIT mode + extrude (push out) the vert by pressing E.



There's now 2 verts + an edge! Blender has made + selected a duplicate vert in order for an end-point to exist + an edge to connect it to the original. If you don't finish the extrusion operation, that second vert + edge still exist *on top of the original*, making it hard to detect. (NB CTRL+Z to UNDO).

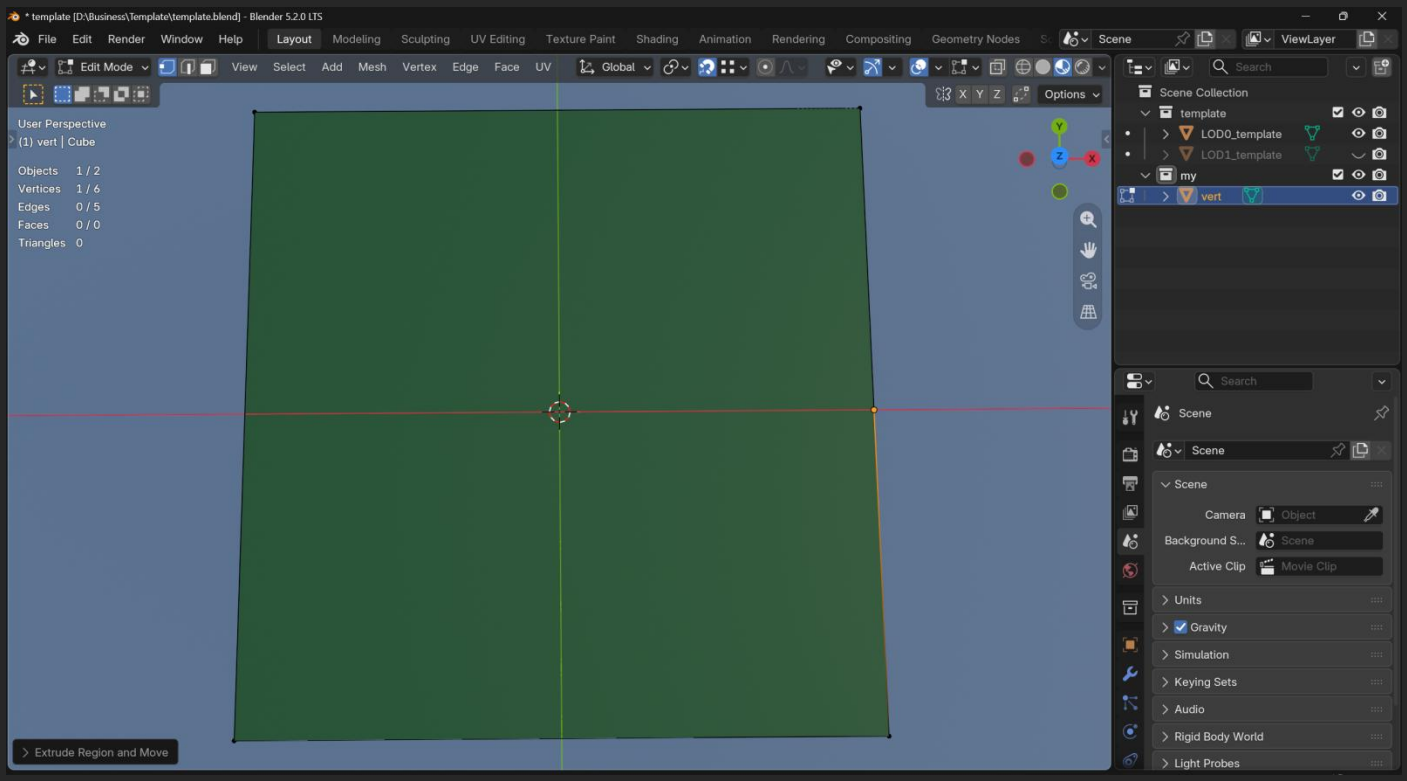
(Challenge) How many metres + on what axis lands the new vertex on the top-right corner of the LOD0_template? What keys need to be pressed to tell Blender *+ in what order*?
Notice that Blender is telling us that the next input is for the 3D co-ords of where we want to land the vertex.

Press Y to select the y-axis. Type 500 followed by ENTER to confirm.



(Challenge) Extrude the vert all the way round to complete the square, finishing up where you started.

E -> X -> -1000 -> ENTER -> E -> Y -> -1000 -> ENTER -> E -> X -> 1000 -> ENTER -> E -> Y -> 500 -> ENTER

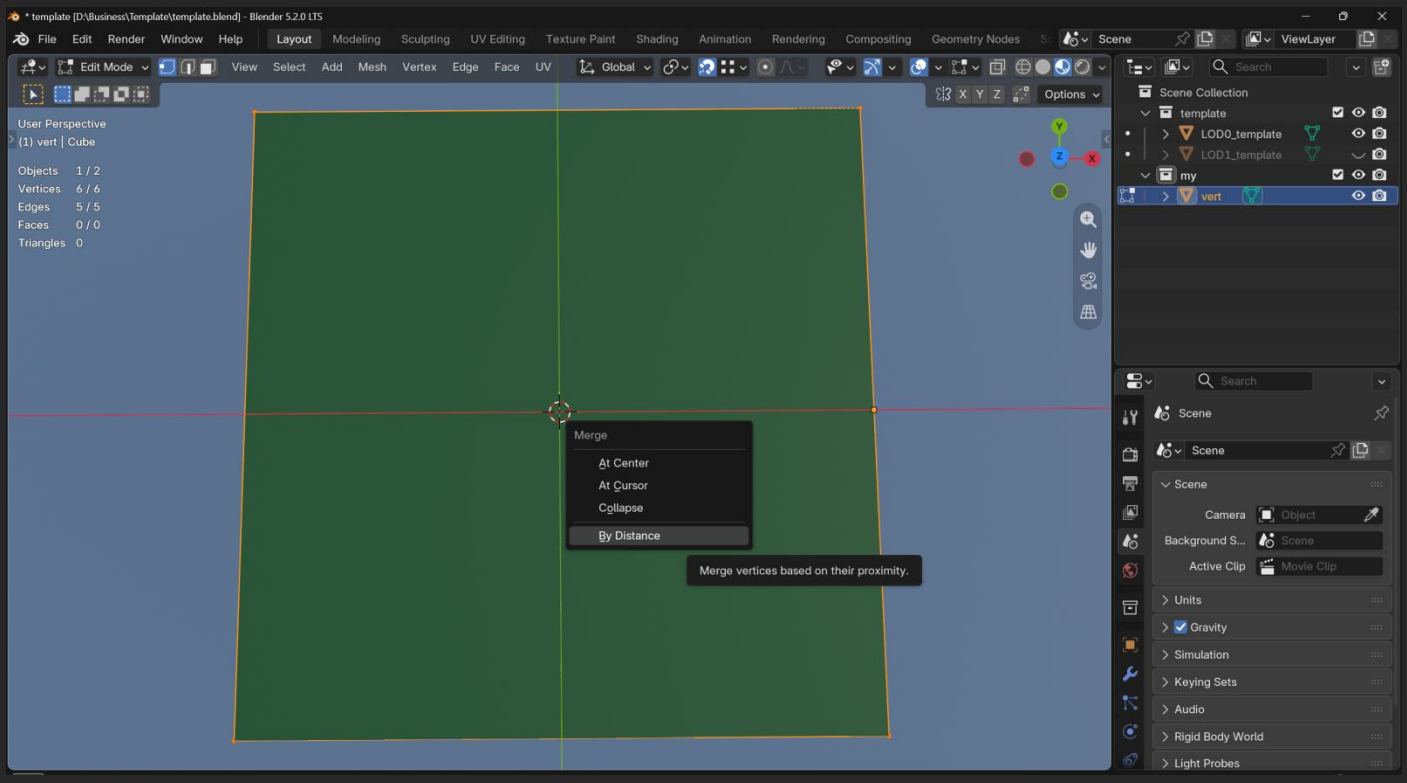


(Homework : 6 verts + 5 edges?!?)

6 verts : original vert + last vert (overlapping) + 1 on each corner. Total : 6 verts.

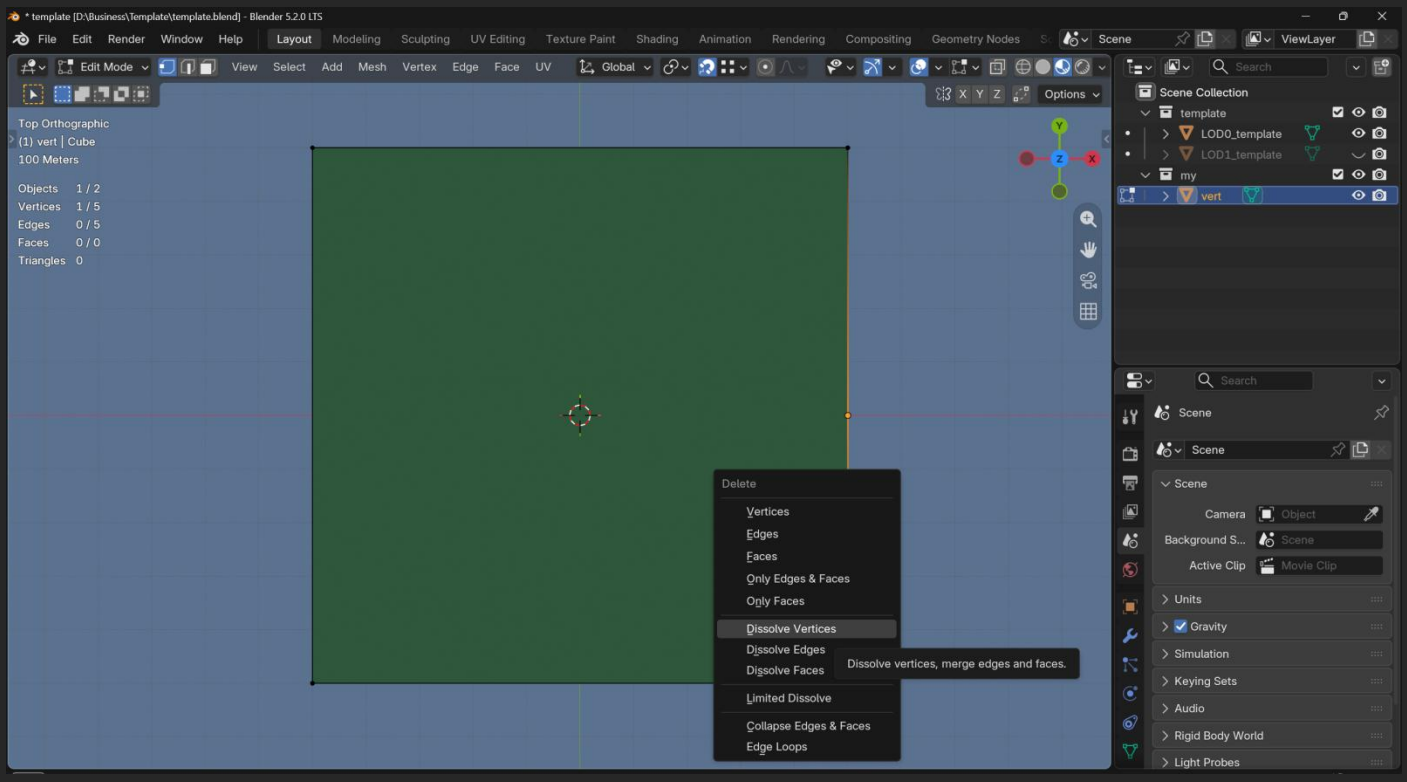
5 edges : original vert started in the middle of what becomes an edge. To get from there to the top-right corner = 1 edge. Connecting to the remaining 3 corners = 3 edges. And a final extrude takes us back to the starting point of extrusion = 1 edge. Total : 5 edges.

With all vertices selected (press A to confirm), press M to bring up the Merge menu + click ‘By Distance’.

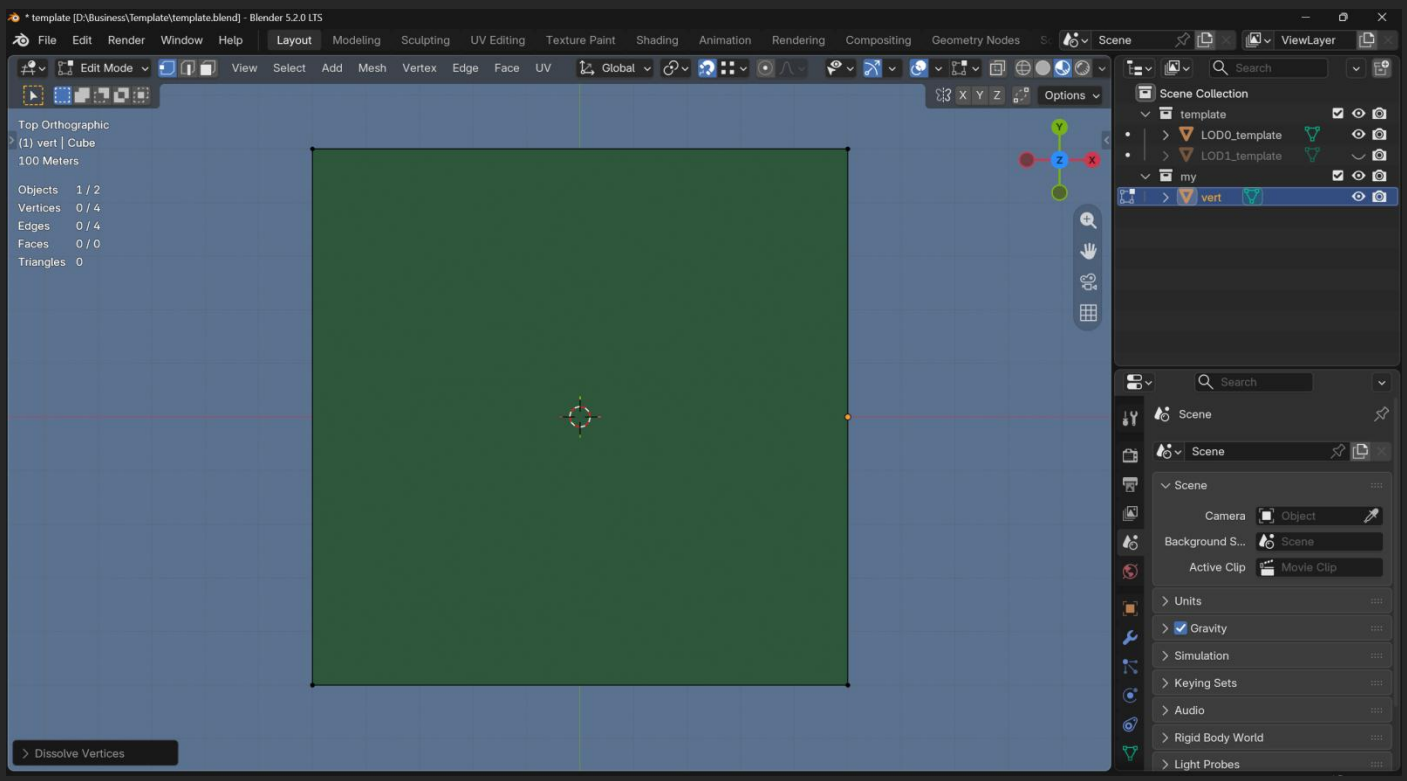


1 vertex has been removed. (Challenge : which vertex?). 5 remain : 1 on each corner + 1 in the middle of an edge where we started/ended. We don’t need that vertex to exist. It’s complicating the geometry of the square by intersecting an edge, so let’s get rid of it.

Select that vertex by holding down the left mouse-button + dragging a selection box around it. Make sure it’s the only vertex selected + press X to bring up the Delete menu.



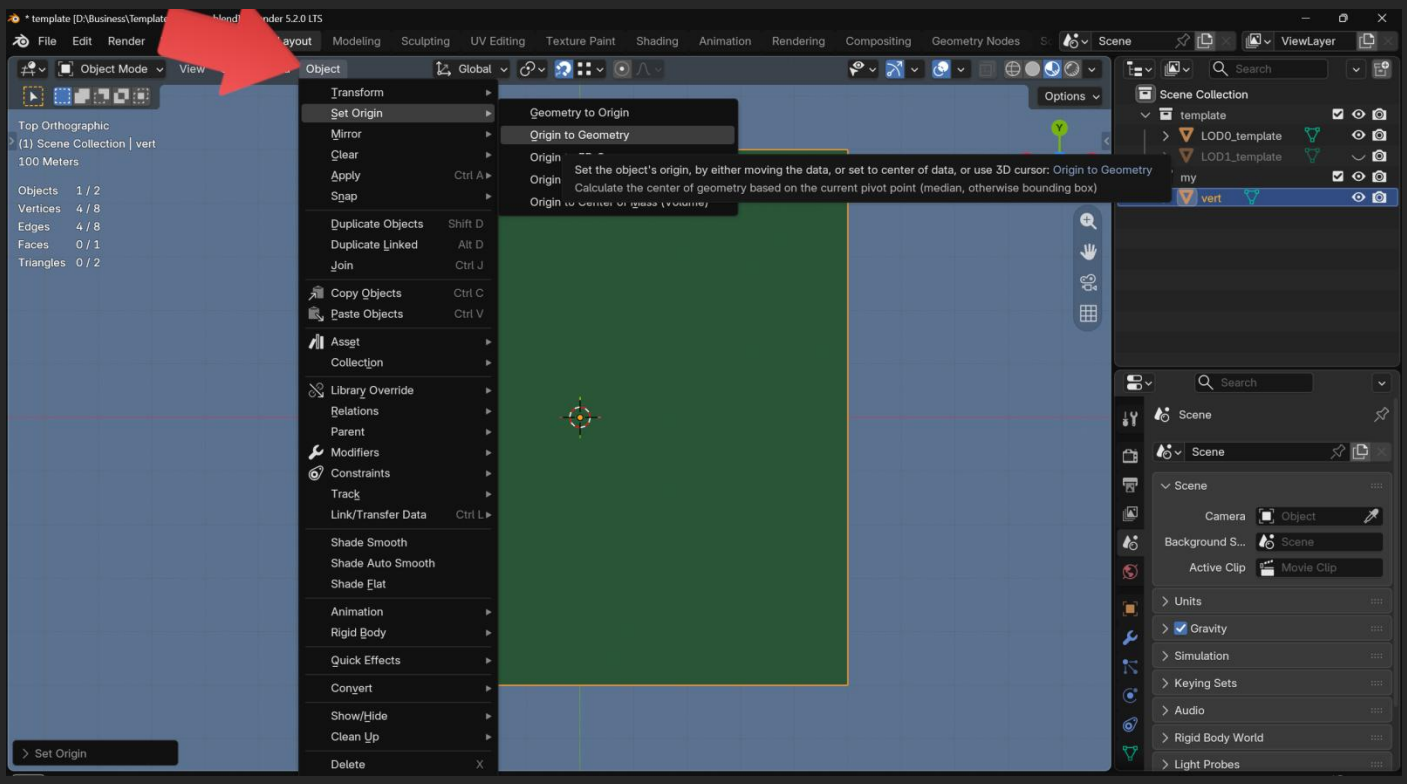
Select Dissolve Vertices.



But wait… it looks like a vertex is still there! It's not a vertex, it's the Object's Origin.

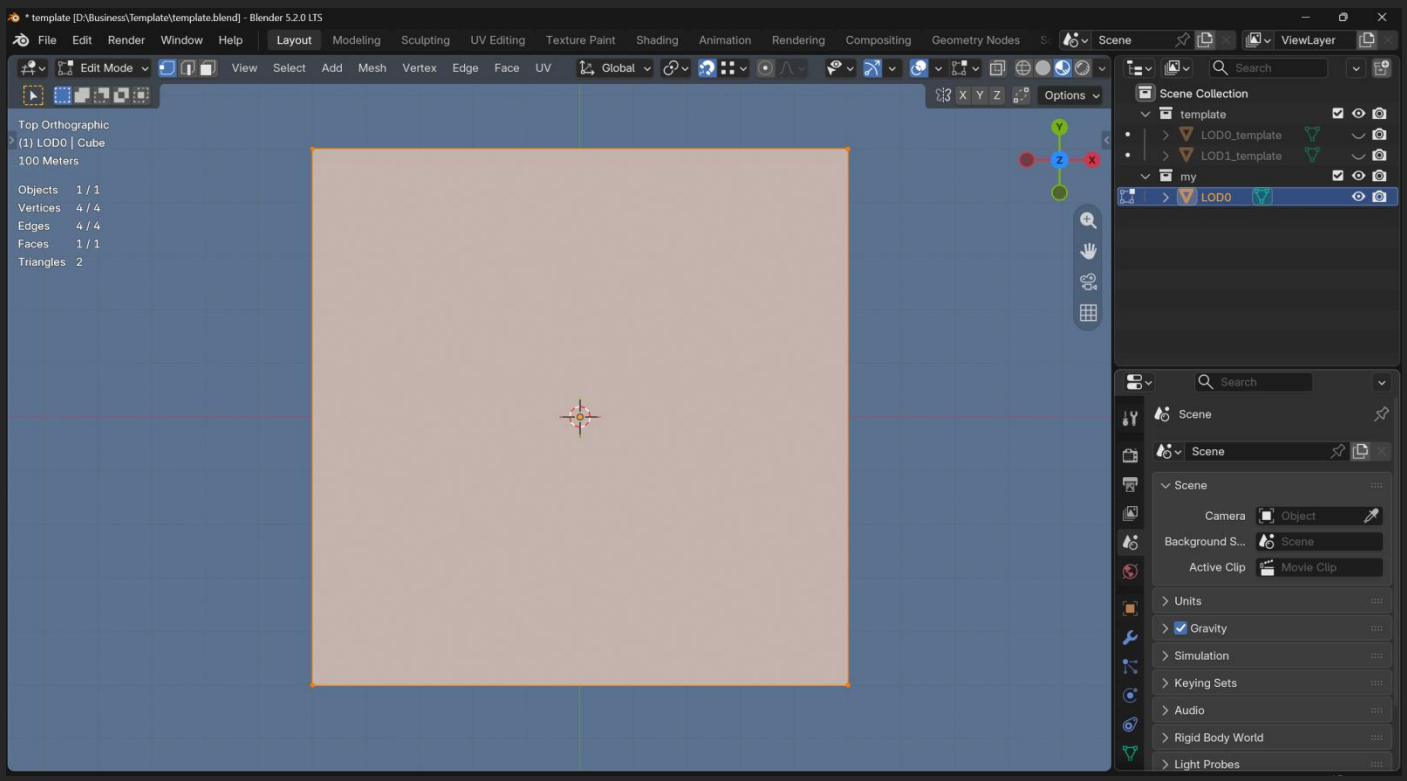
(homework) Research the Origin. Why was it automatically set at that point?

To make the Origin snap to the centre, TAB into OBJECT mode -> Object menu -> Set Origin -> Origin to Geometry.



Let's make a face.

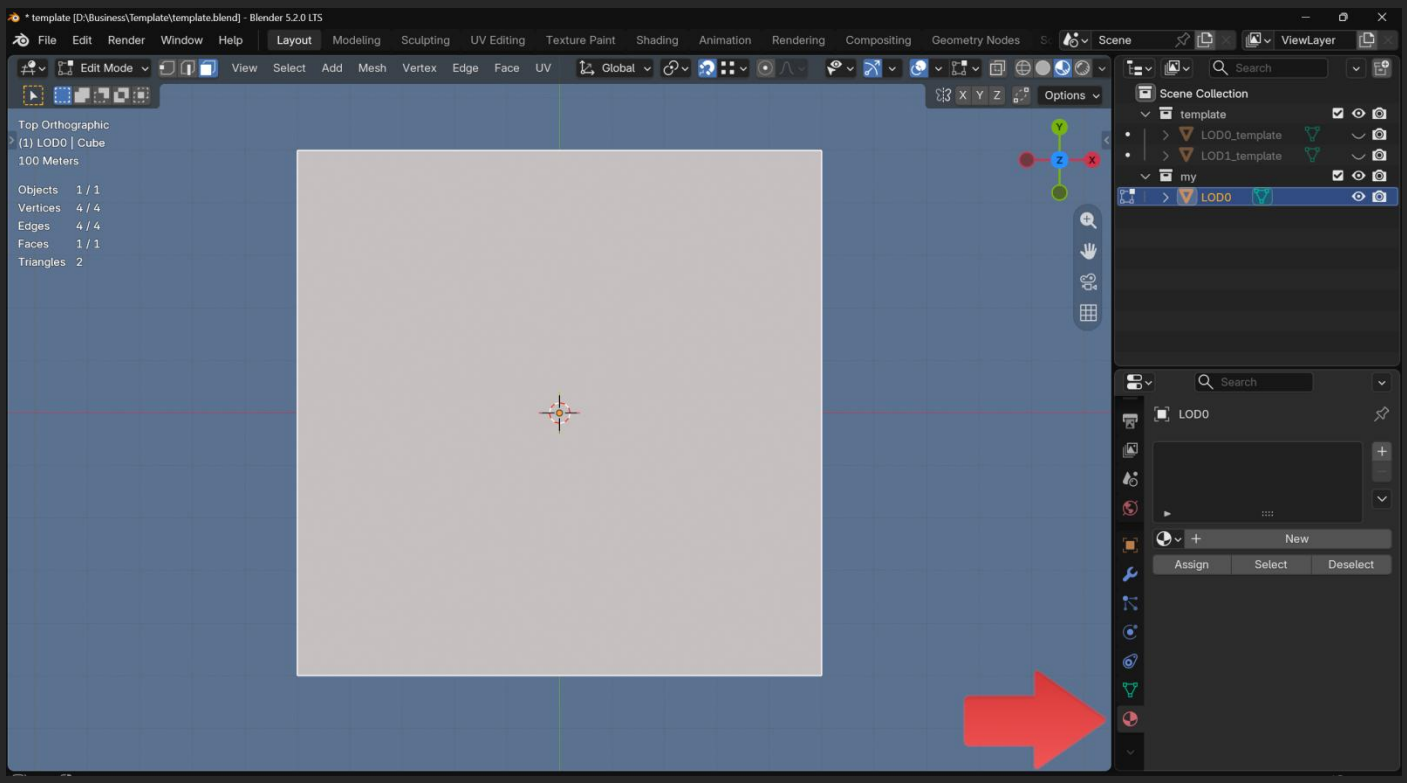
Our object is still named vert, so rename it LOD0. Hide the LOD0_template, TAB into EDIT mode (with our LOD0 selected), select all the vertices with A + press F to create a face.



Check that this model fits perfectly with the template model by unhiding LOD1_template in Outliner. Hide LOD1_template when you're satisfied.

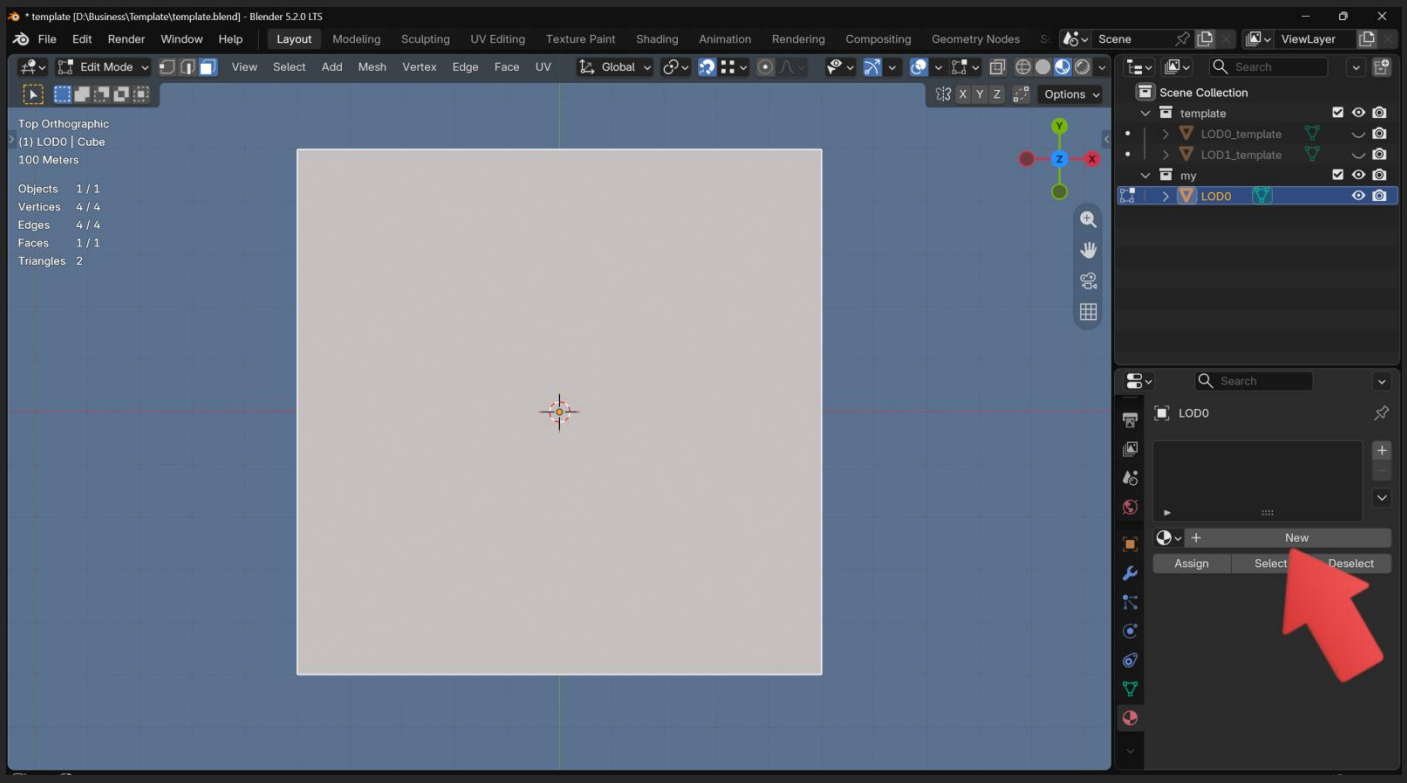
(Challenge) What color is the square?

Press 3 to go into Face mode. Open the Material panel in the Properties window. You may have to use the scroll-wheel whilst hovering over the icon list to reveal it.

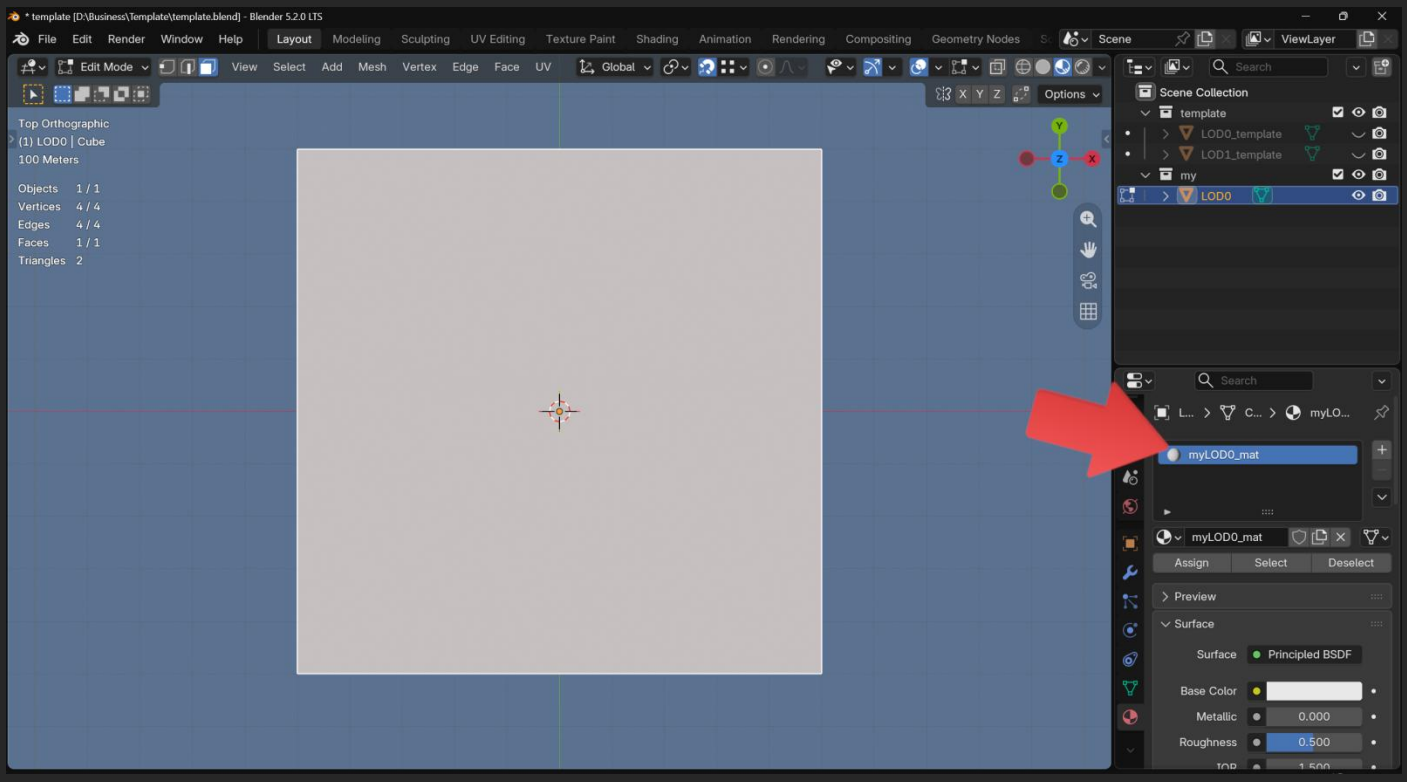


An empty Materials list means that no material has been assigned to this face, even though it's represented on the 2D monitor in beige.

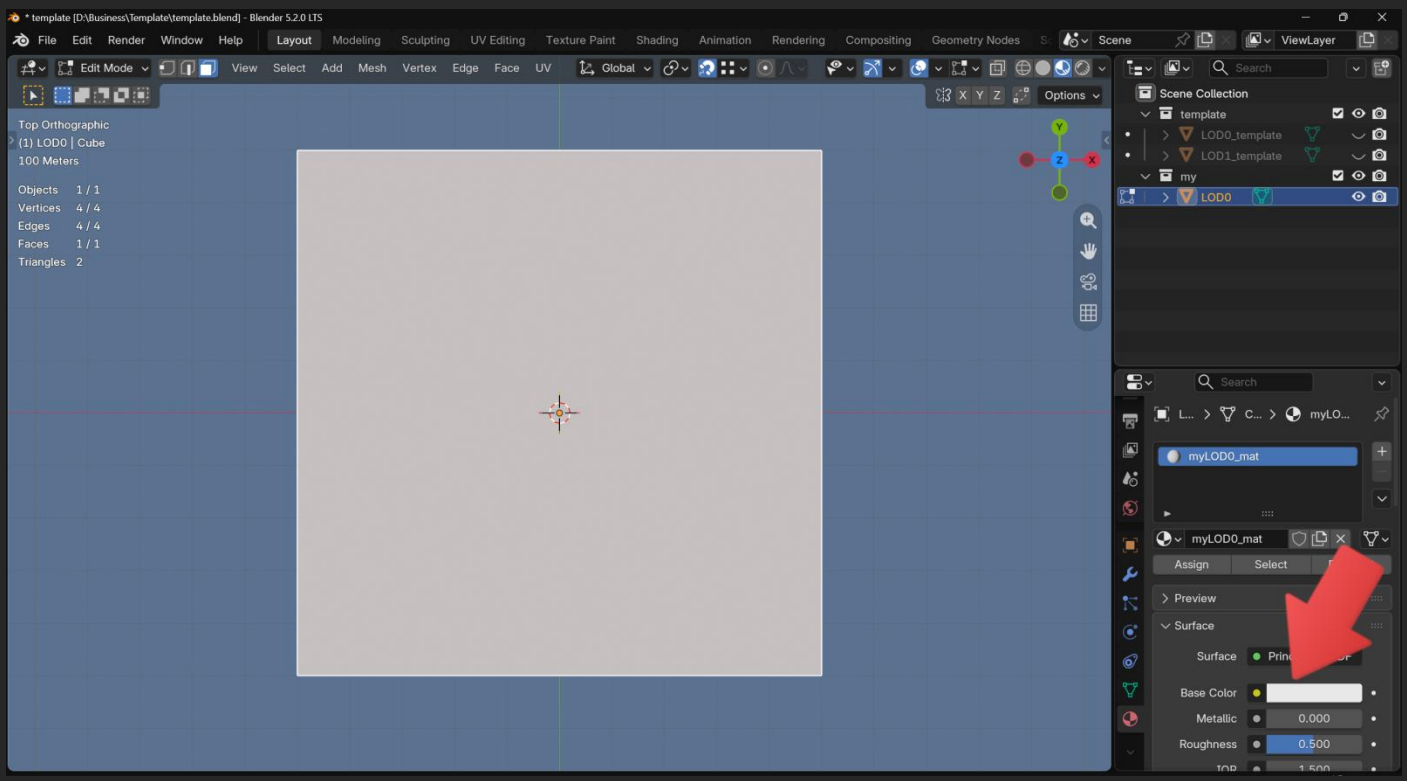
In the Properties window, click New.



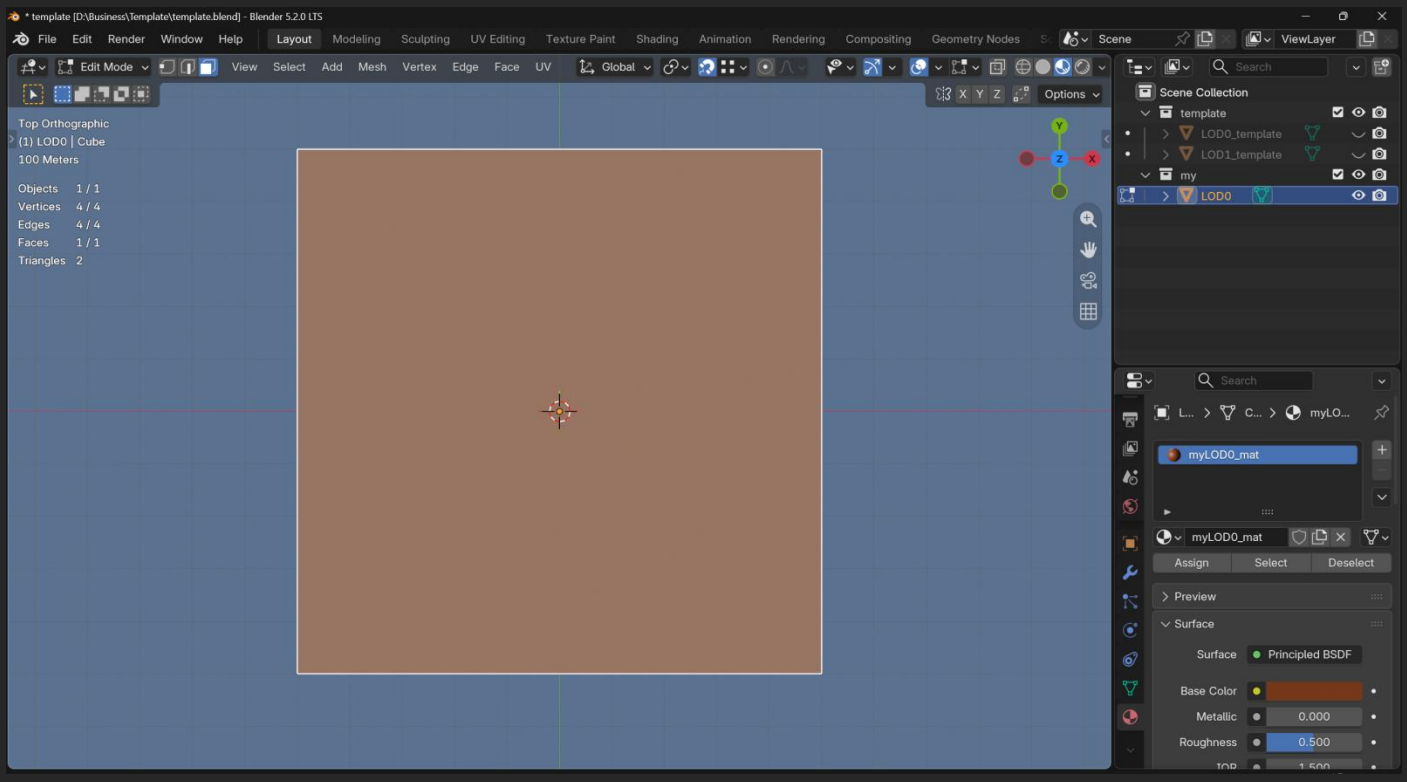
Rename the Material that's now populating the list to myLOD0_mat.



Change the color of the Material by clicking in the white box next to Base Color in the Properties window.



Play with the many different ways to pick a color. When ready, choose something other than green (to tell it apart from the template). Blender automatically applies an only Material to a selected Face (usually, we have to explicitly Assign a Material).

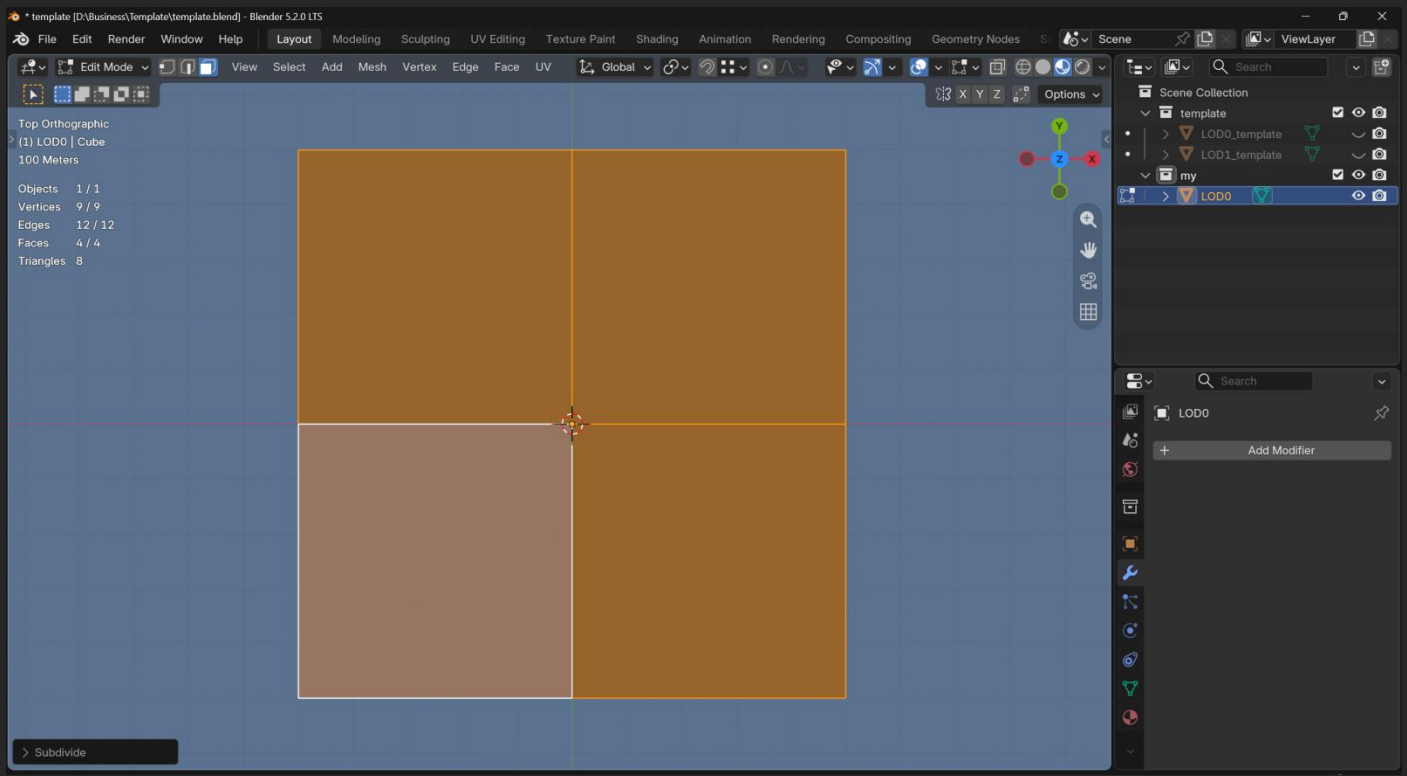


You've now finished replicating the LOD0_template model. It's a flat 2D square… let's landscape it.

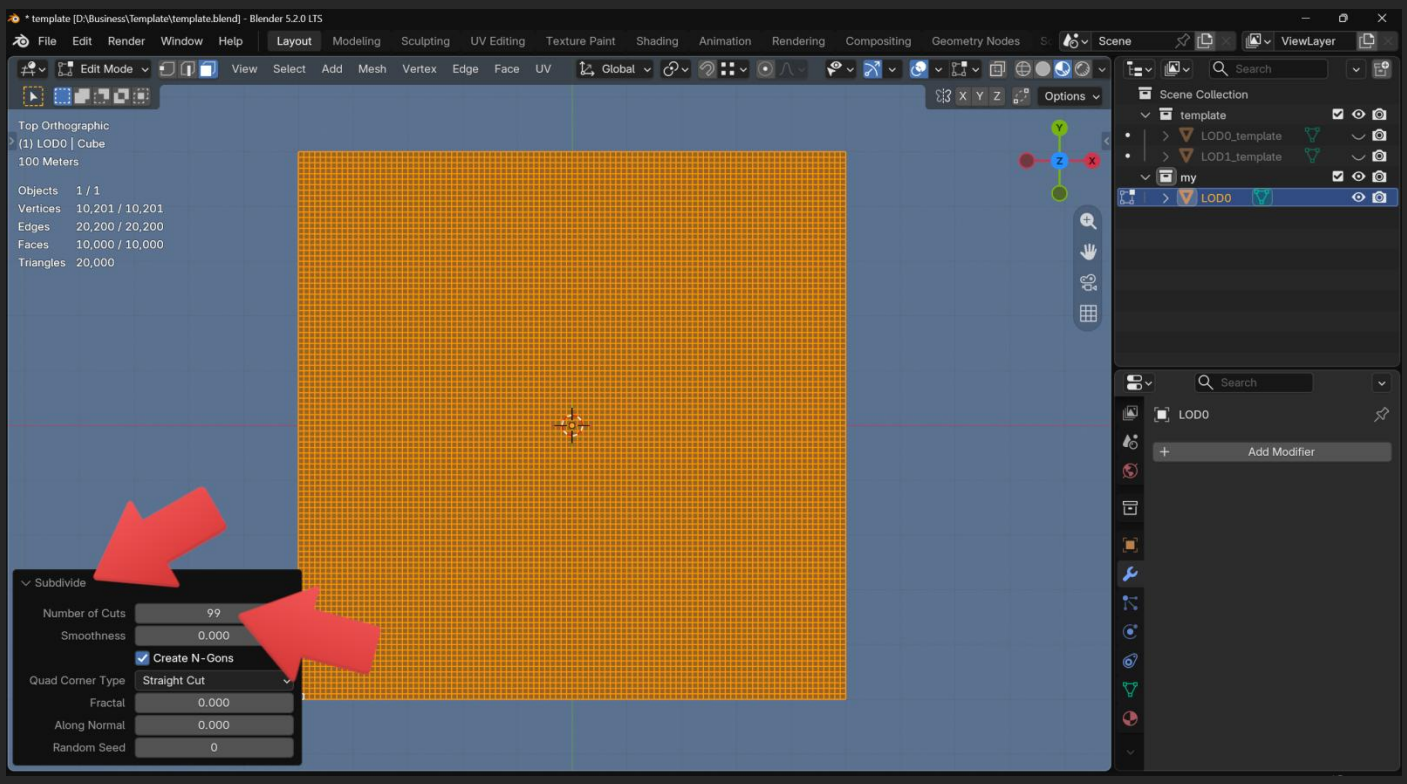
iii. Landscape

Blender has a snap function that's really useful, but turn it off for now with SHIFT+TAB.

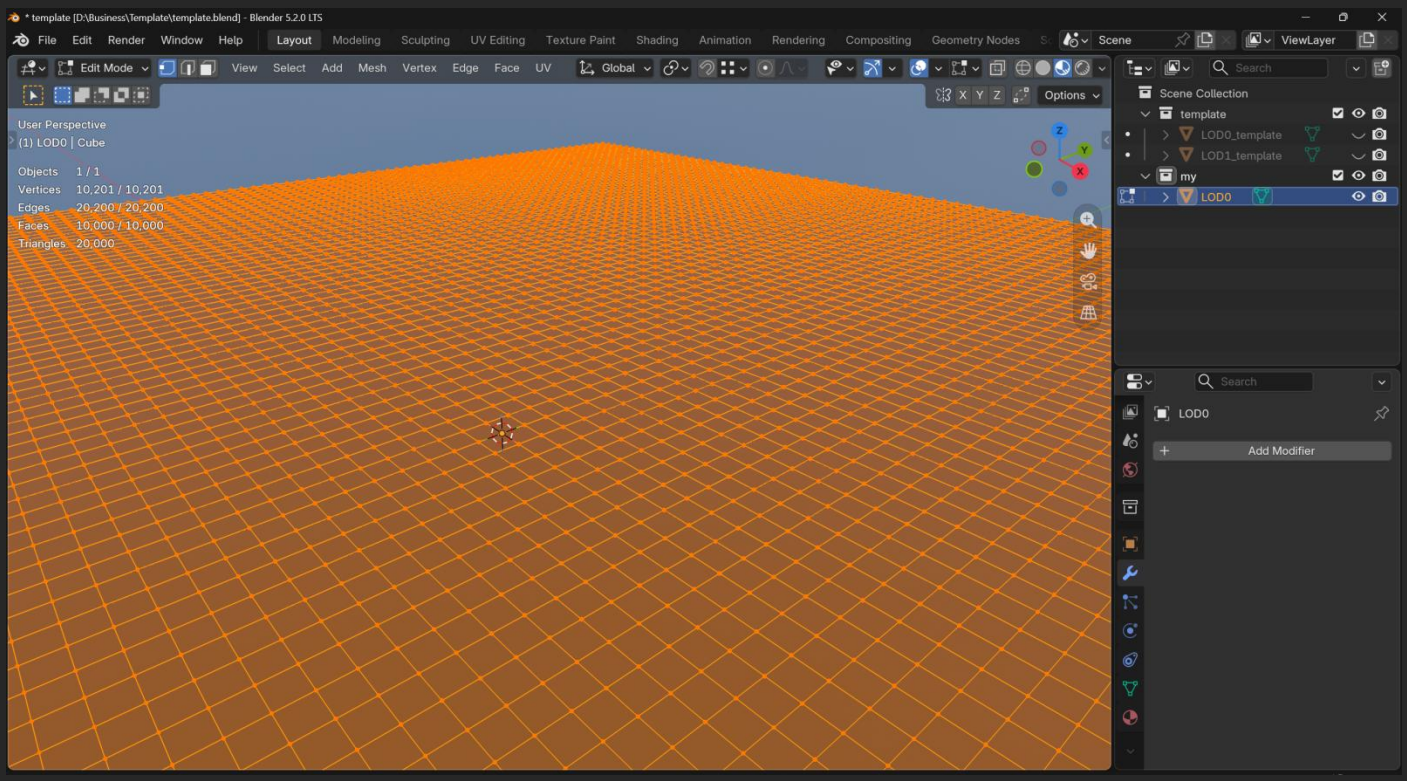
In EDIT/FACE mode, right-click on the selected model -> Subdivide



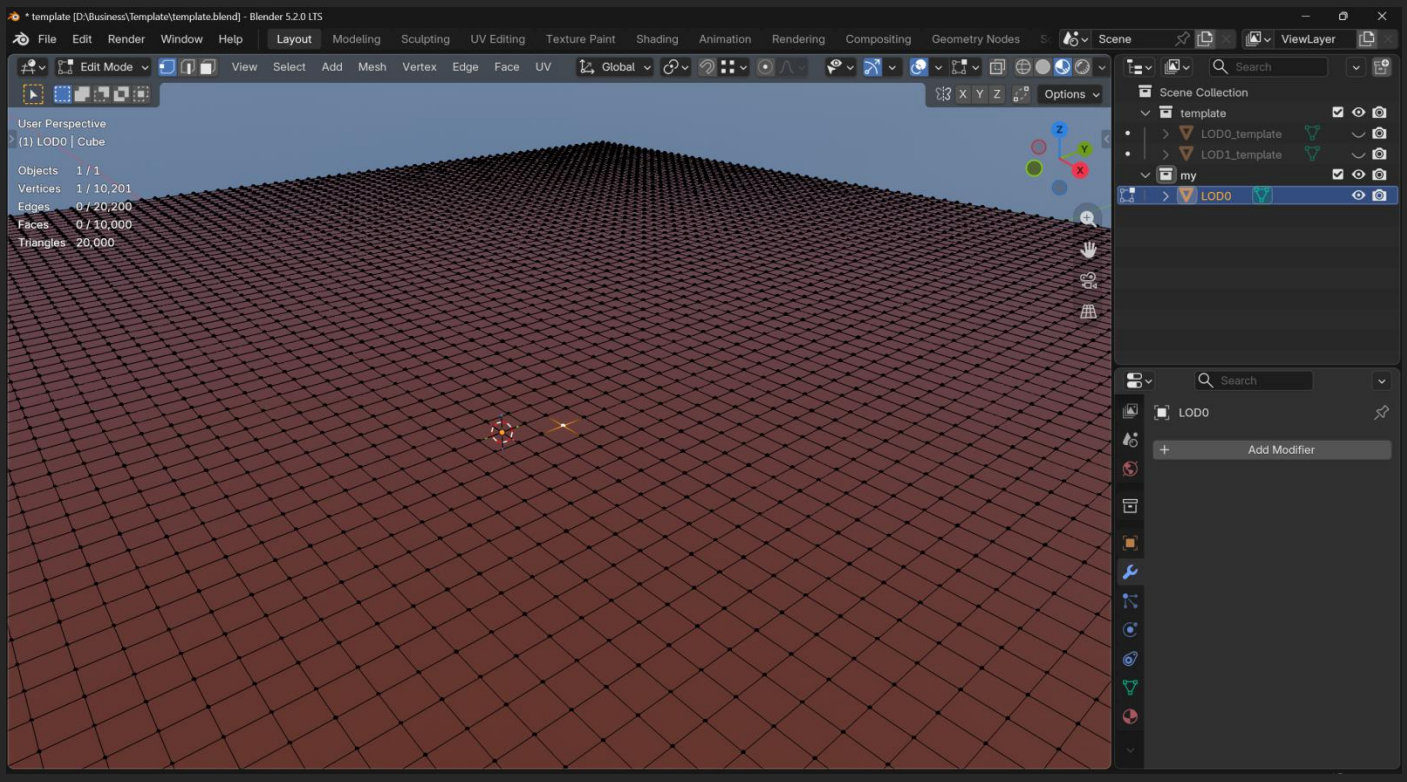
Click the >Subdivide button in the bottom-left corner + change Number of Cuts to 99.



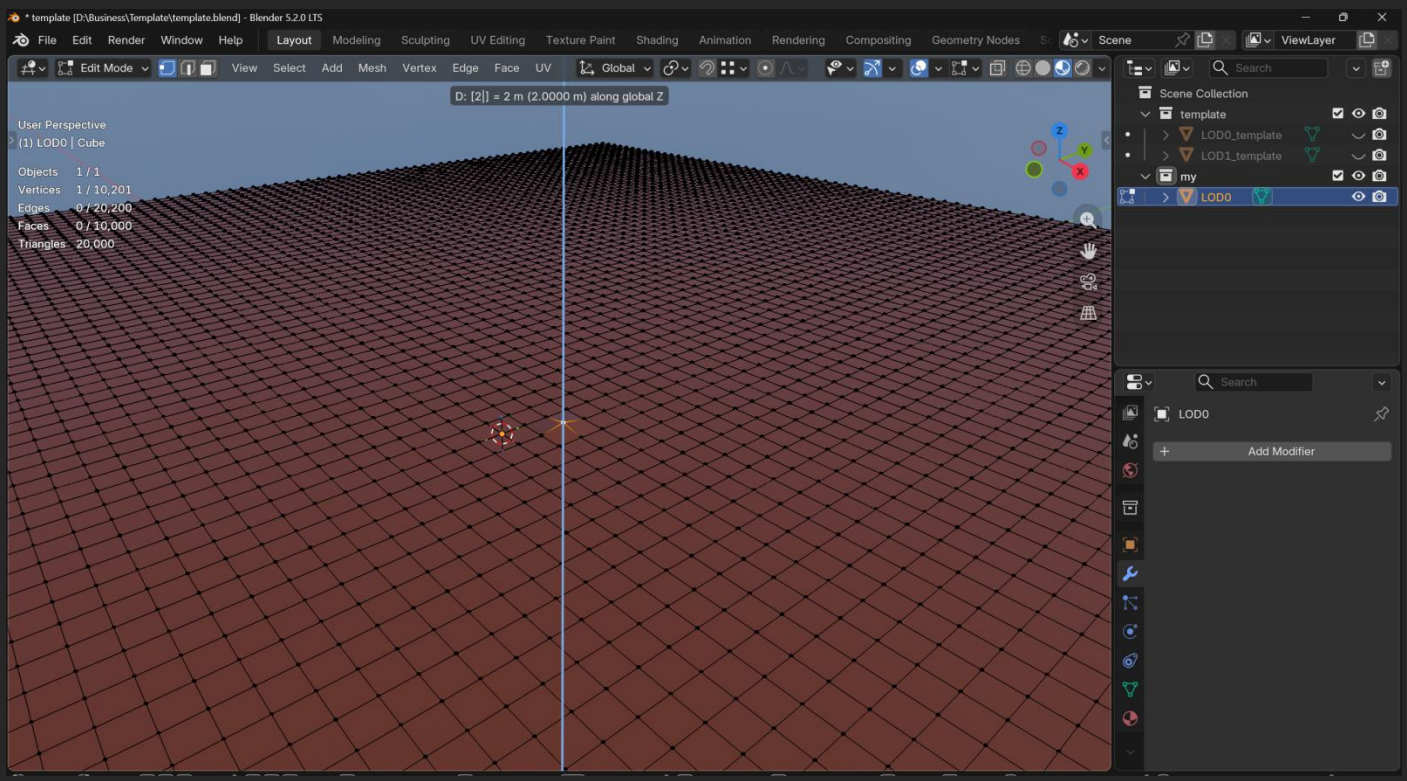
Blender has automatically added a mesh of vertices, linked them with edges + created triangles (two for each square). We're still looking at a 2D plane… let's drag the view to better show 3D space, enter VERTEX mode by pressing 1 + zoom in to near the centre.



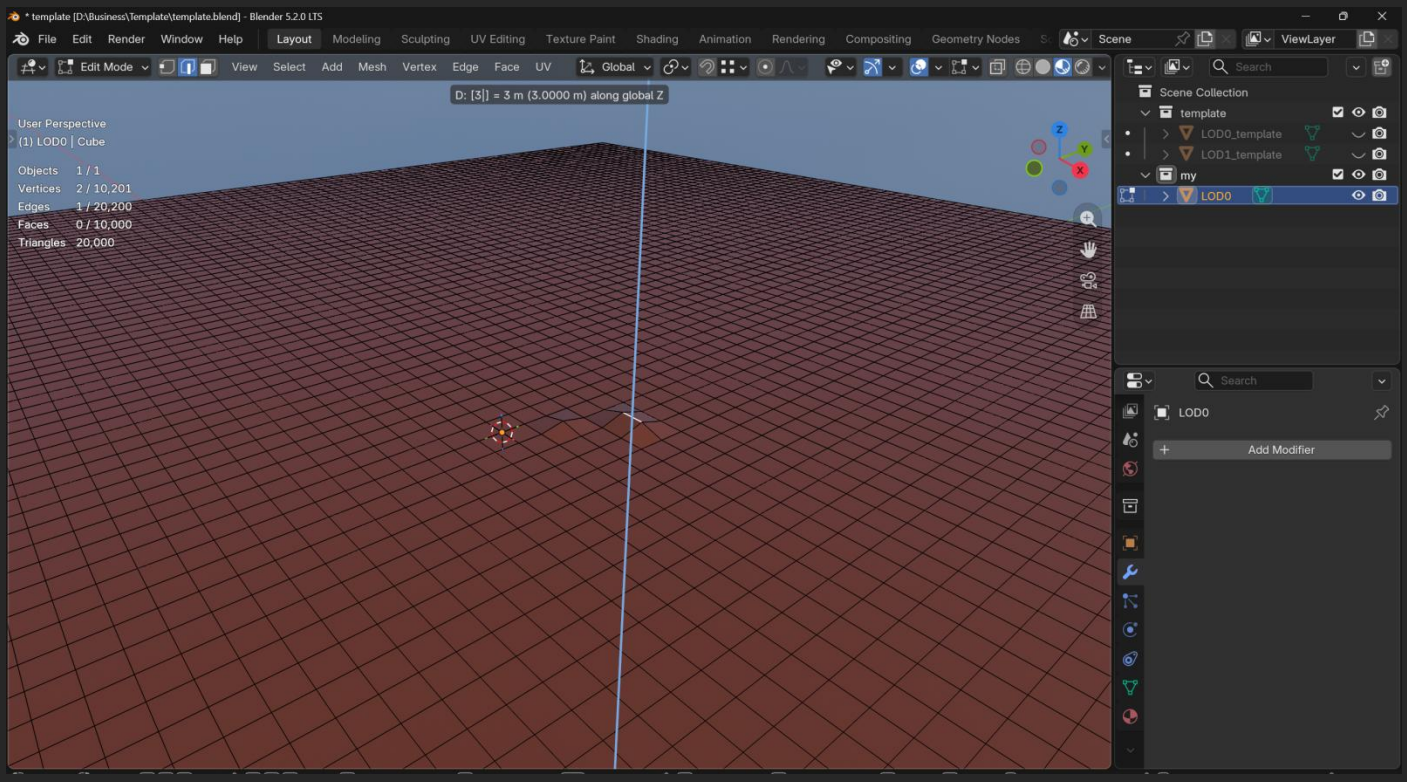
Select any vertex by moving the mouse close to it + left-clicking.



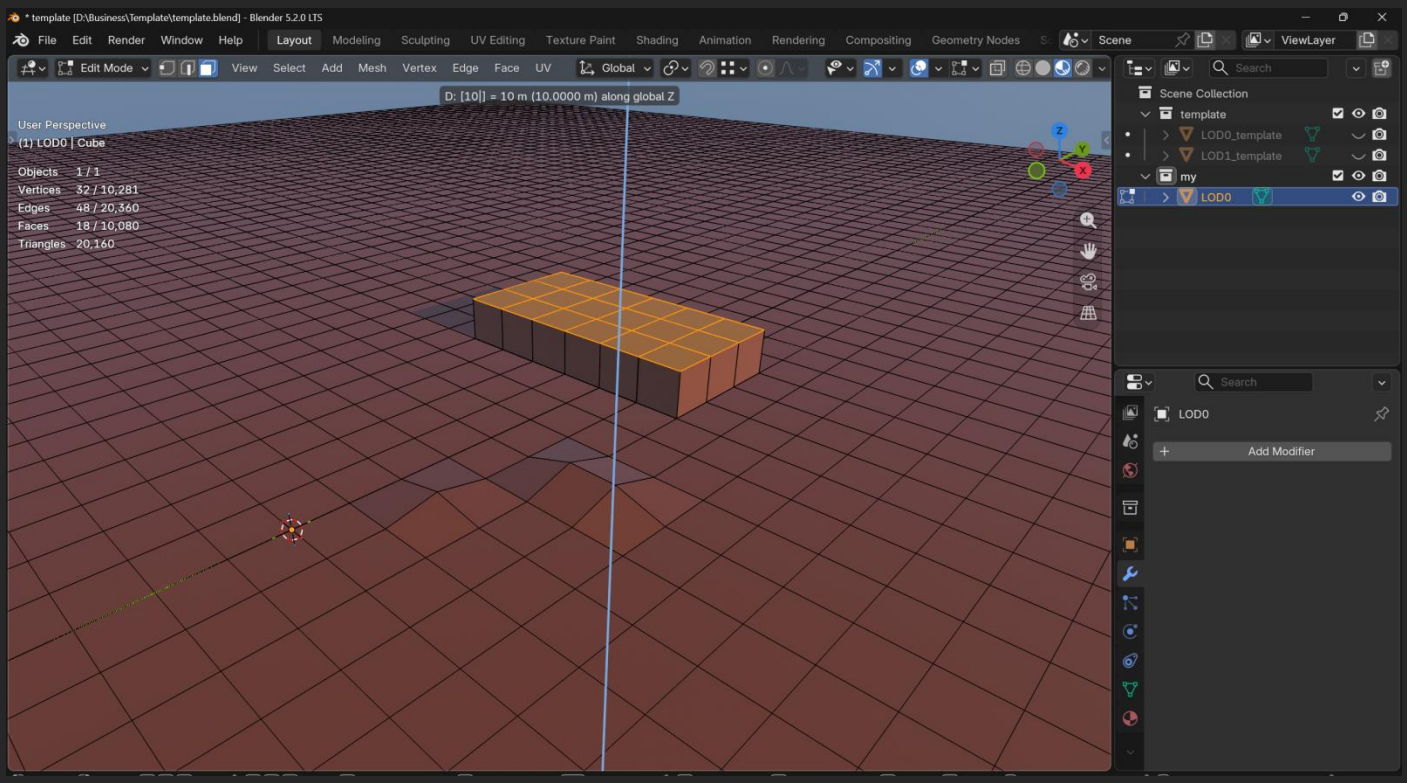
Press G to enter MOVE mode, press Z to specify the up/down axis + enter 2 to move the vertex 2 metres up.



Press 2 to enter edge mode. Select an edge + move it 3 metres up.

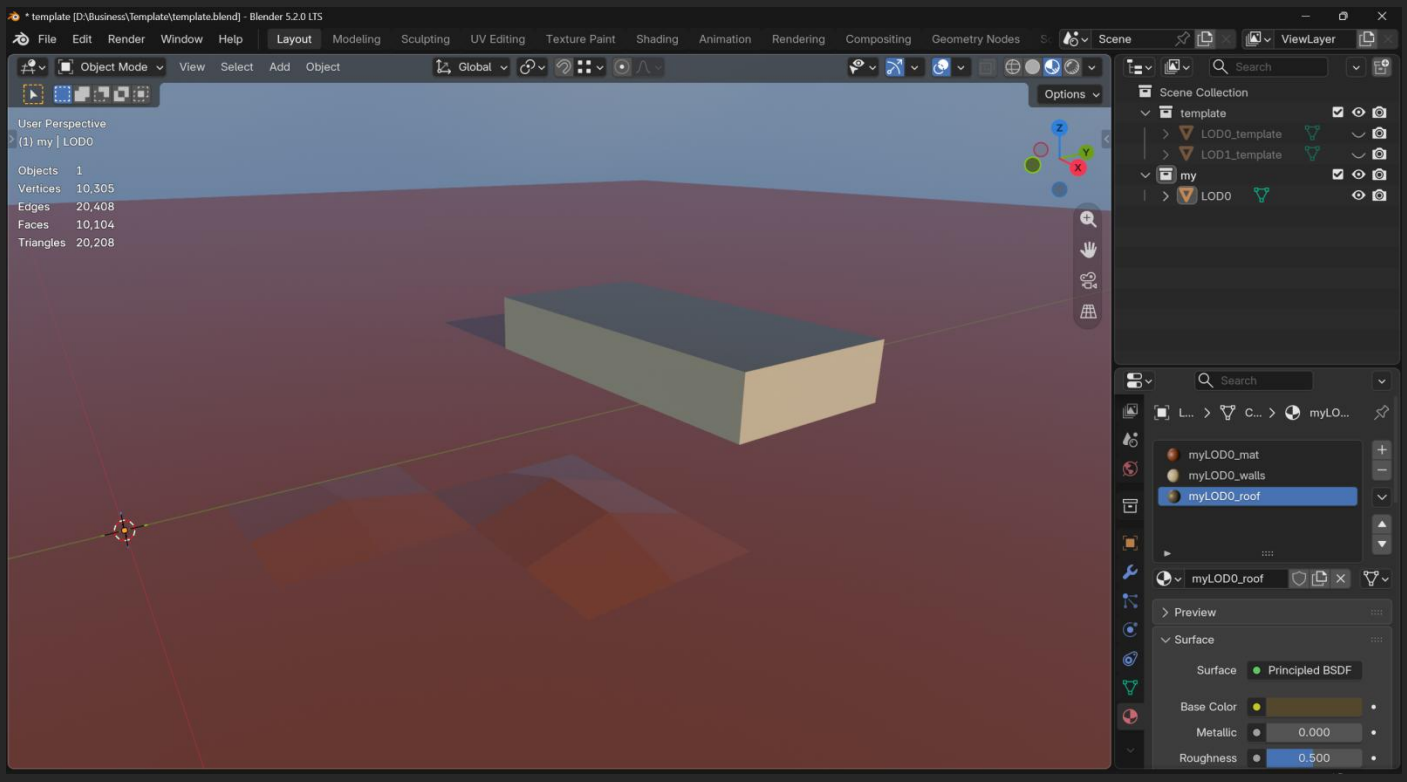


Press 3 to enter FACE mode. Select a block of faces that could be the foundation of a house. Press E to extrude + move the faces 10 metres up to form a house. Blender may start in z-axis mode when you press E… there's no need to press Z if it's already selected (you can tell by the presence of the blue line representing the z-axis).



MOVE (M) + EXTRUDE (E) are powerful creation tools for 3D modelling.

Select the newly-created faces that represent the four walls of the house. Create a new material + assign it to the walls. Paint the roof a different material. TAB into OBJECT mode to get a better view.



Save your project.

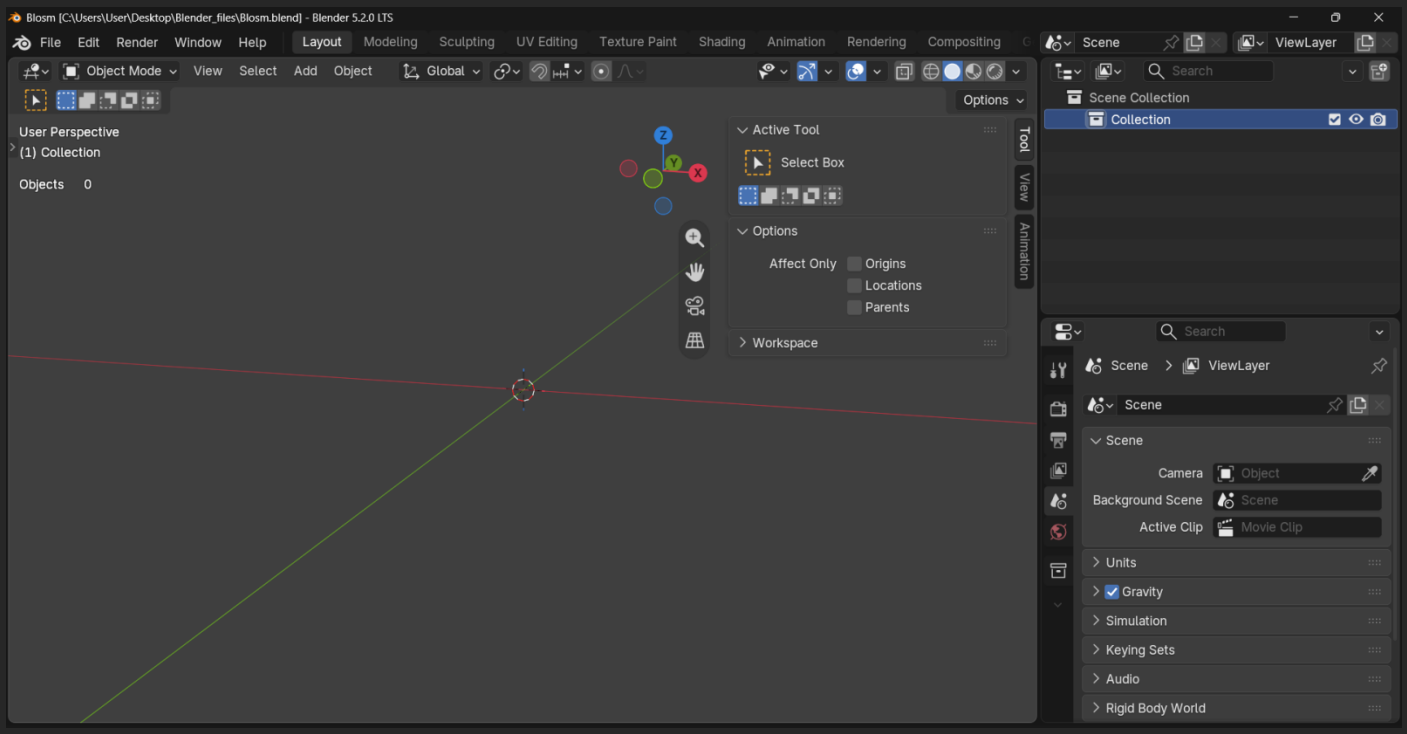
(iv) Blosm.

The Blosm[23] add-on for Blender massively simplifies import of landscape data. There are detailed instructions for installation, including on how to obtain a GIS[24] token (ARCGIS recommended). The authors suggest a donation for the base version (if you can) or ~£15 for the Pro version.

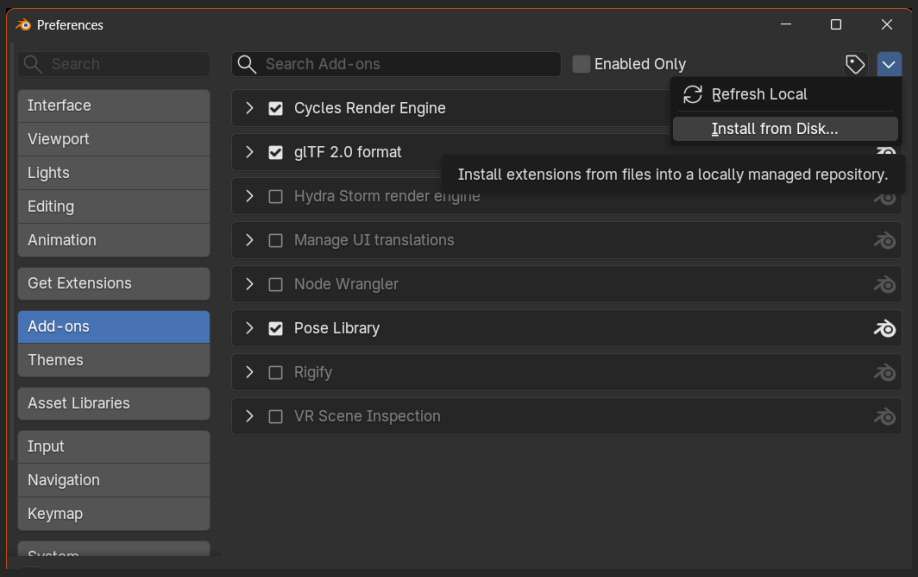
Open a new project.

Press N to open the Tools panel.

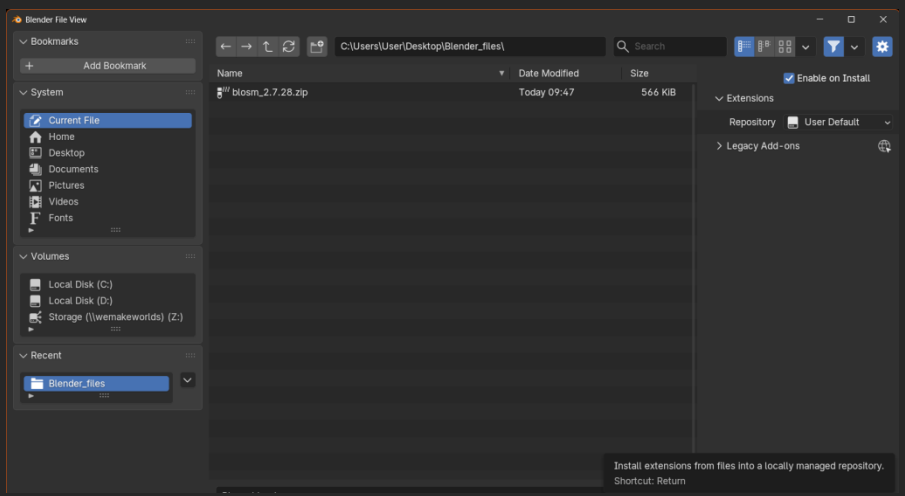
[23] github.com/vvoovv/blosm
[24] Geographic Information System



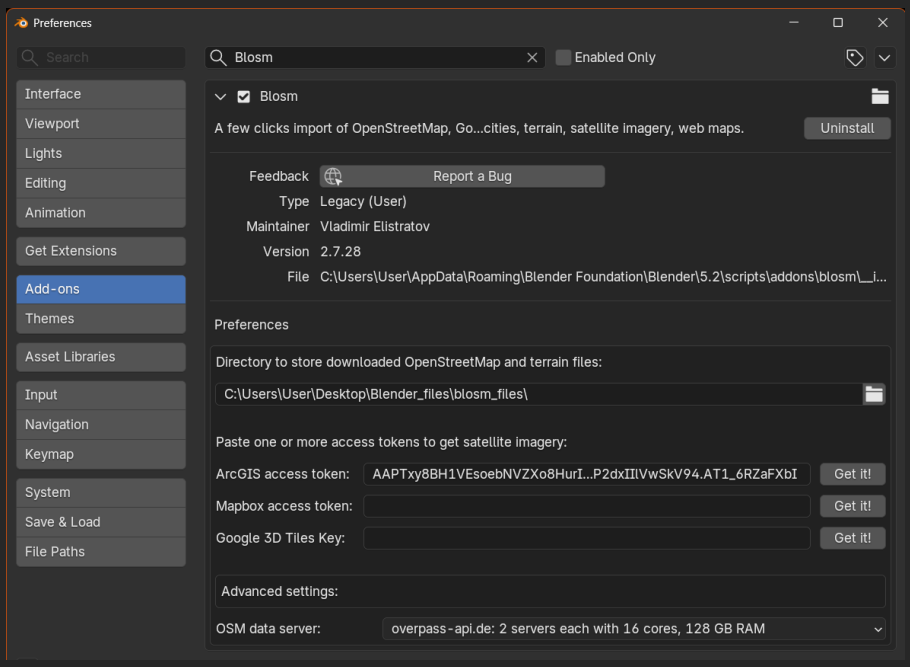
Edit -> Preferences -> Add-ons -> Add-on settings (chevron under exit button) -> Install from disk.



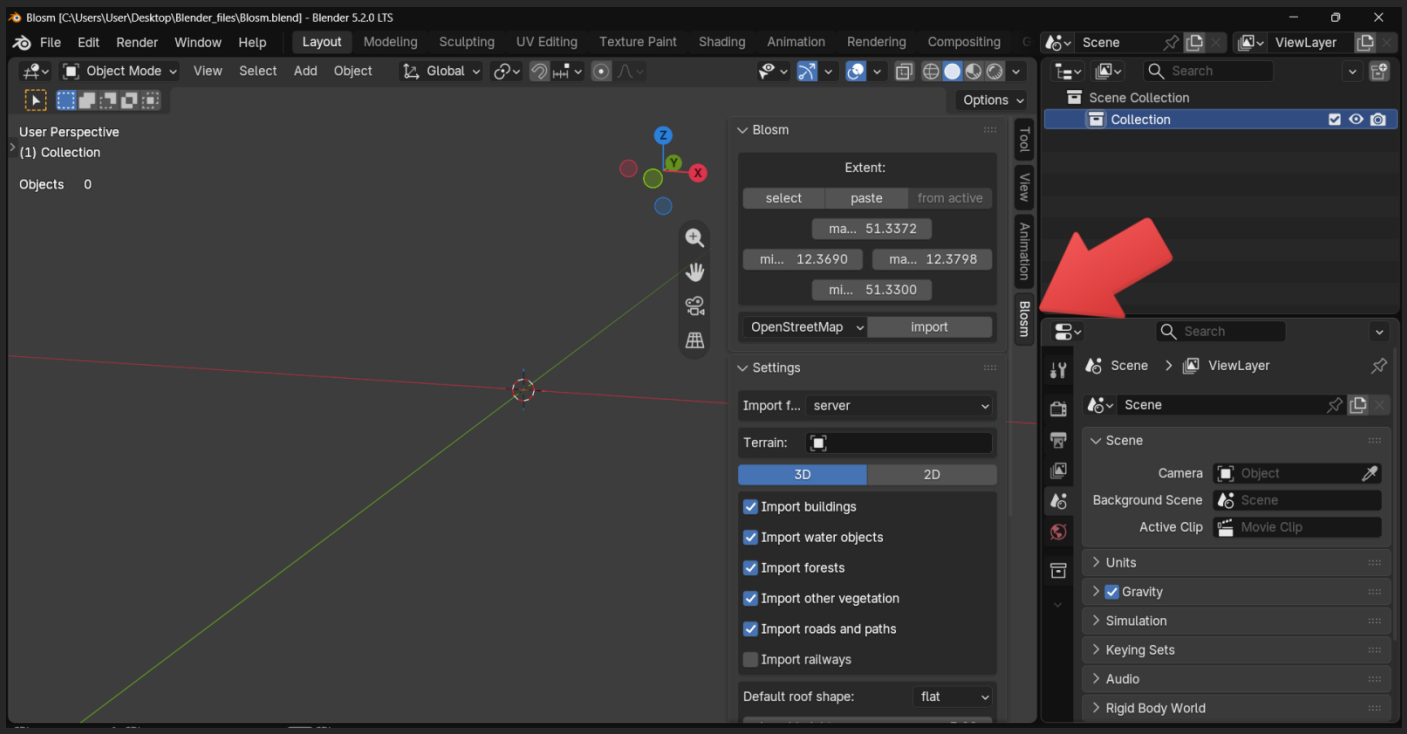
Point Blender File View to the Blosm.zip file you downloaded earlier.



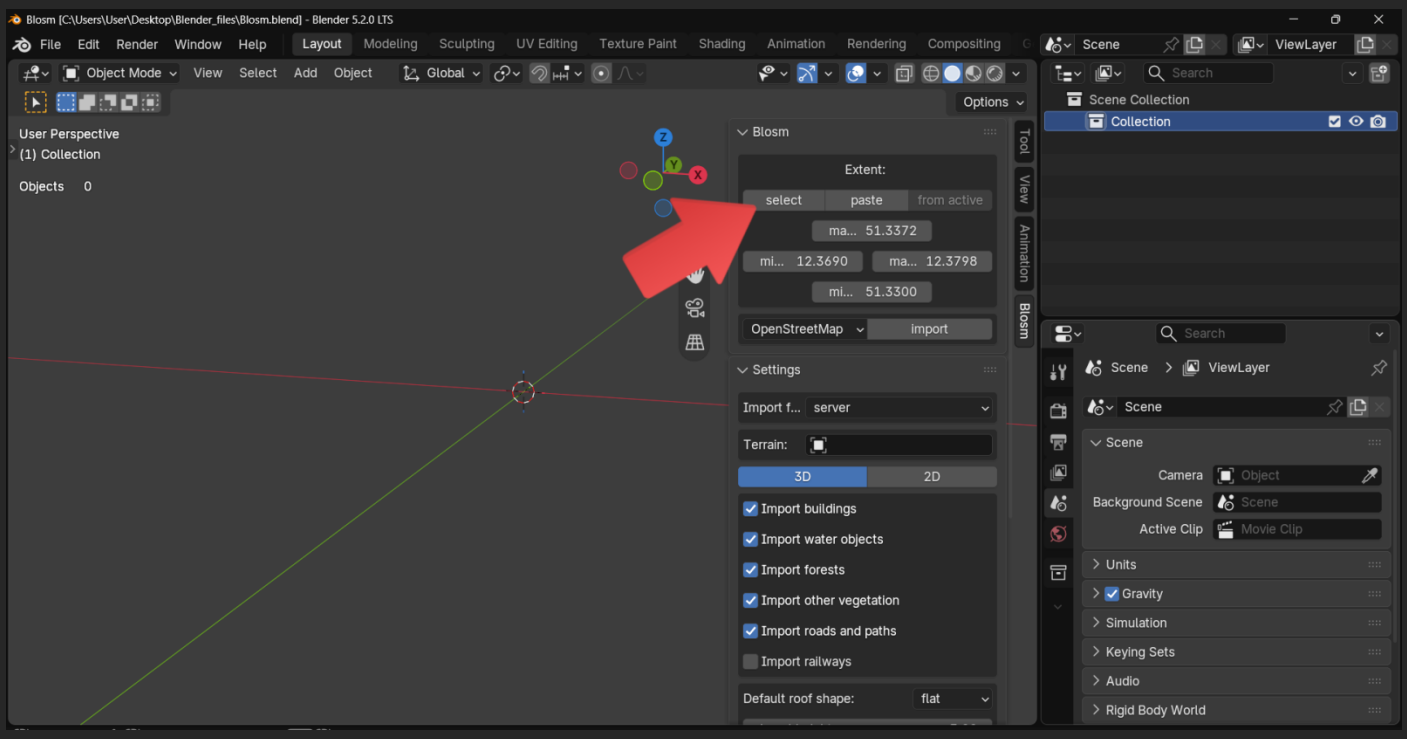
Point Blosm to a directory it can use to store files + enter an access token to get satellite imagery.



Close Preferences + click on the newly-appeared Blosm entry in the Tools menu.

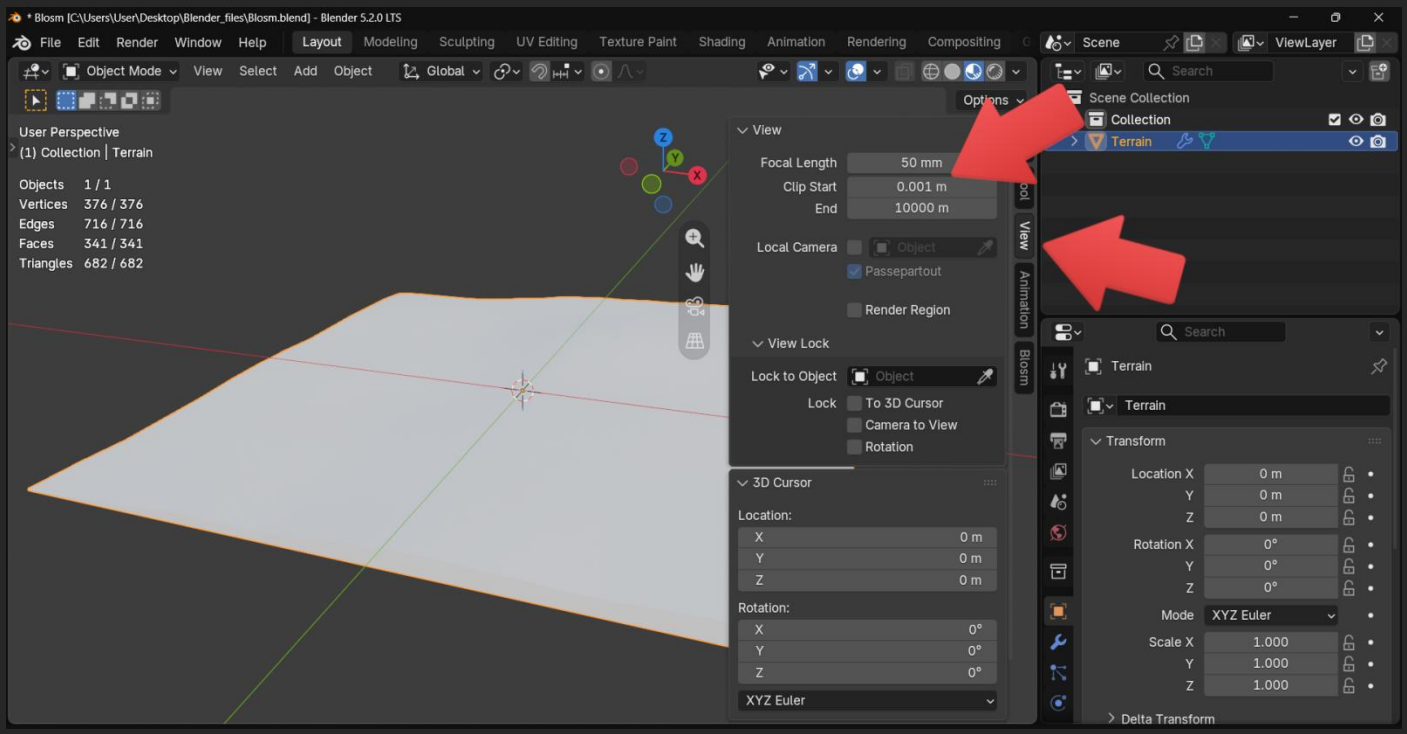


Click Select to open up a map of the world in your default web-browser. Follow the instructions to select an area ~300-500m wide + click Copy. Back in Blender, click Paste to automatically fill in the Extent. Here, I've chosen an area around Worcester Cathedral.

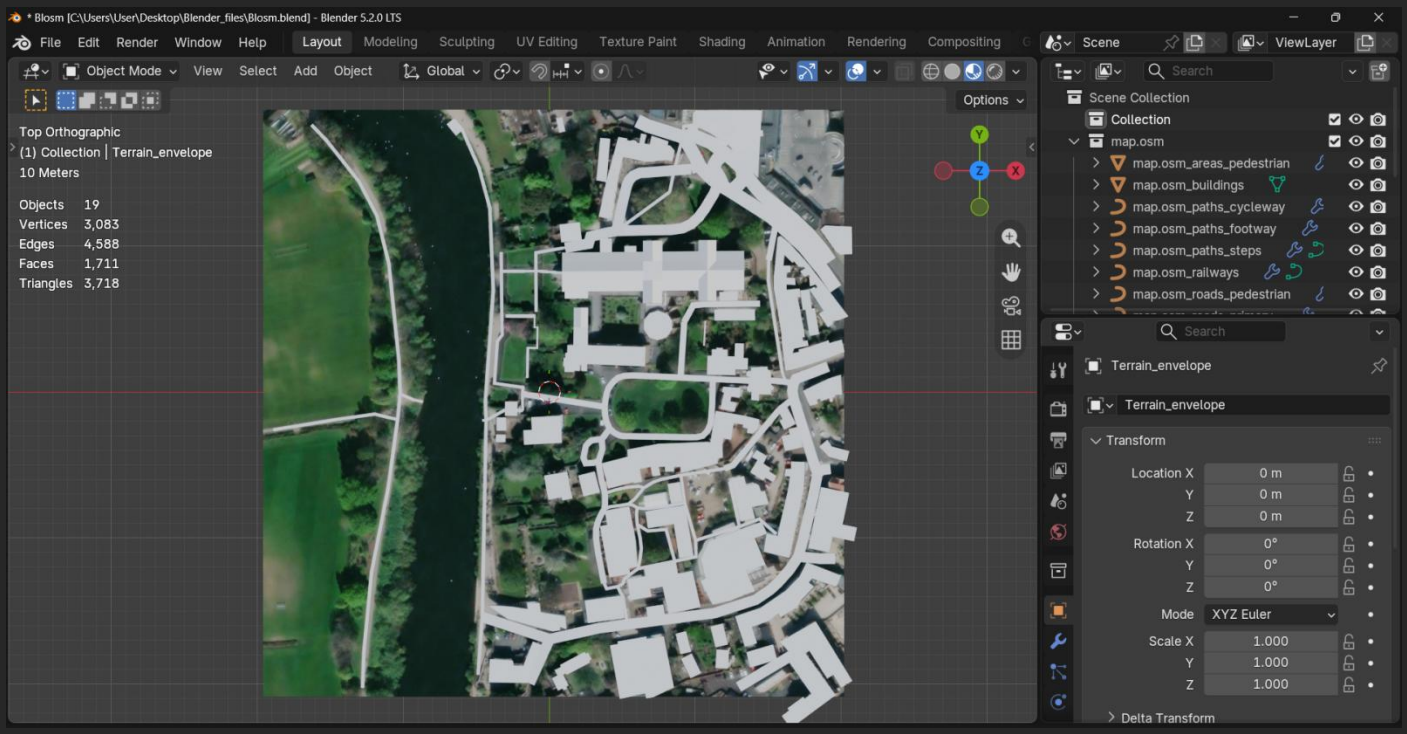


Immediately below the Extent values, change ‘OpenStreetMap’ to ‘terrain’ + click Import.

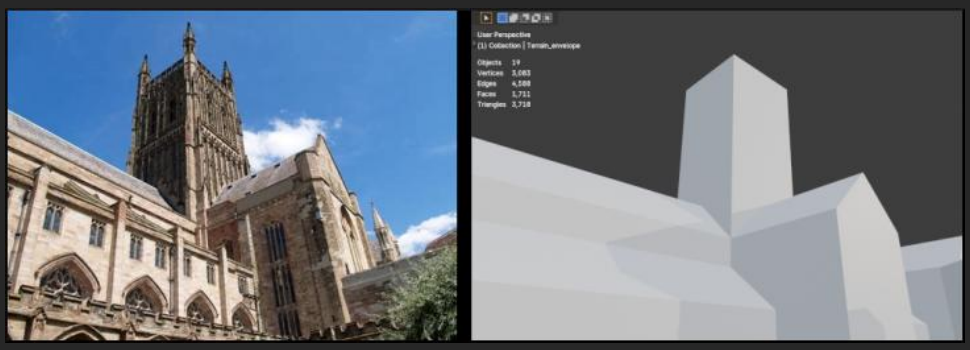
Change the Viewport Shading to Material + increase the View distances to 0.001m Clip Start, 10000m End.



Back in the Blosm tab, use the drop-down menu (to the left of the ‘import’ button) to import an image overlay + OpenStreetMap. Centre the view to take it all in.



SHIFT+B let's you draw a rectangle on the model to zoom to that camera position. This allows accurate jumping to any part of the map.



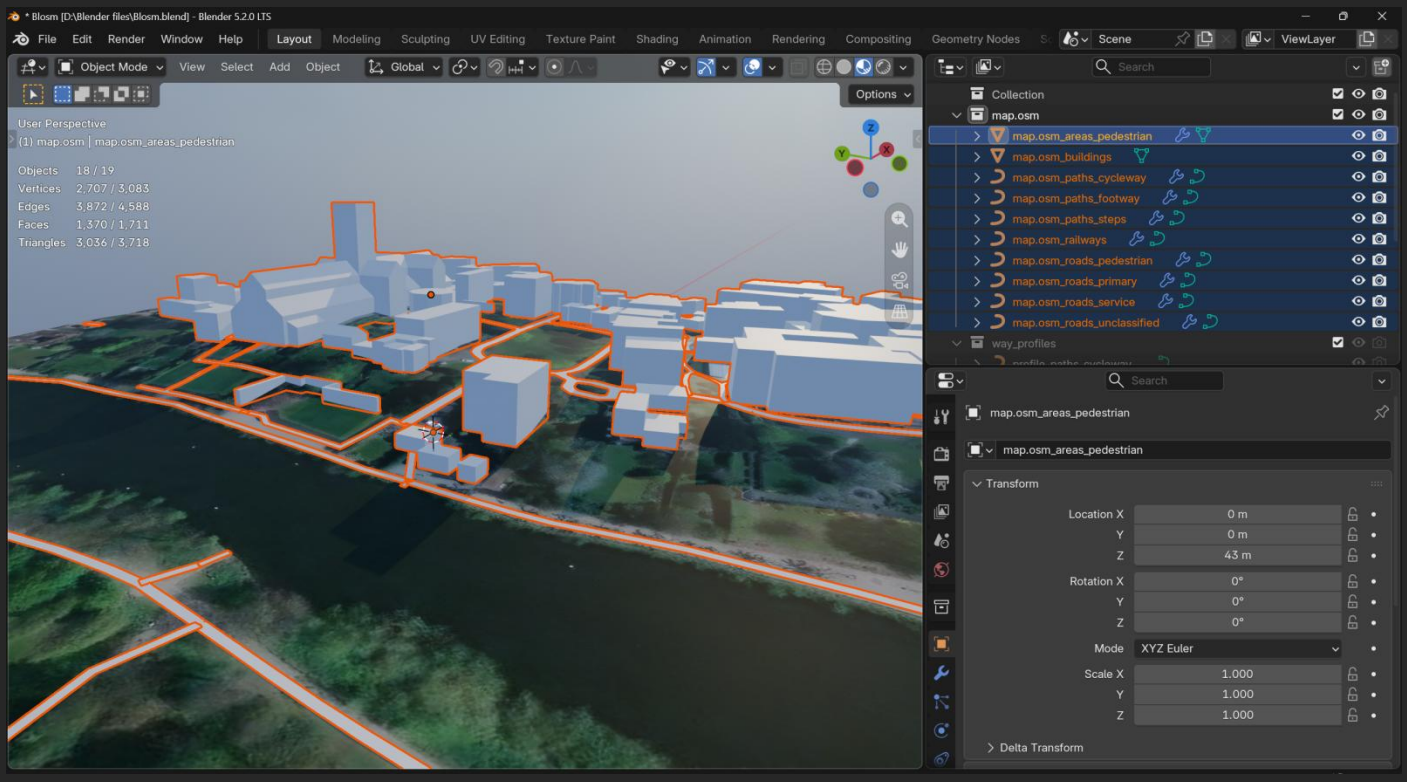
We've now got a quick + dirty 3D representation of our high-detail LOD0 area for later refinement. Let's get it into a format that Unity can understand.

(v) Export.

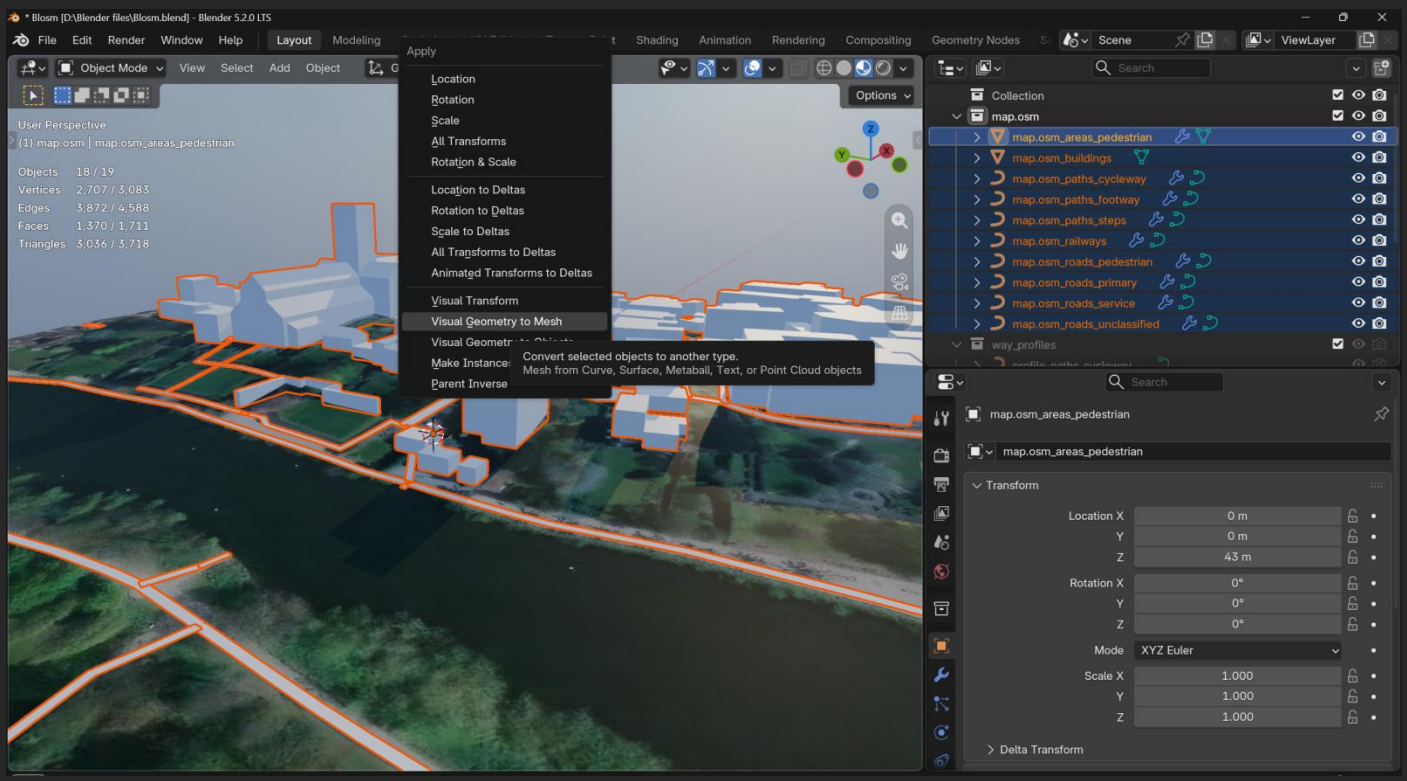
Create a folder called 'myLOD0_OBJ'. This folder will hold the files that represent your models.

But the model isn't ready for export yet… it's in pieces. To put it together :

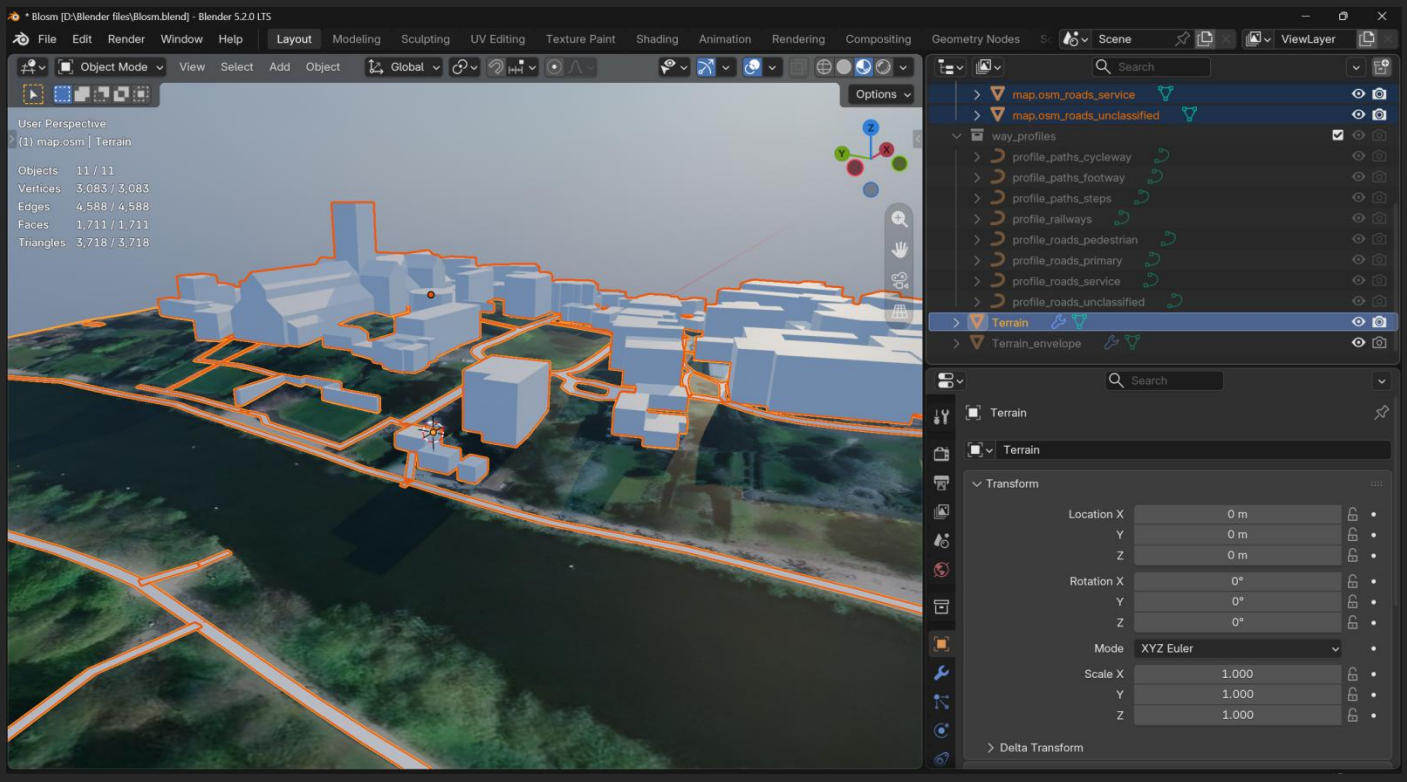
i. Select all the entries under 'map.osm' in Outliner.



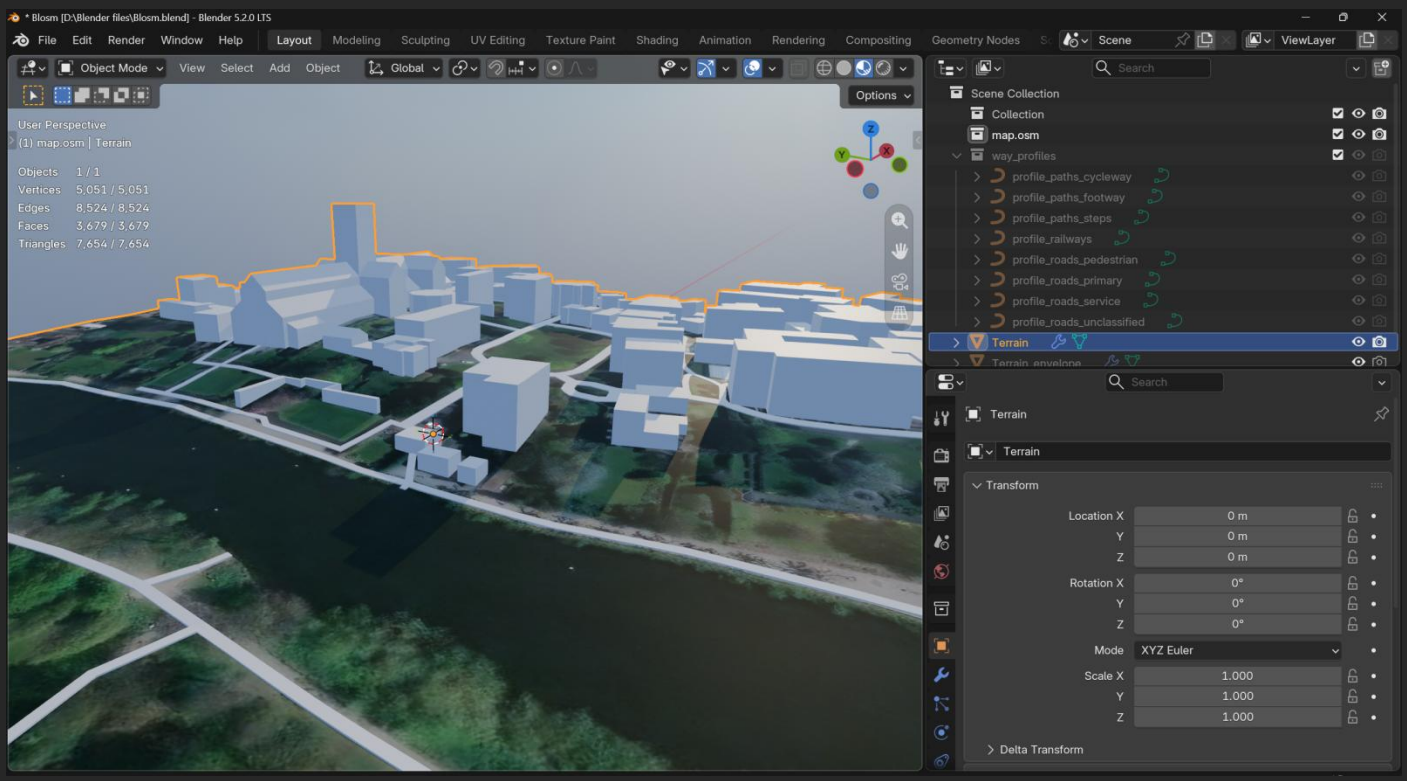
ii. Move the mouse into the 3D Viewport -> CTRL + A -> Visual Geometry to Mesh.



iii. With the ‘map.osm’ entries still selected, CTRL + left-click ‘Terrain’ in Outliner to add it to the selection.



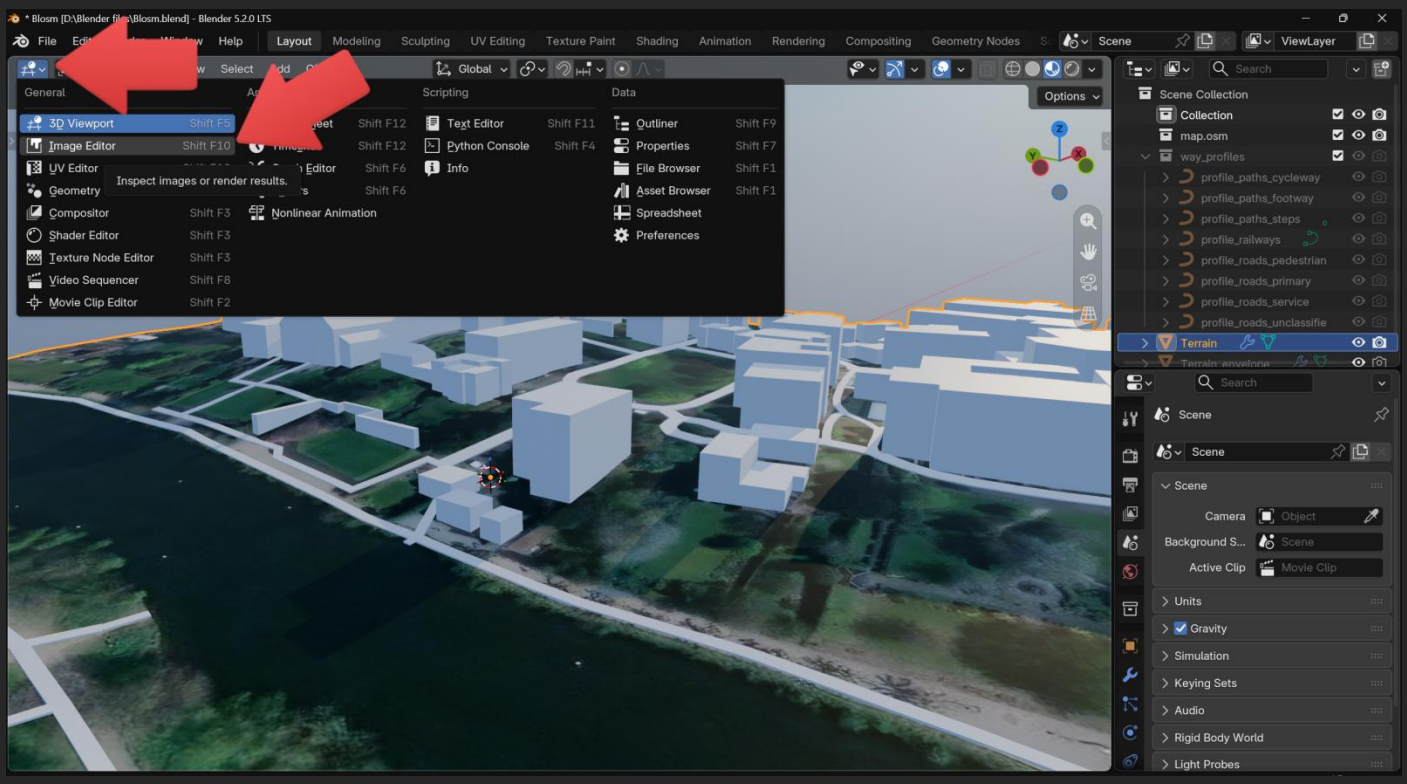
iv. With the mouse still in the 3D Viewport -> CTRL + J to join the selected meshes.



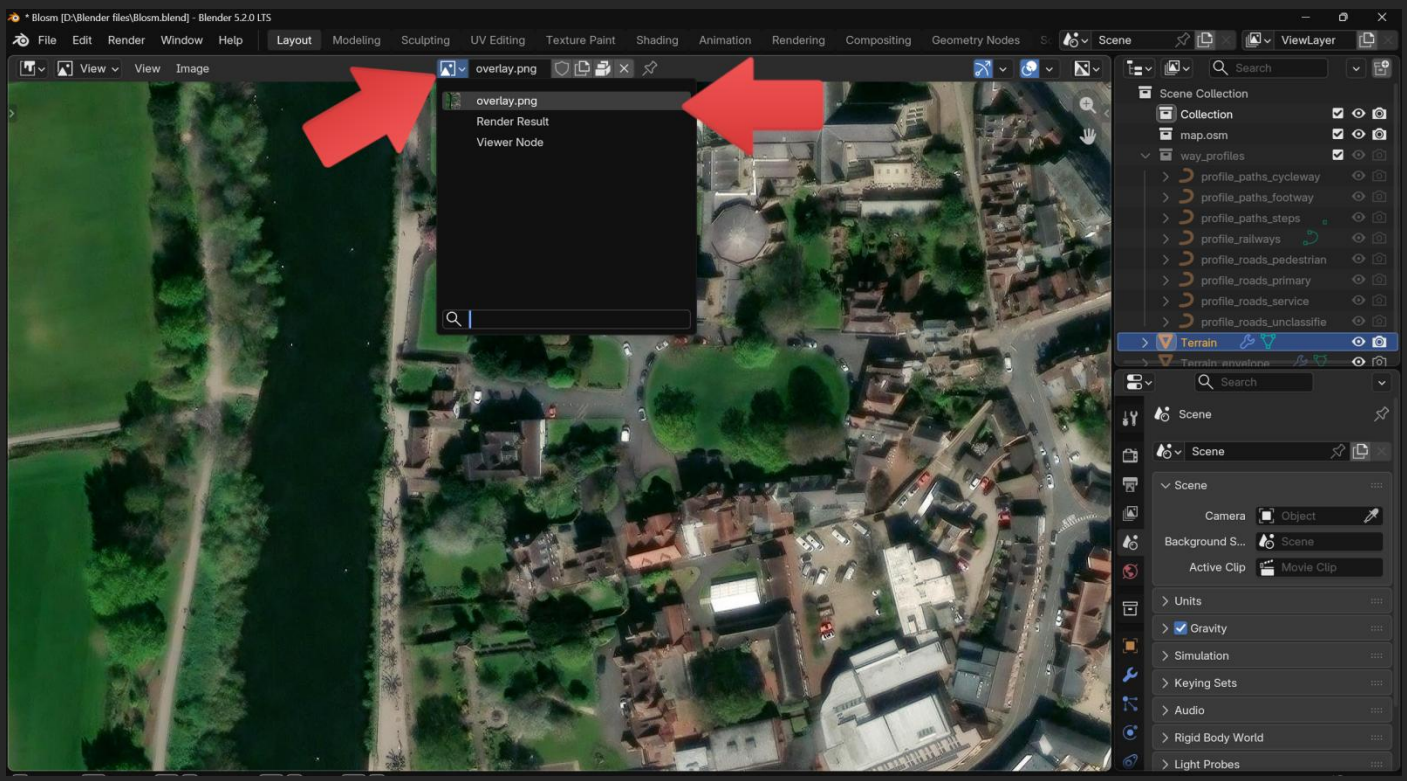
The model is now represented by a single ‘Terrain’ entry in Outliner.

We need to clean it up a little :

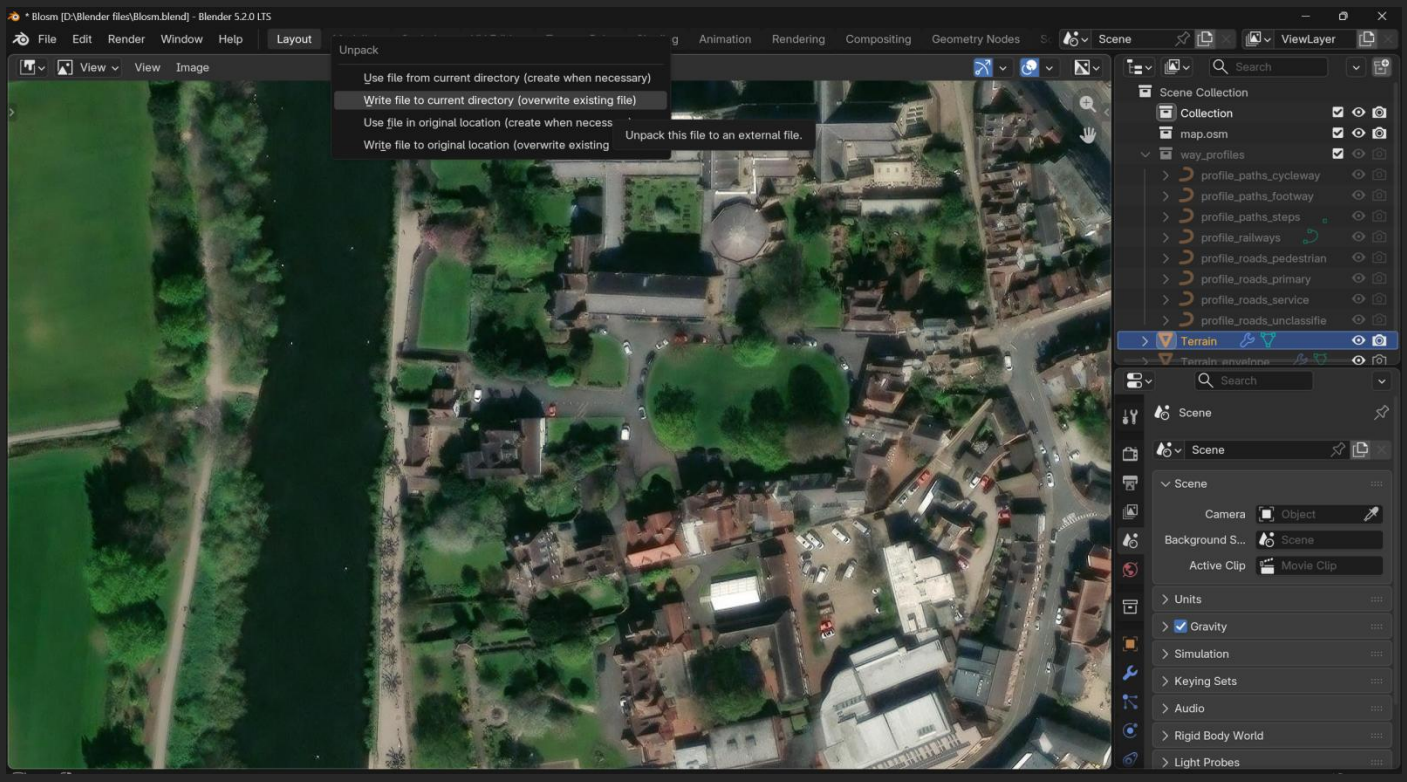
i. Change the main window from 3D Viewport -> Image Editor.



ii. Select the overlay.png

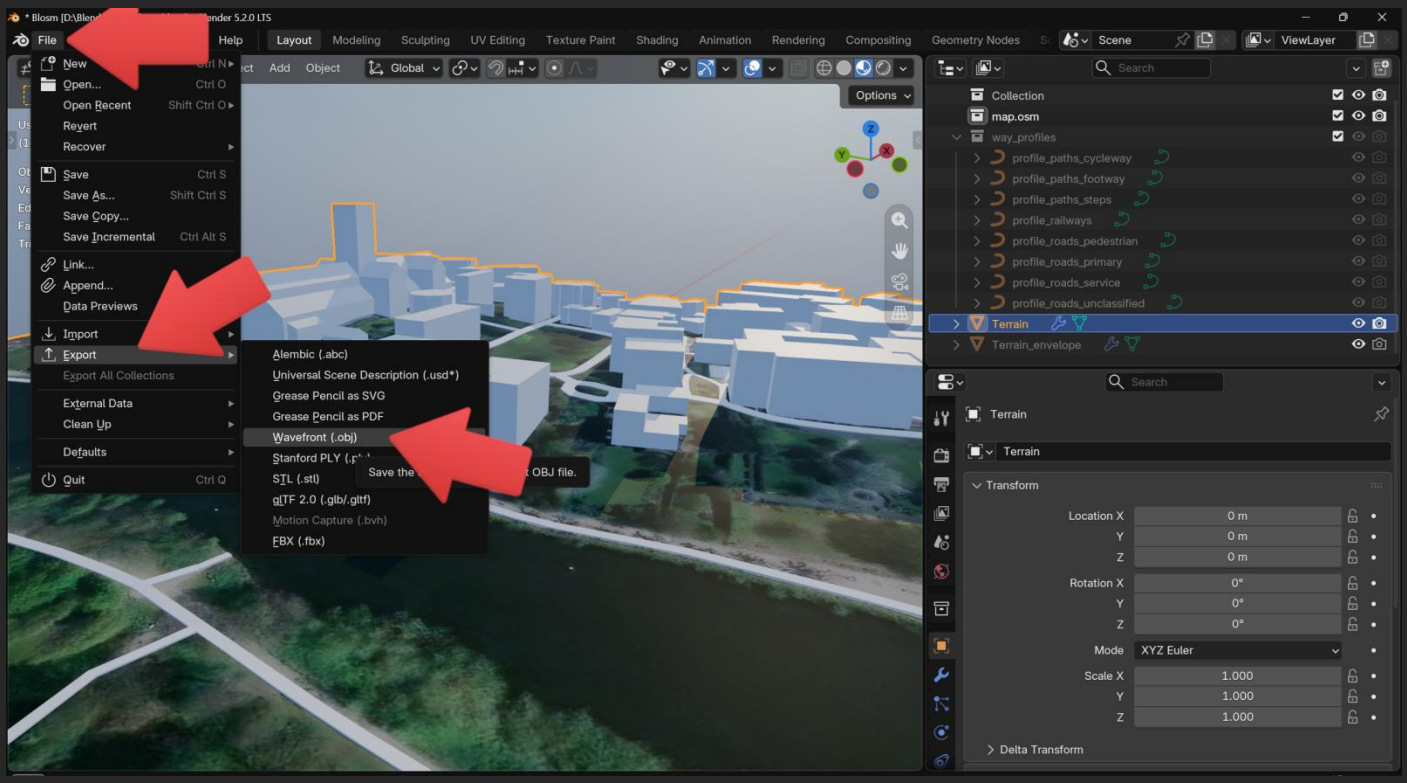


iii. Click Unpack -> Write file to current directory (overwrite existing file).

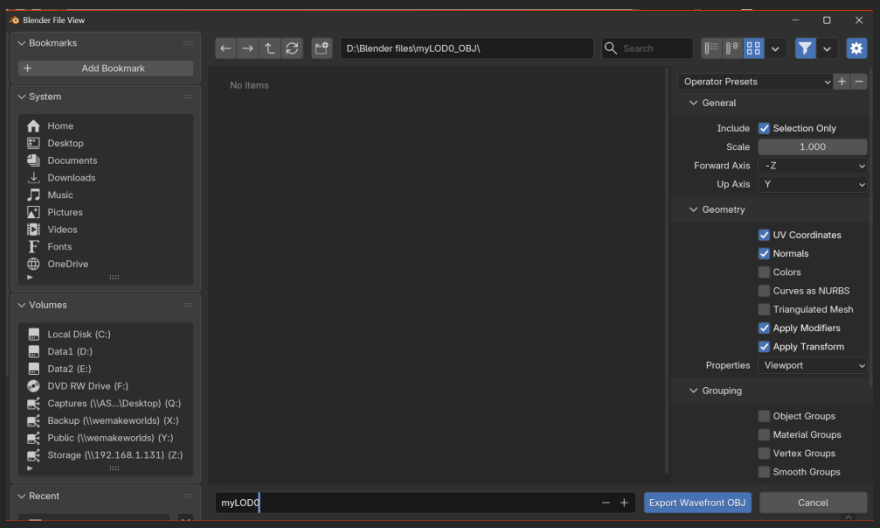


That's taken care of the ground texture. Change the main window back to 3D Viewport.

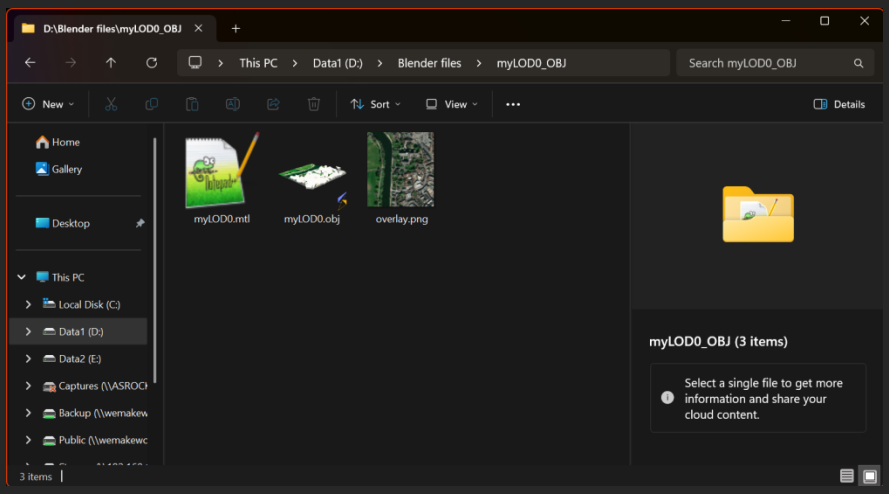
Ready to export! With 'Terrain' selected -> File -> Export -> Wavefront (.obj).



Blender File View shows the settings for .obj export on the right-hand side. Tick the box for 'Selection Only' + change the Path Mode to Copy. In the bottom text box, title the output file 'myLOD0' -> Export Wavefront OBJ.



Confirm that you have .obj, .mtl + .png files in myLOD0_OBJ :



Test the exported files match the original by importing myLOD0.obj into a fresh Blender project.

PART 2C : MAKE WORLDS WITH UNITY

(i) Install the Unity Editor.
(ii) Open the template project.
(iii) Welcome to the Unity.
(iv) Swap the LOD0 model.
(v) Unity settings
(vi) Build

Unity is our software development environment. It's where everything comes together. We can deploy the same project to 27 different devices with Unity, including the Meta Quest headset.

This guide uses a template map (see below), a complete Unity project that has much of the mechanics of an immersive map already provided. You are welcome to start a fresh Unity project + build from scratch, but using a template cuts development time and avoids reinventing the wheel. You can always look under the hood later. For now, let's concentrate on getting your models imported + built into a working immersive map.

The goal is to swap the LOD0_template model for the myLOD0 model that you made earlier.

(info box 1) In our fast-paced tech landscape, programs evolve. Twenty years after first release we have Unity 6, brimming with new ideas and features like the Building Block - pre-made chunks of components, scripts + assets (of which more later) to simplify specific jobs. There's also a new plug-in to handle the headset, a move that forks the immersive development path. With any new advance, the risk is that something will break. The fear is that a lot will break + suddenly you're fighting the engine rather than progressing towards the goal. Being on the bleeding-edge is dangerous.

(info box 2) We Make Malvern was written in a 2021 version of Unity + still holds up. It uses a deprecated Oculus Integration package + the soon-to-be-phased-out OculusXR plug-in, but 90% still works.

This guide uses a recent LTS (long-term support) version of Unity 6 so that we can play with the Building Block toys later, but resists embracing the new OpenXR plug-in until properly documented + tested.

	Unity version	XR Plug-in	Integration	Building Blocks
We Make Malvern	2021.3.45f2	OculusXR	Oculus SDK	No
Template	6000.3.19f1	OculusXR	Meta SDK	Yes

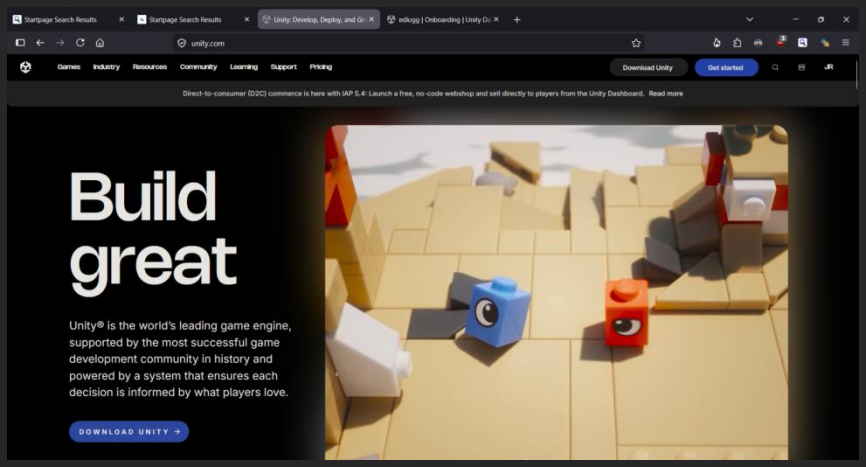
(info box 3) Whilst Blender uses the y-axis for forwards/backwards, Unity uses the z-axis. Blender is said to be y-forward; Unity is z-forward. Luckily, Blender has an option to export models as z-forward.

(info box 4) Integration.

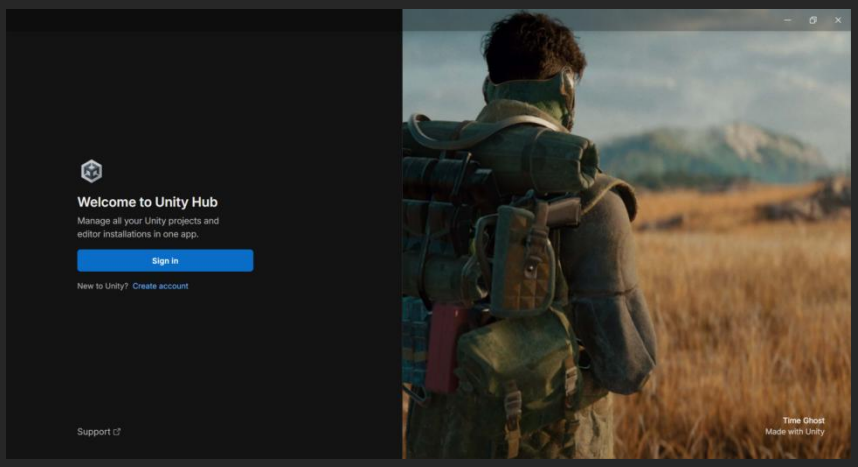


(i) Install the Unity Editor.

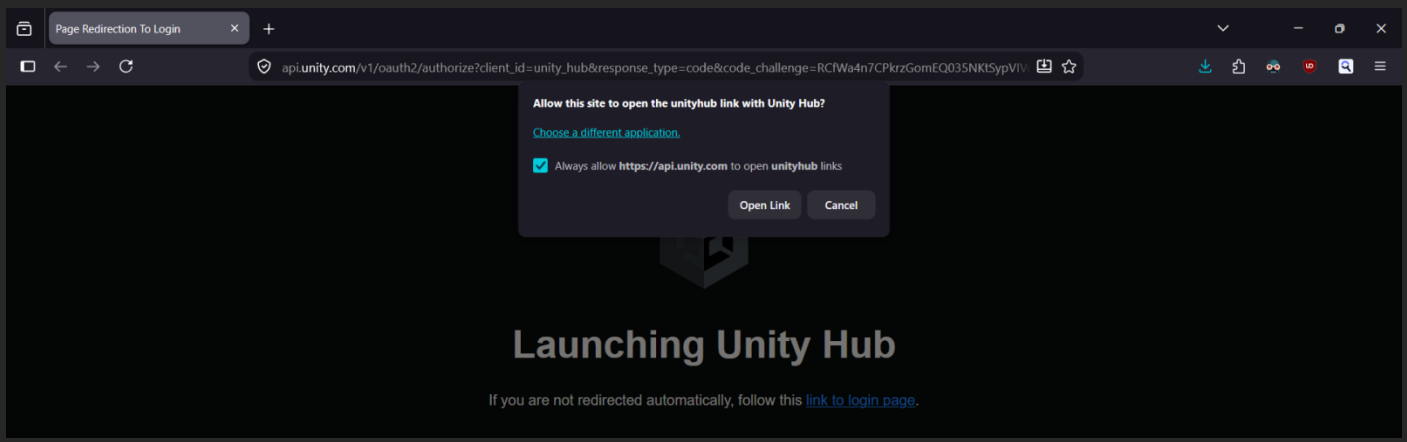
Go to unity.com and click 'Download Unity'. Create an account or sign in, then choose the Windows x64 package to download Unity Hub (if it didn't automatically start). This is the program that will actually install the Unity Editor and take care of licensing.



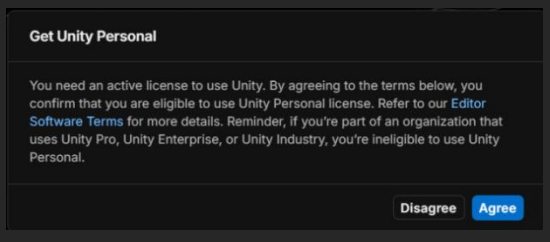
If Unity Hub fails to open, you may need to install Microsoft Visual C++ Redistributable 2015-2019[25].



After account creation + sign-in, you may have to associate the Unity API (Application Program Interface) with Unity Hub links. Check the tickbox and click Open Link.



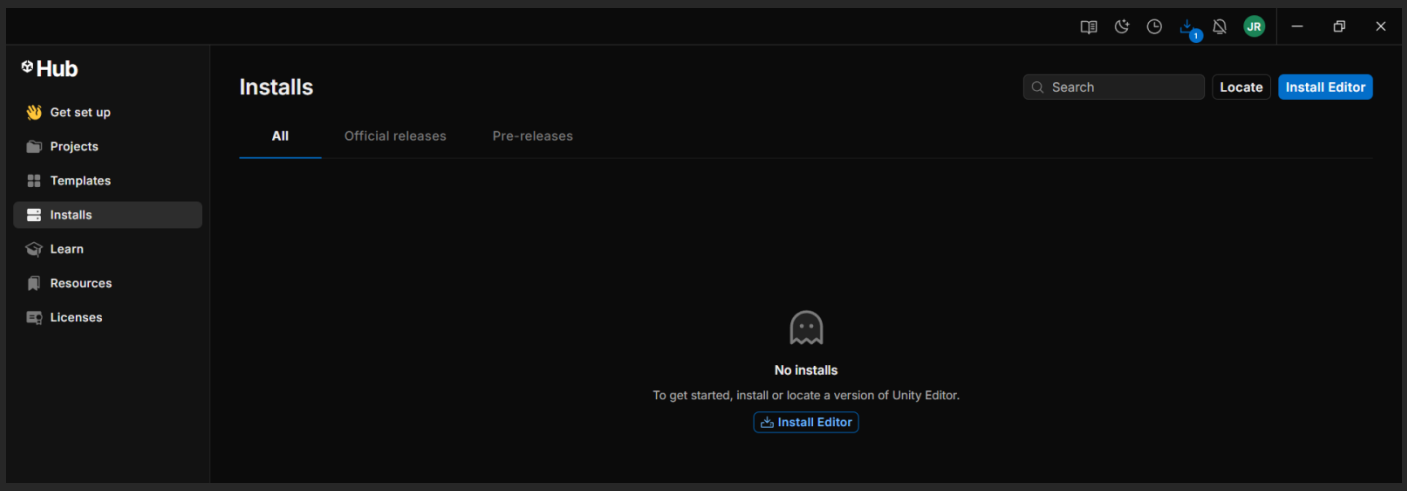
Agree to the free Unity Personal licensing.



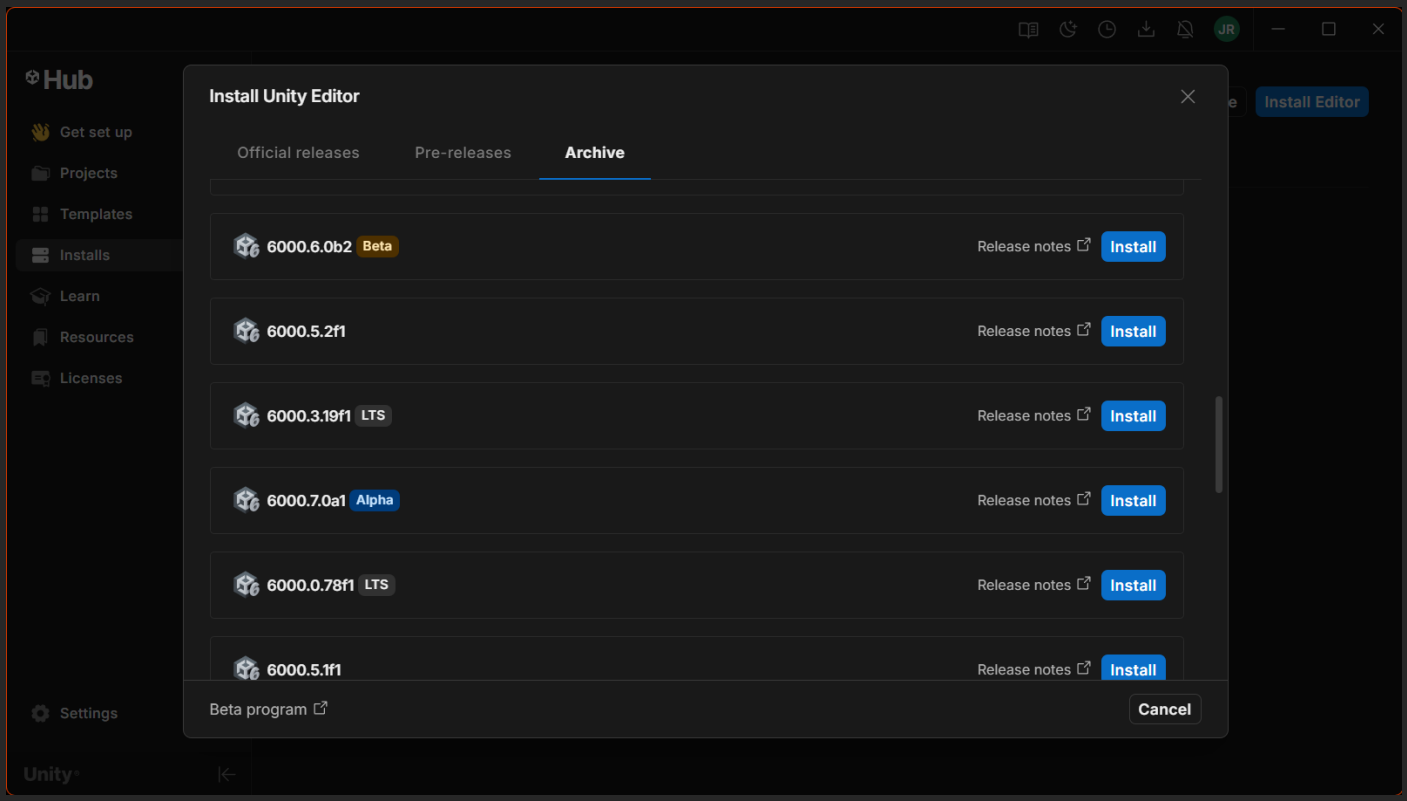
[25] tinyurl.com/2s3fw37p

NB. When Unity Hub opens for the first time, it cheekily starts downloading the latest Unity Editor. This isn't the version we want, so cancel it as soon as you can. It doesn't matter if it installed.

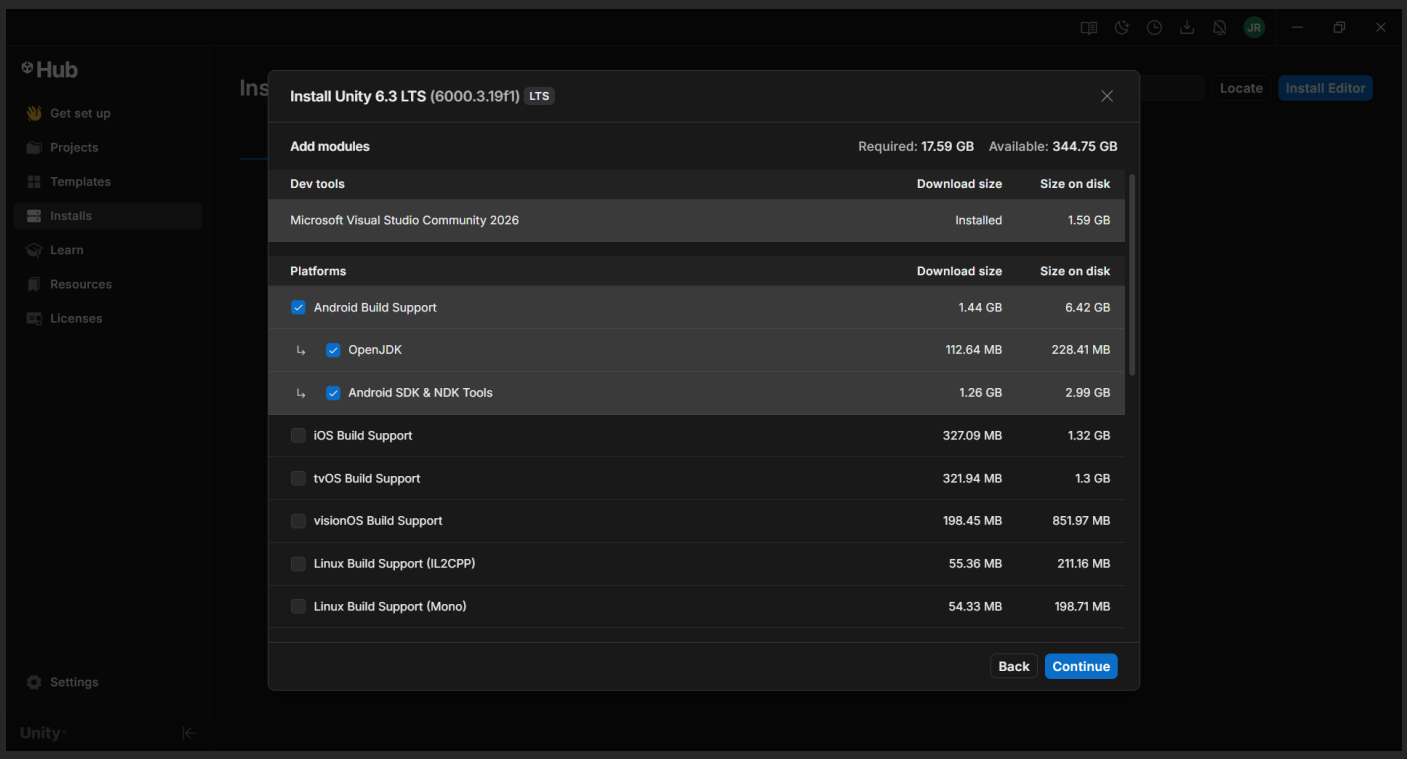
Select 'Installs' from the left-hand menu.



Select Install Editor -> Archive -> Unity 6000.3.19f1 LTS -> Install.

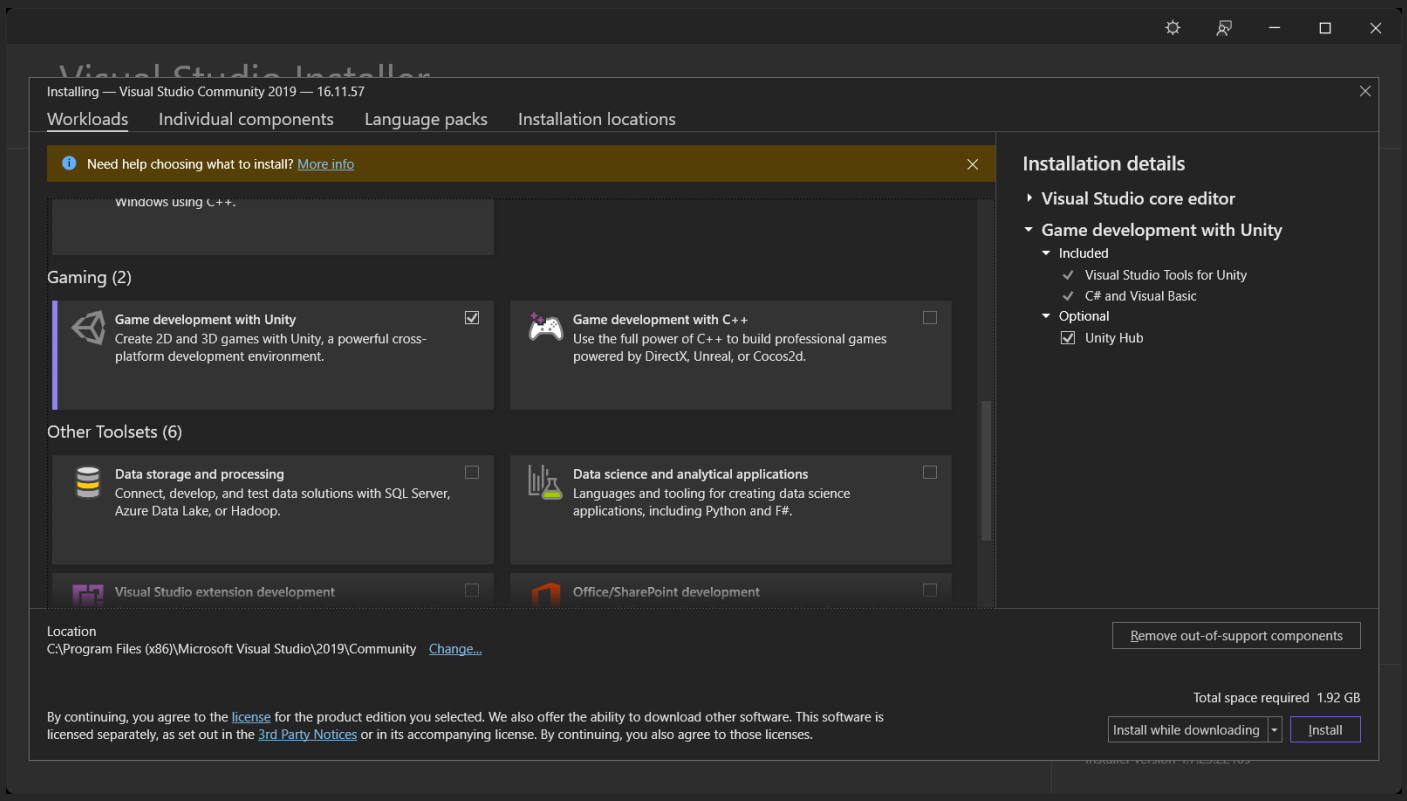


Include 'Android Build Support' from the 'Add modules' screen to support OpenJDK and Android SDK + NDK.



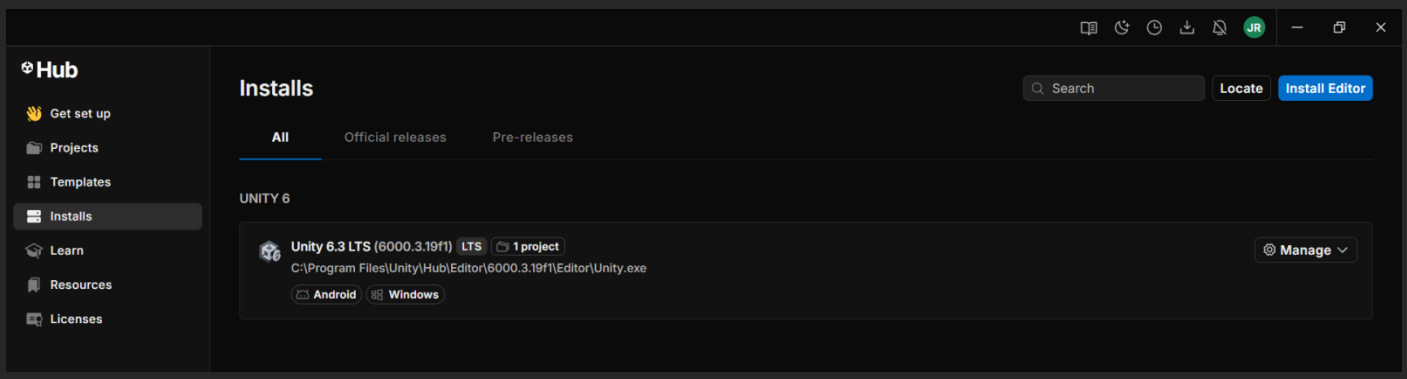
Accept the terms and wait for the many downloads + installs to complete. Unity may require user-input for User Account Control (UAC) progression + will fail otherwise.

When Visual Studio is installing, it will ask for modules to be added. Check ‘Game development with Unity’ + click Install. When Visual Studio has finished installing, exit the installer.



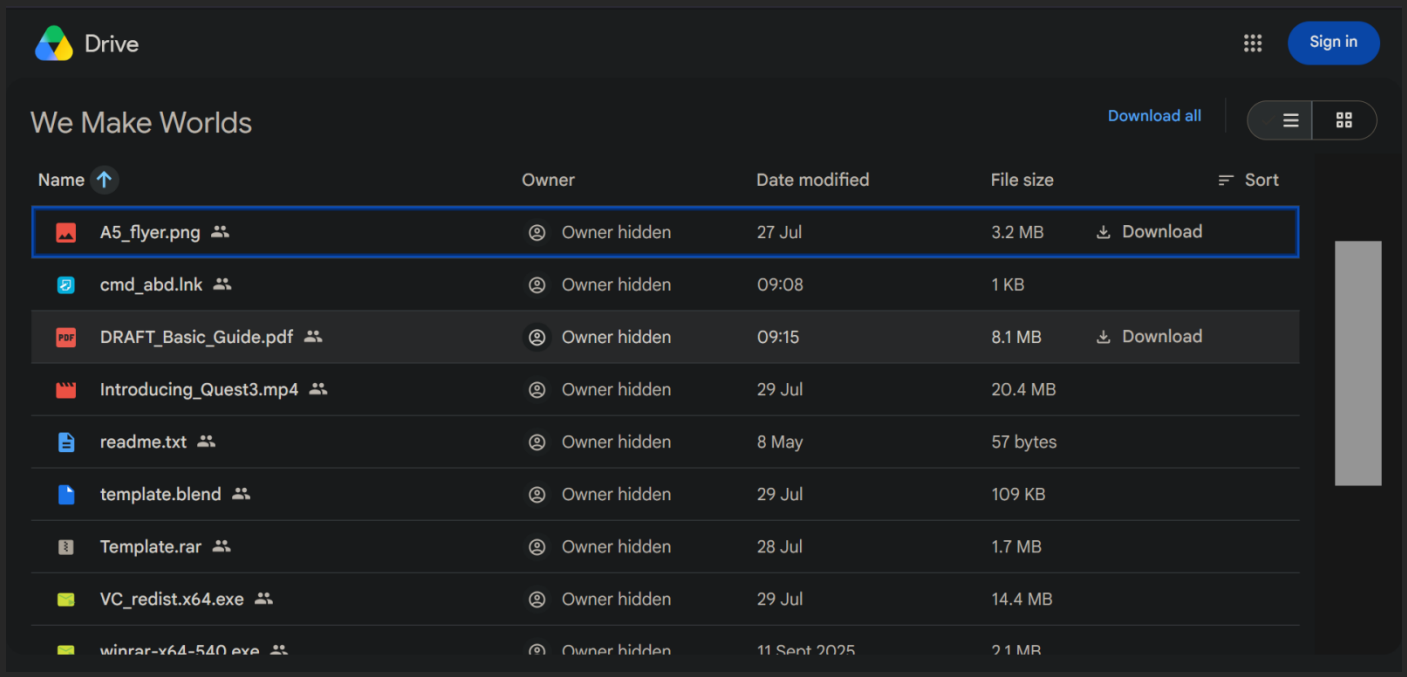
Exit the Visual Studio installer manually when finished to continue.

You can now check that the correct Unity Editor has been successfully installed at Unity Hub -> Installs.

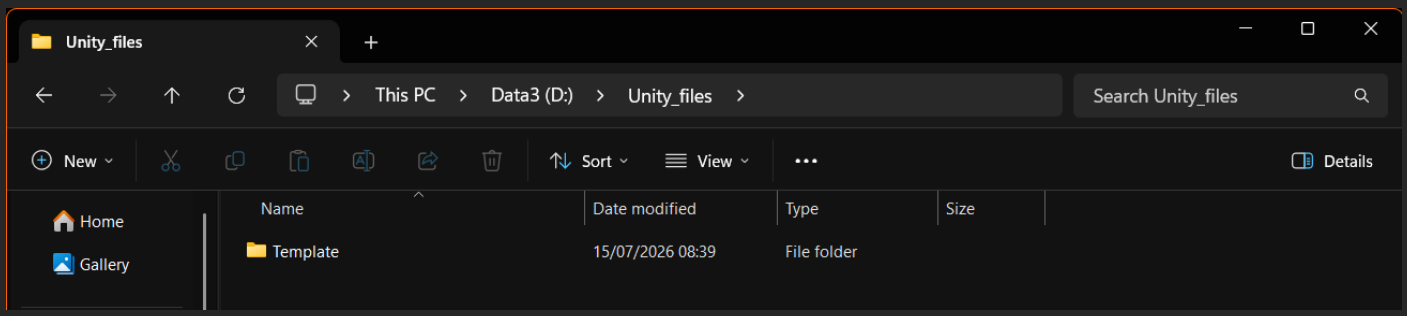


(ii) Open the template project.

Grab the compressed template project[26] 'Template.rar'.

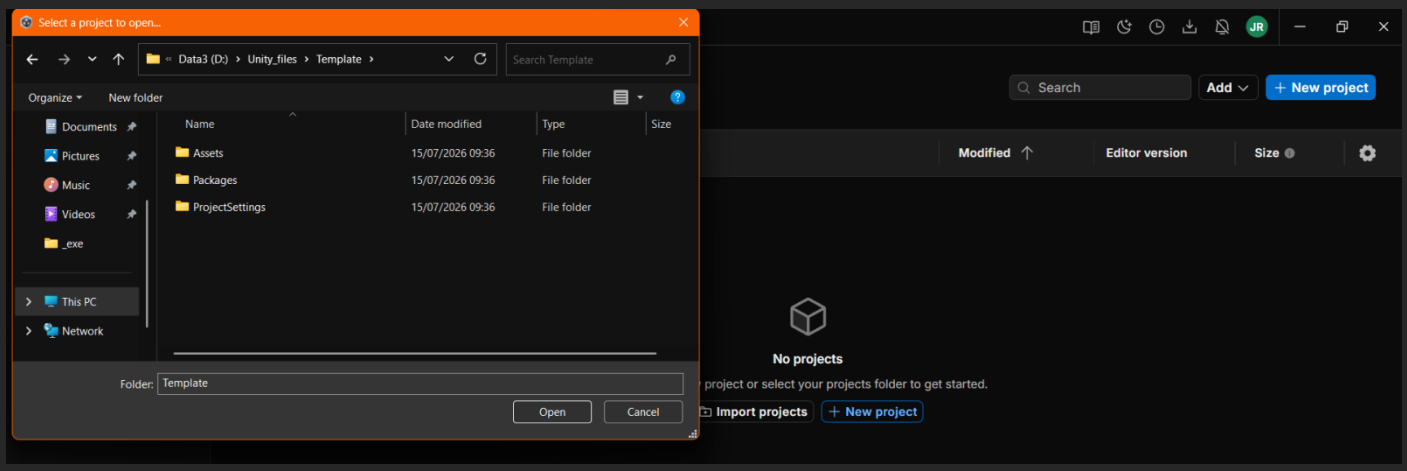


Unpack the Template.rar file to a destination of your choice. I use a folder called 'Unity_files'. NB. If your Template.rar icon is blank, you'll need to install a program to handle .rar files (try winrar).

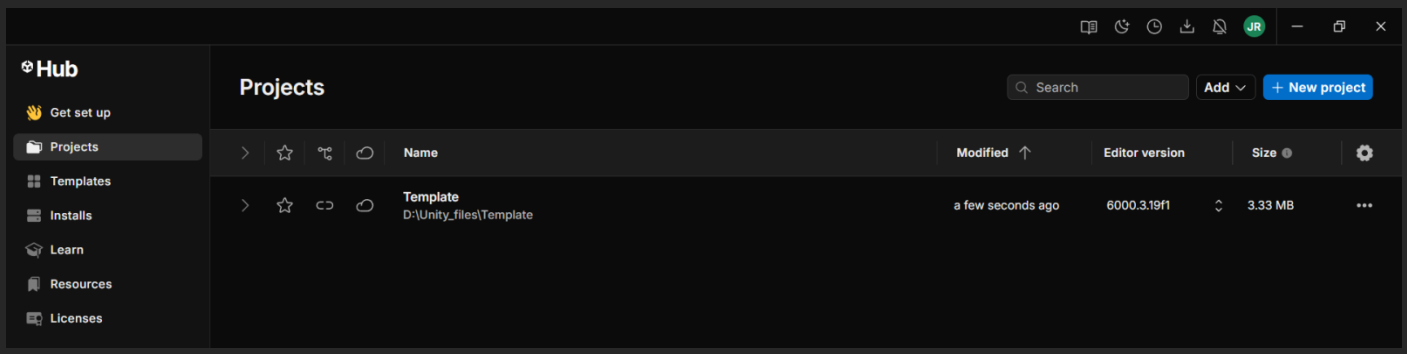


Point UnityHub to the new project to open it with Add -> Project from disk -> [directory]. You need to descend into the 'Template' directory before clicking 'Open'.

[26] tinyurl.com/2s3fw37p



You should now be able to see the template in UnityHub -> Projects. Single-click to open it in Unity.

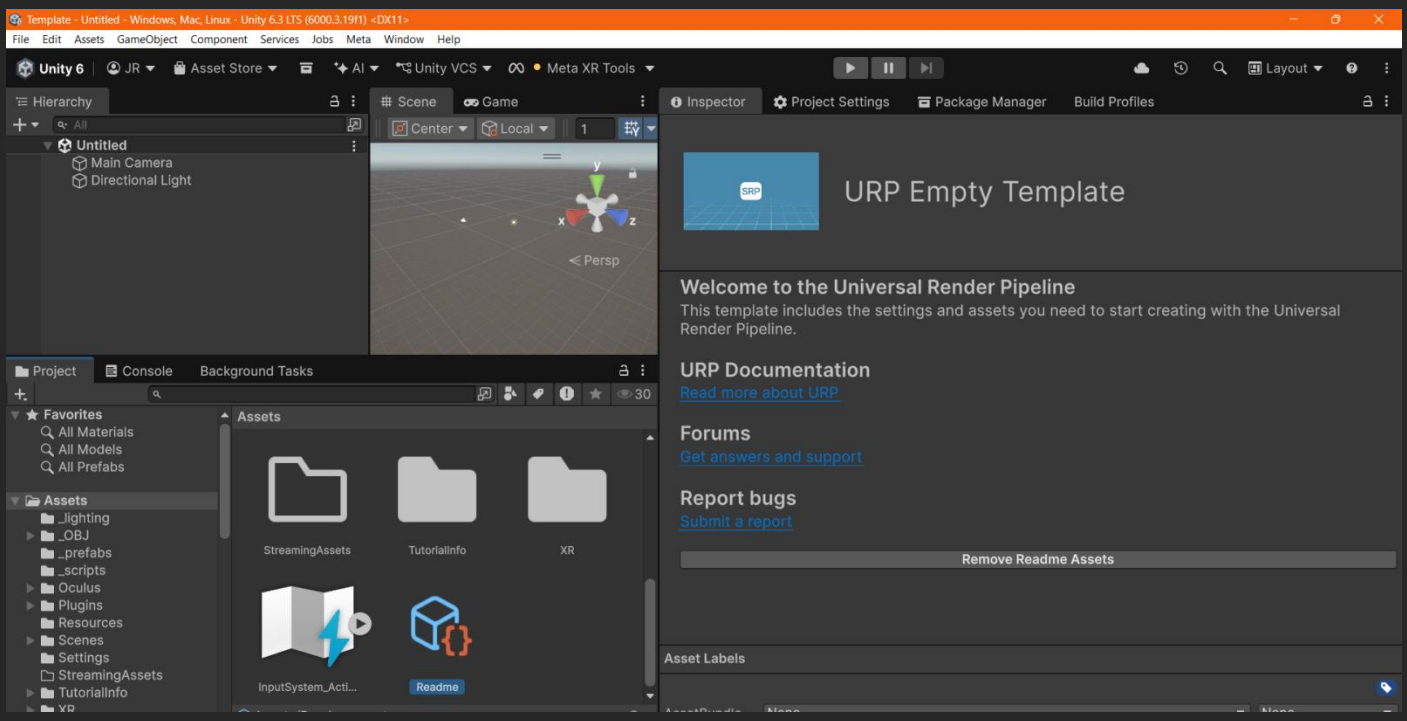


Single-click the Template entry + agree to the Licensing Terms.

The first time Unity starts takes a long time as it reconstructs the template project on your computer. Subsequent starts will not take as long.

(iii) Welcome to Unity.

Unity uses the concept of an empty GameObject as a container for our assets. *Components attached to these elements work together to create the computer-generated display inside the headset that allows us to experience the digital world*.



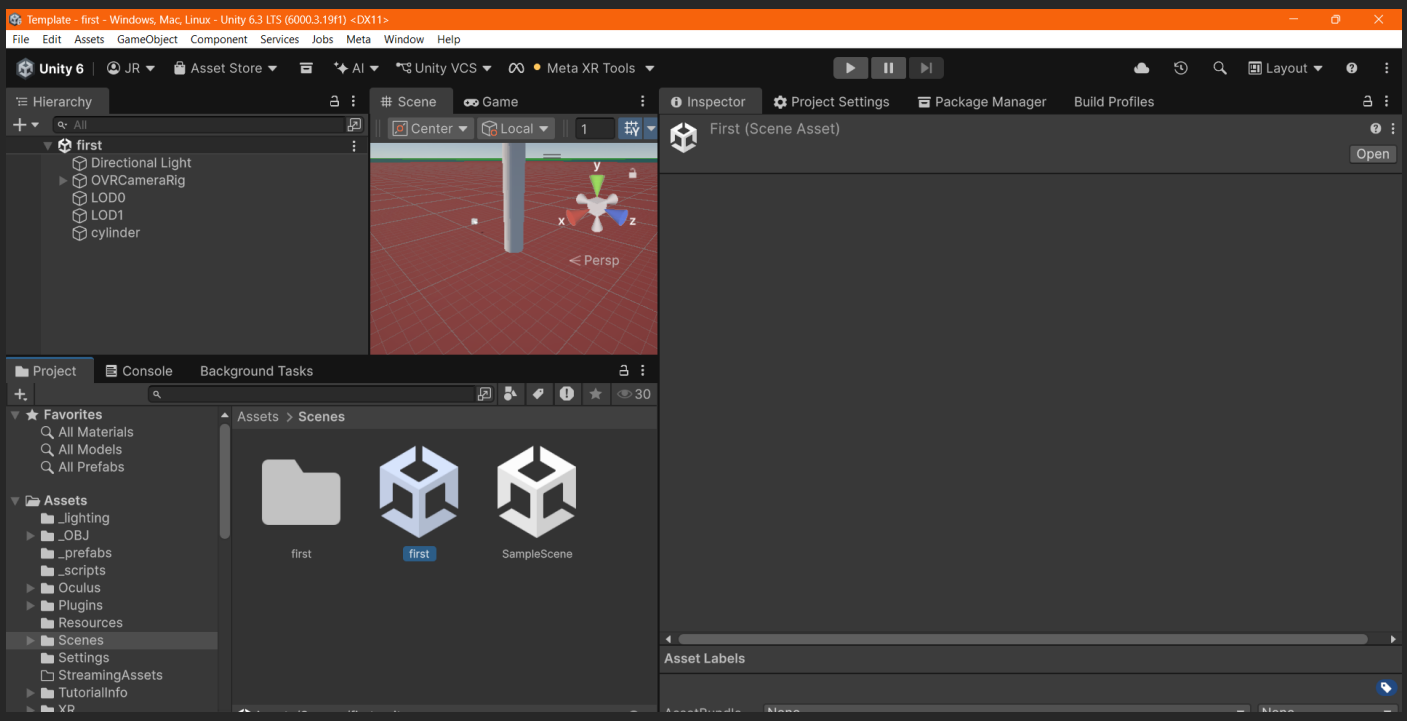
There's five main parts to the Unity interface :

Hierarchy	GameObjects in the project
Project	All available assets that can be attached to GameObjects
Assets	Models, scripts, etc
Inspector	Information on GameObjects
Scene	A 2D representation of the 3D view from the camera

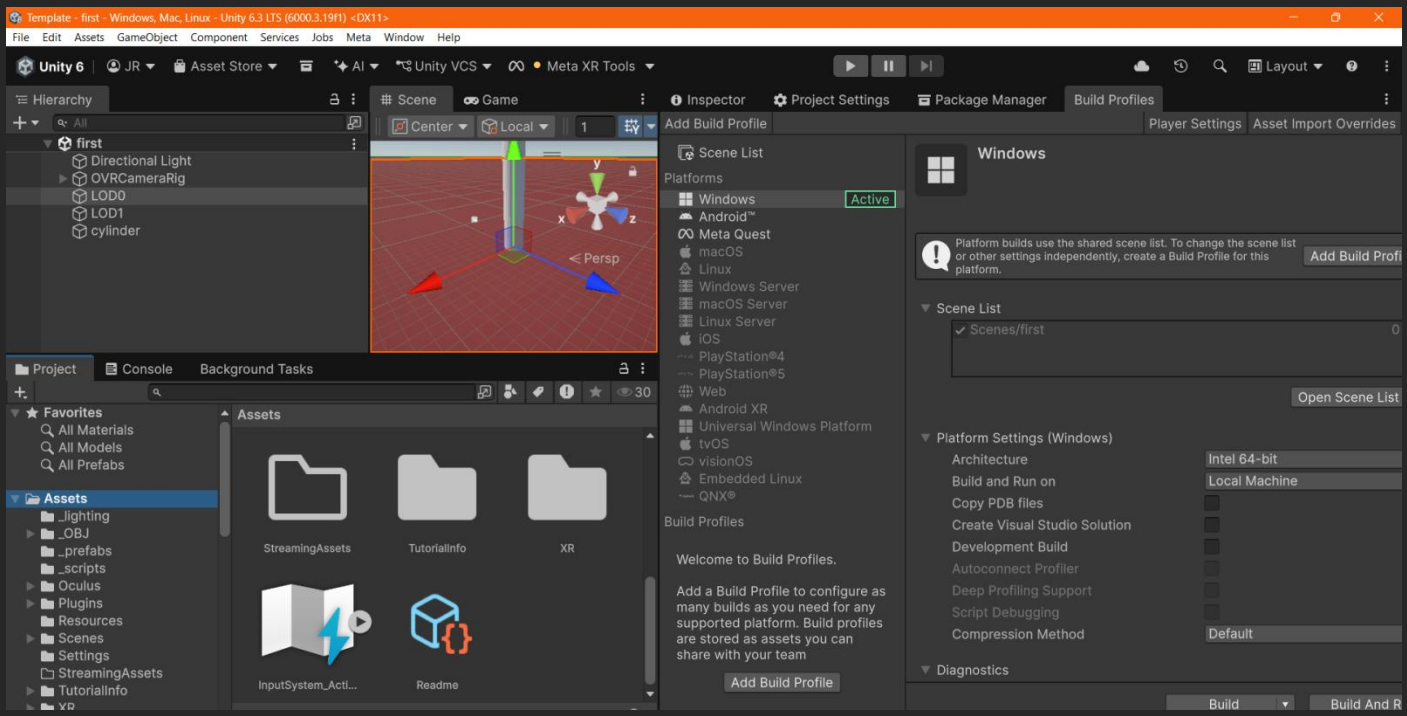
There are dozens of others, but we won't need them. 80/20 rule : 80% of the time we'll only use 20% of the functionality. Resist becoming overwhelmed by the shock of seeing the complexity of the Unity. Nothing is on the screen without a good reason.

Resize windows to suit using two-headed horizontal/vertical arrows that appear when hovering over window borders. Get really good at this, as information is often hidden in a window that’s too small.

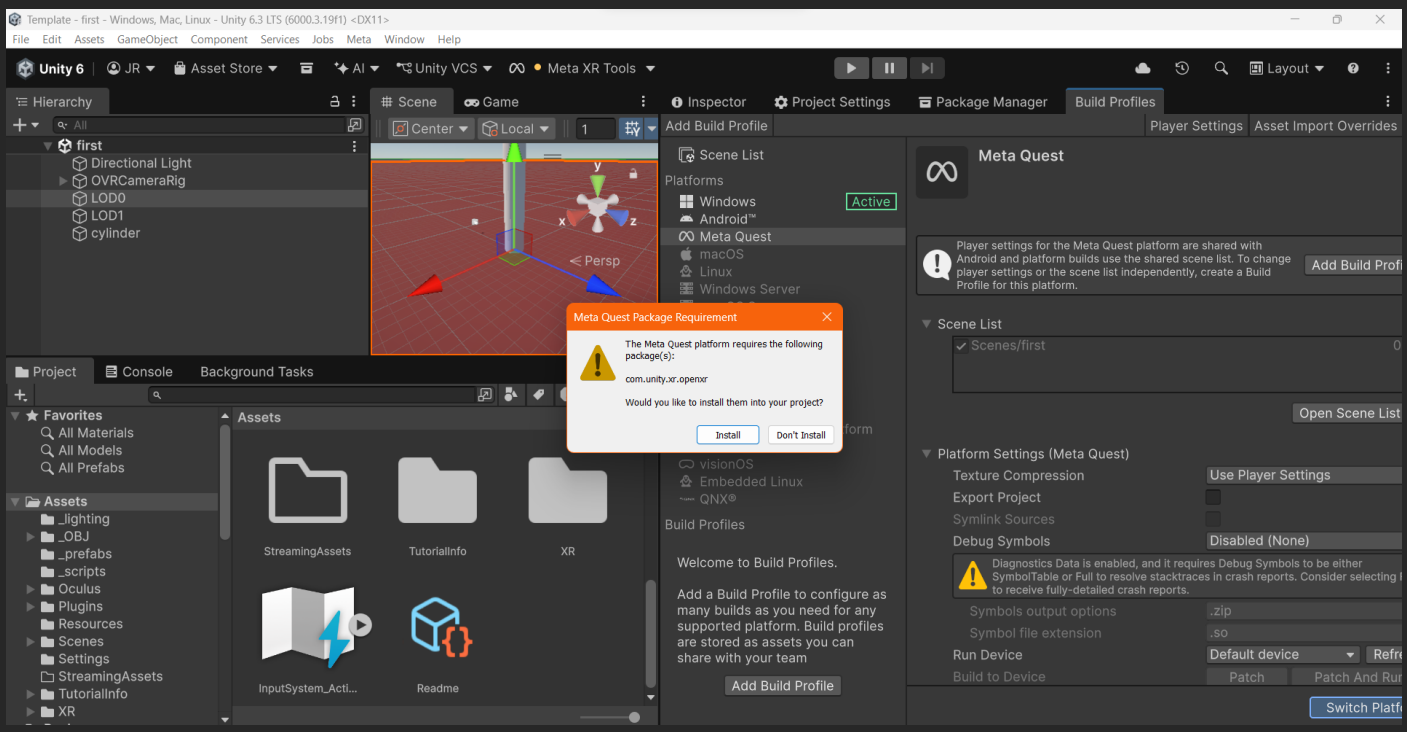
The template comes with a pre-built scene that has simplified LOD0 + LOD1 models. Open it by double-clicking Project -> Scenes -> first (the one with the Unity logo).



Change the Build platform from Windows to Meta Quest with Build Profiles -> Platforms -> Meta Quest -> Switch Profile.



When offered the openxr package, decline with ‘Don’t install’.



Wait for Unity to compile (indicated by a spinning icon in the bottom-left corner changing to a tick in a circle).

(iv) Test the template map.

With your headset connected to the PC, press Play (top-middle) + put on the headset to confirm that the template map works.

(v) Swap the LOD0 model.

We can now import the model you made earlier to replace the model that came with the template.

1---> delete LOD0 template from Hierarchy.

2---> drag'n'drop **your** myLOD0_OBJ folder that you made in Part 2B into the **_OBJ** folder in Assets.

3---> drag'n'drop Assets -> _OBJ -> myLOD0_OBJ -> myLOD0.obj into the Hierarchy. Reset it's Transform position to (0,0,0) in Inspector.

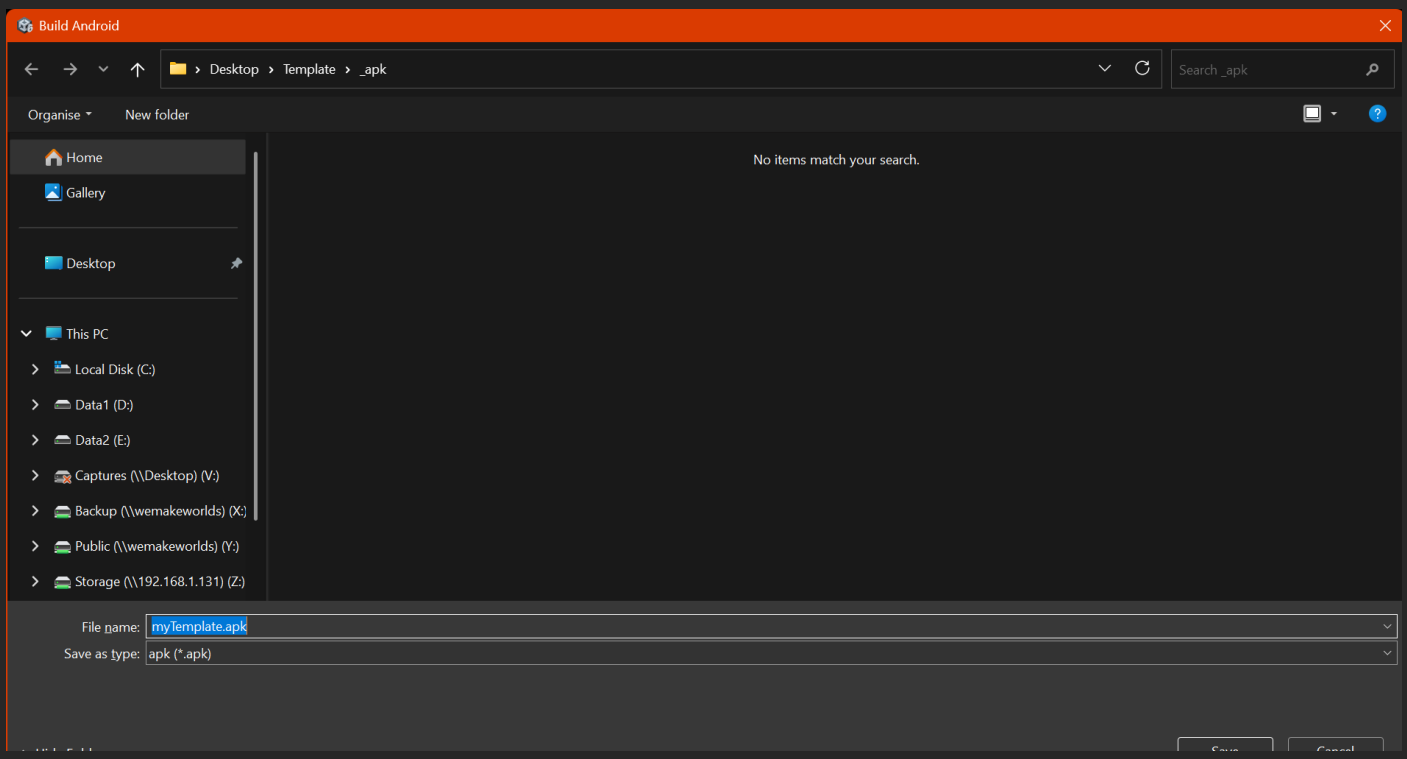
Save your scene with File -> Save and save your project with File -> Save Project.

(vi) Build.

The object for Unity is to make + output a .apk (Application Package, AKA software), known as a build, which contains the compiled source-code of the program + a manifest file. It's everything the headset needs to display a world. The Meta Developer docs have detailed instructions on how builds can be transferred to the headset.

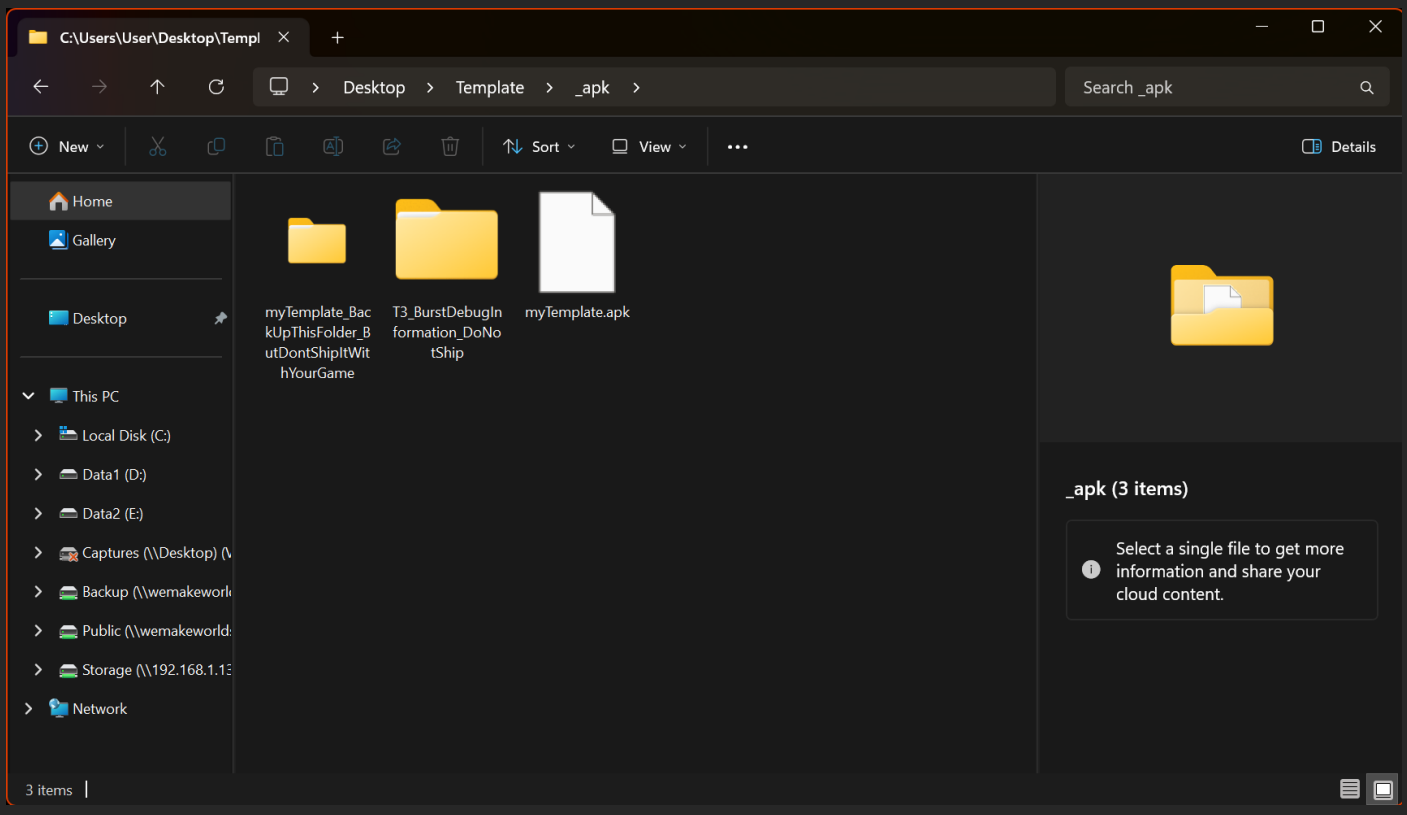
a. Build Profiles -> Build.

b. Create a new folder '_apk' + name your project 'myTemplate'.



Wait for Unity to build the package. This may take some time.

When finished, Unity will show the output file in File Explorer.



We use adb (Android Development Bridge) to push a build directly to the headset for testing. It's included in the Unity Editor installation.

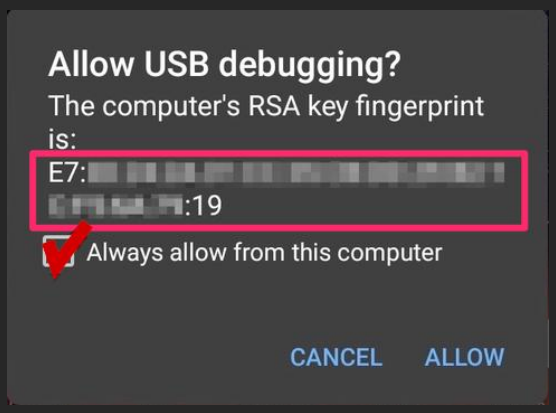
c. Attach the headset to the computer + turn it on. If this is your first time connecting, confirm the key fingerprint displayed inside the headset matches that of your computer's corresponding public key, found on your computer at :

C:\Users\[Username]\.android\adbkey.pub

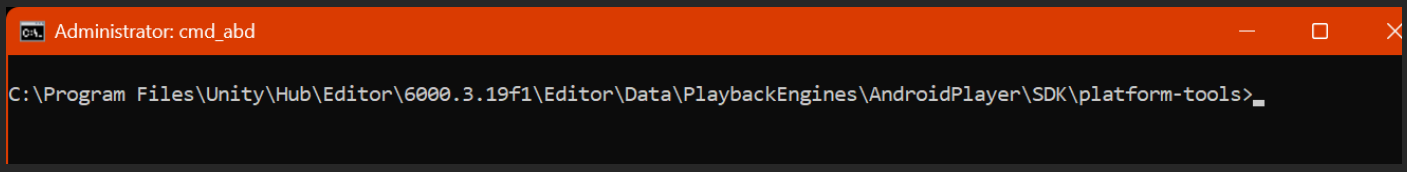
To extract the fingerprint, open PowerShell[27] + enter :

certutil C:\Users\[Username]\.android\adbkey.pub MD5

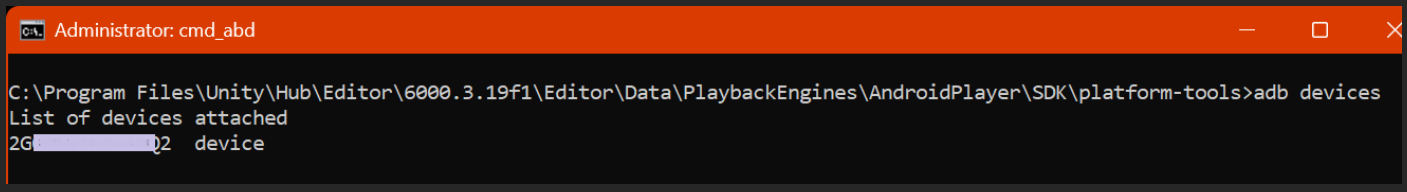
Confirm matching fingerprints, then tick the checkbox for 'Always allow' + click 'Allow'.



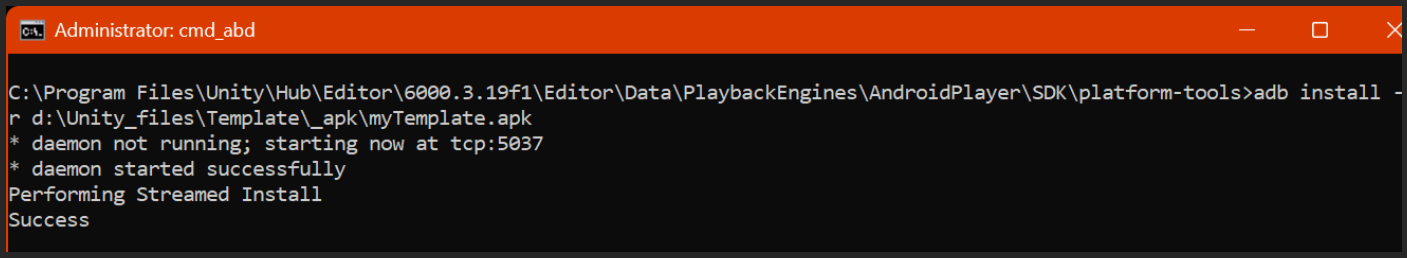
d. Download + run the cmd_adb shortcut[28]. It's pre-configured to start where the adb executable is located.



e. Check the computer can see the Quest headset with 'adb devices'. The serial number of the headset is shown, together with a status code. 'device' means that adb has established communication with the headset.



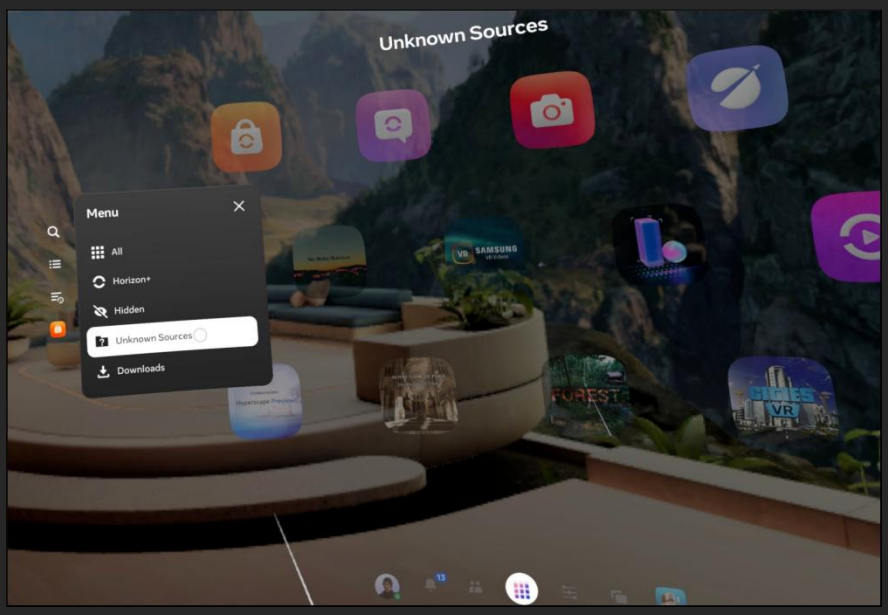
f. Push the .apk to the headset with 'adb install -r [path to].apk'.



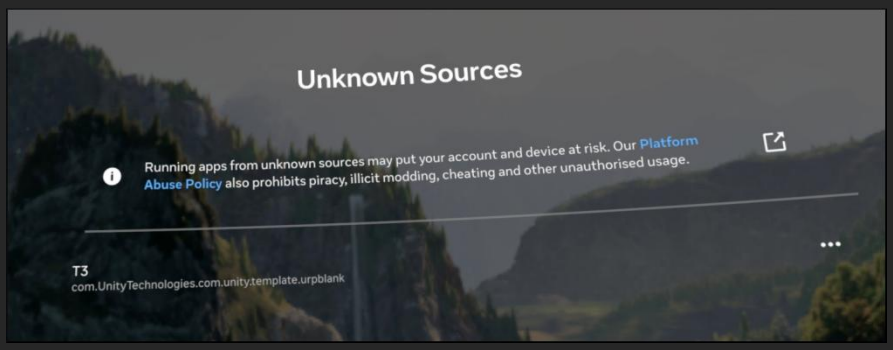
(Question : what's the –r flag for?) A : overwrite existing file.
(Question : what's a daemon?) A : a program that runs in the background.

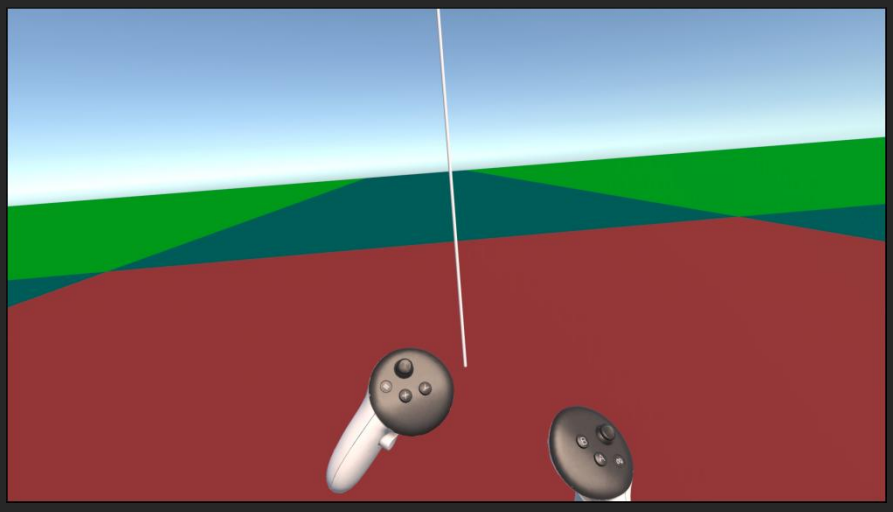
Unplug the headset + put in on to find your map. It's only visible under Unknown Sources in your Library Menu.

[27] In the Start menu, search for PowerShell -> Run as Administrator
[28] tinyurl.com/2s3fw37p

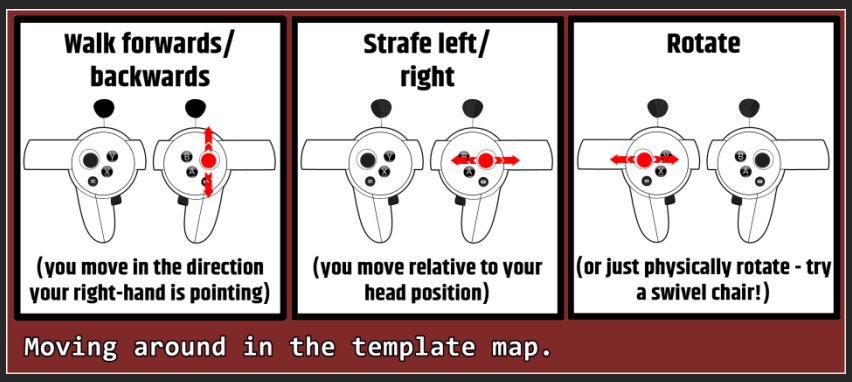


The title is ‘T3’. Click to start.





Congratulations! You’ve successfully made a proof-of-concept immersive map.



Now grab a pencil + paper… it's time to find + collect data about your world.

PART 3 : GET REAL-WORLD DATA

The data you need is dictated by decisions made in Part 1. From Part 2, you have experience of how that data is used by the Blender -> Unity -> Quest workflow. The trick is to identify the *right* data for *your* map.

The quality of this data determines the sense of presence that your map can achieve. Any digital information can be part of an immersive map.

(info box 0) For spatial data, environments + objects in the centre of the map (LOD0, the high-detail part) will benefit from millimetre accuracy + high-quality photography, whilst distant objects (LOD1, the low-detail landscape) do not. *Be as accurate as you need to be.*

(info box 1) Always ensure you are safe from traffic + hazard.

(i) Some data-gathering ideas to get started :

Start at your chosen latitude/longitude. This will be on or very near the starting point for visitors. Aim for a good view or vantage point with something to orient the visitor.

With tape measure + trundle wheel, define the position of the corners of your LOD0 high-detail area + the boundary that connects them. Keep within the maximum limit of 1000m^2.

Measure the dimensions of rooms, contents, objects, fixtures + fittings.

Take extensive LOD0 reference photography of everything from many angles. Don't rely on memory to remember what things look like.

Estimate the height of trees, rooftops + other tall objects. Pythagoras can help[29].

Use photogrammetry software to leverage your GPU[30] to create base 3D models from multiple 2D images. Your smartphone or tablet may have a LiDAR[31] scanner for 3D photogrammetry + reconstruction.

Capture 2D/360 video for moving objects + features.

Capture 360 photo's to incorporate photospheres into your map.

---

[29] tinyurl.com/3kbr8jb5
[30] Graphics Processing Unit : accelerates graphics rendering.
[31] Light Detection + Ranging : provides accurate measurements of objects + surfaces using lasers.

For interactive objects, think about the weight of the object. Record any special characteristics that you'd like to replicate such as glint.

Make drawings + sketches.

Find plans, maps + schematics. Root around your local library, community centre + the internet.

Grab free public datasets from the internet (eg. elevation data at 1m spatial resolution for all of England[32]).

Talk to the inhabitants + stakeholders of the map's area for local knowledge. Record conversations to replay in the map.

Record the ambient sound of a location to incorporate background audio.

Record traffic flow to model streets + roads. Present traffic-calming measures + identify safe play areas for kids.

Find position + depth of cables, pipes + underground channels.

Identify flora + fauna.

PART 4 : MAKE 3D MODELS WITH AI

Employ AI, either web-based (free or paid) or local (on your own computer), to generate 3D models from 2D pictures. This recent advance in computer technology is fast becoming a major boon for those to whom manual creation is not feasible. Beware, though… an AI-generated model will still need to be manually refined (reducing polygon count, filling holes, resizing etc.), so you'll still need to put some time into learning Blender.

(i) Web-based

There are a lot of web-based services that purport to create 3D models. The only one I can recommend is a paid-for service called Meshy (meshy.ai, limited free demo, pro version ~£15pm w/ usage cap). There are no free services that I can recommend.

Meshy takes 2D pictures, drawings and text and magically constructs a solid 3D mesh. It's quick and reliable, with a clean interface that allows for customization w/o overwhelming the user. The end results are impressive, albeit with a higher polygon count than manual creation. Automatic texturing works well, but does result in a large texture file. Models can be imported directly into Blender with a free bridge plugin.

(ii) Local

If you have the resources, local generation is the way to go. Not only can you produce models for no cost, but the wealth of customization options means you can refine output until it hits the sweet spot between file size + fidelity.

A typical workflow comprises an application to control the workflow plus a local engine to run the model components. The minimum system requirements are high :

CPU : i7+ (or equivalent)
GPU : Nvidia 4090+ w/ 12GB VRAM+ (or equivalent).
RAM : 24GB+

There are many guides on the internet for setting up such a system.

[32] tinyurl.com/6xjjmhvn

APPENDIX 1 : SOFTWARE DEVELOPMENT AGREEMENT (SDA)

Software Development Agreement (draft)

Prepared for :
[Client.FirstName] [Client.LastName], [Client.Company]

Prepared by :
[Developer.FirstName] [Developer.LastName], [Developer.Company]

This Software Development Agreement ("the Agreement") states the terms and conditions that govern the contractual agreement between [Developer.Company] (the "Developer") of [Developer.Address], and [Client.Company] (the "Client") of [Client.Address] who agrees to be bound by this Agreement.

WHEREAS, the Client has conceptualized [Short Description of Software] (the "Software"), and the Developer is a contractor with whom the Client has come to an agreement to develop the Software.

NOW, THEREFORE, in consideration of the mutual covenants and promises made by the parties to this Software Development Agreement, the Developer and the Client (individually, each a "Party" and collectively, the "Parties") covenant and agree as follows :

1. Product delivery.

The Client hereby engages the Developer and the Developer hereby agrees to be engaged by the Client to develop the Software in accordance with the specifications attached hereto as Item A (the "Specifications").

a. The Developer shall complete the development of the Software according to the milestones as described in the Specifications. In accordance with such milestones, the final product shall be delivered to the Client by [Delivery Date] (the "Delivery Date").

b. Except as expressly provided in this Agreement, the Developer shall not be obligated under this Agreement to provide any other support or assistance to the Client.

c. The Client may terminate this Agreement at any time upon material breach of the terms herein and failure to cure such a breach within 21 days of notification of such a breach.

d. If the Software as delivered does not conform with the Specifications, the Client shall within 21 days of the Delivery Date (the "Acceptance Date") notify the Developer in writing of the ways in which it does not conform with the Specifications. The Developer agrees that upon receiving such notice, it shall make reasonable efforts to correct any non-conformity.

e. The Client shall provide to the Developer written notice of its finding that the Software conforms to the Specifications within 21 days of the Delivery Date unless it finds that the Software does not conform to the Specifications as described in Section 2(a) herein.

2. Developer obligations

The Developer represents and warrants to the Client the following:

a. For a period of 21 days after the Delivery Date, the Developer shall provide the Client attention to answer questions or solve problems with regard to the operation of the Software up to 90 hours free of charge and billed to the Client at a rate of £50 per hour for any

assistance thereafter. The Developer agrees to respond to any reasonable request for assistance made by the Client regarding the Software within 7 days of the request.

b. For a period of 10 days after the Acceptance Date, the Software shall operate in accordance with the Specifications. If the Software malfunctions or in any way does not operate according to the Specifications within that time, then the Developer shall make every reasonable effort to ensure the Software operates according to the Specifications.

c. The Developer shall provide to the Client after Delivery Date a cumulative 2 days of training with respect to the operation of the Software, if requested by the Client.

d. The Developer agrees to indemnify, defend and protect the Client from and against all lawsuits and costs of every kind, including reasonable legal fees, pertaining to the Software due to the Developer's infringement of the intellectual rights of a third party.

e. Development and delivery of the Software under this Agreement are not in violation of any other agreement that the Developer has with another party.

f. The Software will not violate the intellectual property rights of any other party.

3. Client obligations.

The Client represents and warrants to the Developer the following:

THE CLIENT SHALL

a. Provide all documentation and subject matter relevant to the documented Specifications within 5 business days of the date of this Agreement. This may include any relevant directions, mock-ups, notations or memos, source code, legal documents and/or other briefs, or any other content which is necessary for the Developer to begin work.

b. Continue to furnish the Developer with any and all materials necessary to continue development relevant to the Specifications after the project is underway.

c. Make available to the Developer and it's employees, sub-contractors or independent contractors acting in it's stead any facilities, offices, personnel or internal services necessary to enable the Developer to carry out it's obligations as set forth by this Agreement.

d. Ensure that it's representative is available as reasonably required by the Developer to coordinate with and liaise between both parties in order to negotiate project progress.

e. Bring any claim it has against the Developer in relation to this Agreement within 28 days of the date on which the cause of action arose.

THE CLIENT AGREES to bear full responsibility for stages in the product lifecycle that are not herein initialled by the Developer:

	Analysis	Planning	Design	Develop	Testing	Deploy	Maintain
Man hours							
FTE							

4. Changes to Specification.

The Client may request that reasonable change be made to the Specifications and tasks associated with the implementation of the Specifications. If the Client requests such a change, the Developer will use it's best efforts to implement the requested change at no additional expense to the Client and without affecting the Delivery Date of the Software.

a. In the event that the proposed change will, in the sole discretion of the Developer, require a delay in the Delivery Date or would result in additional expense to the Client, then both Parties shall confer and the Client may either withdraw the proposed change or require the Developer to deliver the Software with the proposed change and subject to the delay and/or additional expense.

b. The Client agrees and acknowledges that the judgement as to if there will be any delay or additional expense shall be made solely by the Developer.

5. Compensation

In consideration, the Client shall pay the Developer a maximum total fee for all work under this Agreement of [balance].

a. The Client agrees to pay the Developer a non-refundable 25% portion of the balance prior to work commencing.

b. The Developer may provide additional services to the Client as agreed in writing by the Parties. These additional services shall be subject to additional fees, as agreed by the Parties.

c. Fees billed under the hourly rate shall be due and payable upon the Developer providing the Client with an invoice. Invoices will be provided for work completed by the Developer once every calendar month.

6. Dispute process.

a. If any dispute arises in connection with this Agreement, the Parties will attempt to settle it by negotiation. Each Party will use reasonable endeavours to resolve any dispute amicably by direct discussion.

b. If the dispute is not resolved by negotiation within 28 days of receipt of a written 'invitation to negotiate', the Parties will attempt to settle it by mediation in accordance with the Centre for Effective Dispute Resolution (CEDR) Model Mediation Procedure.

c. To initiate the mediation, a Party must serve notice in writing (the "ADR" notice) to the other Party requesting mediation. A copy of the request should be sent to CEDR.

d. If the dispute is not resolved by mediation within 60 days of the service of the ADR notice, or either Party fails to fully participate or to continue to participate in the mediation, the dispute shall finally be resolved by the courts of England and Wales.

e. Notwithstanding the existence of a dispute, each Party shall continue to perform it's obligations under this Agreement as far as possible as if no dispute had arisen pending the final settlement of any matter referred to mediation or court.

f. The costs of mediation shall be borne by both parties, unless otherwise agreed in writing by both Parties.

7. Confidentiality.

The Client understands that it may be necessary to reveal trade secrets, intellectual property and other confidential information throughout the duration of this Agreement in order for the Developer to complete it's work.

The Developer understands the business risk to the Client and agrees to make every reasonable effort to protect this information from a material breach, in addition to agreeing the following:

a. The Intellectual Property Rights in all original documentation (including source code and object code), together with any related materials or software provided by the Client for the duration of this Agreement shall remain the property of the Client.

b. The Intellectual Property Rights in any new software generated by the Developer on behalf of the Client (including source code and object code), along with any relevant project documentation or materials created as a part of this Agreement shall become the property of the Client upon final payment for services rendered under this Agreement.

c. The Developer shall not

(i) disclose to any third party the business of the Client, details regarding the Software, including, without limitation any information regarding the Software's code, the Specifications, or the Client's business (the "Confidential Information"),

(ii) make copies of any Confidential Information or any content based on the concepts contained within the Confidential Information for personal use or for distribution unless requested to do so by the Client, or

(iii) use Confidential Information other than solely for the benefit of the Client.

8. Intellectual property rights in the Software.

The Parties acknowledge and agree that the Client will hold all intellectual property rights in the Software including, but not limited to, copyright and trademark rights. The Developer agrees not to claim any such ownership in the Software's intellectual property at any time prior to or after completion and delivery of the Software to the Client.

9. No modification unless in writing.

10. Force Majeure.

Neither Party shall have liability under or be found in breach of this Agreement for delays or failures in performance which are beyond the circumstances of the reasonable control of the Party. If such circumstances continue for an extended period (10 weeks or more), both Parties accept the following:

a. Costs incurred from such a delay shall be the obligation of the Party from which the delay occurred.

b. If such a delay continues for more than 10 weeks, either Party may terminate the Agreement by providing written notice to the other Party, with both Parties accepting that

the Client shall pay the Developer a reasonable sum with respect to any work carried out prior to such termination.

11. Applicable law.

This Agreement and the interpretation of it's terms shall be governed by and construed in accordance with English law.

IN WITNESS WHEREOF, each of the Parties has executed this Software Development Agreement, both Parties by it's duly authorized office, as of the day and year set forth below.

[Signature]
[Date]
[Developer.Company]
[Developer.FirstName] [Developer.LastName]

[Signature]
[Date]
[Client.Company]
[Client.FirstName] [Client.LastName]

Item A : Specifications.

i. Software description
ii. Technical specification
iii. Functional requirements
iv. Testing requirements
v. Milestones

i. Software description. The Software to be developed under this Agreement is [Description of Software].

ii. Technical Specification. The Software shall be developed using the following technologies : [List of Technologies]. The Software shall be compatible with the following platforms : [Compatible Platforms].

iii. Functional requirements. The Software shall include the following features and functionality : [Features and Functionality].

iv. Testing requirements. The Software shall perform as expected when tested under the following conditions : [Test Acceptance Criteria].

v. Milestones. [Milestones].

Appendix 2 : Specifications for We Make Malvern

[Client.FirstName]
Any

[Client.LastName]
Person

[Client.Company]
Made Up Company

[Client.Address]
1 High Street, Somewhere, AA1 1AA.

[Short Description of Software]
A community energy transition showcase for the Malvern Hills, UK.

[Delivery Date]
29th October 2026

[Balance]
£3,500

[Long Description of Software]

When a community has a proposal for rapid energy transition, they need a presentation tool that can help achieve consensus and green-light funding.

This map shows working models of (i) how + where the energy (electricity) used by the community is made, (ii) how + why it's consumed and (iii) how placing wind-turbines + solar panels in the community affects electricity prices, energy resilience + views.

Advanced presentation tools. By inviting residents, land-owners and permission-holders to transparently see how ideas, plans + possibilities will affect them, it's easier to achieve consensus + green-light funding. Large buttons enable the user to swap models in real-time to produce before+after effects.

The map is a living cartographical document, open 24/7/365 by invite-only. It is maintained by the Client.

[List of Technologies]

Meta Quest headset, Meta SDK, Unity, Photon Unity Networking, C#, Visual Studio, Blender, Android Development Bridge, MetaShape, Scaniverse, RenderDoc, OpenStreetMap, DEFRA LiDAR, QGIS, Gimp, GitHub, Windows.

[Compatible Platforms]
Meta Quest 3+

[Features and Functionality]

(1) *Walk, talk and interact within a 1:1 scale recreation of the Malvern Hills, UK.*

Users should realise their position on the Hills from direct observation, referencing local knowledge. The Beacon + British Camp should be visible. Users can see each other + talk to each other, using their hands to manipulate the environment.

(2) *2 levels of detail (LOD0 : 300m^2 immediate environment, LOD1 : 7km-diameter distance)*

(3) *Visualization of a community energy infrastructure proposal with interactive controls for resource use/cost for different energy sources.*

The purpose of the Software is to demonstrate a local energy transition scheme. A house is presented as consuming/generating an amount of energy in real-time. The cost of this energy is presented as a bill to the householder + as a CO2e emission to the planet. The input source + percentage of grid + local energy can be controlled by the user. The user can turn 6 household appliances on/off. The user can view 4 stages of local solar and wind energy infrastructure.

(4) *User programmable.*

To heighten the sense of immersion in an experience designed to showcase energy transition, the user can program the time of day/year, thus changing the sun's position + changing the value of solar energy generation in real-time. The sun's position in the sky should be accurate for the observer's latitude and longitude.

The user should be able to simulate varying wind-speeds, with similar effects on wind power generation.

The user should be able to vary the speed of time (0, 1x, 10x, 100x, 1000x).

The user should be able to simulate the sea-level rising and falling.

(5) *Max 20CCU.*

The Developer and Client have agreed to use a third-party networking solution to achieve multiuser capability. A license limitation for the 'free' version of the chosen transport architecture as stated in [List of Technologies], PUN (Photon Unity Networking), caps the number of CCU (concurrent users) at 20. It is noted that this 'free' PUN license precludes public release, limiting this Software to non-commercial use.

(6) *Tutorial.*

A short tutorial acclimatises the user with the mechanics of movement + interaction in an immersive map.

(7) *Presentation tools.*

One user can be the 'Presenter'; this user can control what everyone else is seeing AND control whether they can move (rotation is unaffected). This effectively turns the experience into a presentation tool (freeze users -> move them to a new position -> swap models -> unfreeze).

(8) *Expressive, customizable Avatars.*

Users can see each other's mouths move when they talk, with expressive facial features such as eyebrows. Users can customize their Avatar's body and clothing.

[Test Acceptance Criteria]

(1) Max 20CCU.
(a) Detects + survives 21CCU attempt.

(2) Handling networked objects.
(a) Users can pass objects to/from each other's hands.
(b) Presenter can change what people are seeing.

(3) Surviving common faults.
(a) Build detects + survives network connection disconnect/reconnect.
(b) Users can enter/re-enter multiuser session.

(4) Achieve a sense of local immersion.
(a) Users can see Worcestershire Beacon + Herefordshire Beacon from the Wyche Cutting.

(5) Interactive 'Grid Game'.
(a) Users can play with a set of levers and controls to mix energy inputs and outputs for local and national electricity generation.

[Milestones]

12th August	Analysis meeting
14th-18th	On-site survey
18th	Planning meeting
19th	Design meeting
20th	Development---Programming---Networking
22th	Development---Programming---UI
28th	Model---1st draft---LOD0
15th September	Model---1st draft---LOD1
1st October	Testing---first meeting
3rd	Development---Programming---Beta build meeting
5th	Testing---second meeting
10th	Model---completed---LOD0+1
15th	Alpha build meeting
22th	Final build delivery
29th	Deployment

APPENDIX 3 : ERROR LOGS

When software doesn't work as expected or crashes, inspecting the log files is a troubleshooting goldmine.

Blender : recreate the crash in debug mode using
C:\ProgramFiles\BlenderFoundation\Blenderx.x\blender_debug_log.cmd

Unity : [Project name] -> Logs -> Editor.log

Windows OS : Event Viewer -> Windows Logs -> System

Meta Quest Horizon OS : C:\ProgramFiles\Oculus\Support\oculus-diagnostics\OculusLogGatherer.exe

Appendix 4 : Developer background

1982 Sinclair ZX81. BASIC + machine code.

1989 DEC PDP-11. Unix/BSD + Windows. Pascal + C.

1996 Environmental Resource Science BSc.

2015 Oculus Rift Development Kit v2 + driving simulators.

2018 Idea for We Make Earth[33], an immersive map of the planet for visualising the change in resource distribution that achieves equitable sustainability *from any individual's perspective*. Start learning Unity, C# + Blender.

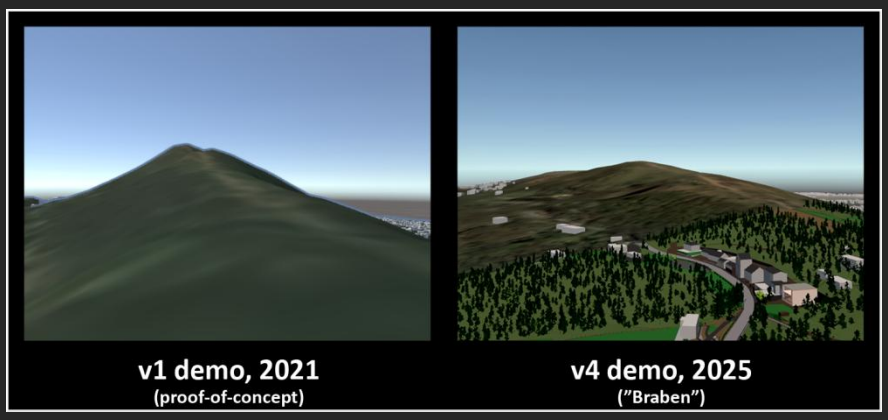
2021 **Demo v1**.

2025 **Demo v4**.
Software Development Agreement.
Website + identity.

2026 **Demo v5** (We Make Malvern[34]) -> public Beta.
Guide to making an immersive map (this document).

Appendix 5 : Credits

Thanks to all the giants who made the software and assets that helped turn this idea into content (please contact me if I've left you out). Apologies to the owners of the ears that I've chewed off for so many years. I hope it was worth it.



Unity : Free 'Personal' licence for small indie teams.

https://unity.com

Visual Studio Community : Free code editor with C#/Unity integration.

https://visualstudio.microsoft.com/vs/community

Blender : Free and open-source 3D toolset.

https://www.blender.org

[33] wemakeworlds.com/earth.html.
[34] wemakeworlds.com/malvern.html.

Blosm : real-world terrain data for Blender in a few clicks. Free (donation suggested) or Pro.
https://github.com/vvoovv/blosm

BlenderGIS : The critical BlenderGIS plugin from domlysz. It bridges Blender with the OpenTopography web service to import GIS data files. Licensed under the GNU General Public License v3.0 for Commercial Use, Modification, Distribution, Patent Use and Private Use.
https://github.com/domlysz/BlenderGIS

QGIS : A free and open-source Geographic Information System (GIS).
https://qgis.org

RenderDoc : RenderDoc is licensed under the MIT License.
https://renderdoc.org

OpenStreetMap : Map copyright OpenStreetMap contributors, data available under the Open Database License.
https://opendatacommons.org
https://www.openstreetmap.org

Hot air balloon : Model by blinkstarcgi.
https://www.cgtrader.com/3d-models/aircraft/other/hot-air-balloon-714ccc0d-f42e-4176-b853-d80884f45c87

Trees : The poplar tree model, dust and falling leaves are from Nature Manufacture's 'Meadow Environment pack'.
https://assetstore.unity.com/packages/3d/vegetation/meadow-environment-dynamic-nature-132195

The poplar was LOD'd using Roundyyy's Tree Imposter.
https://github.com/roundyyy

Conference Centre : Model adapted from DDDviz's 'Conference Centre 3'.
https://www.turbosquid.com/3d-models/conference-room-6-1586062

Quick Outline : The yellow outline effect found on electrical items that turn on/off is from an asset by Chris Nolet.
https://assetstore.unity.com/packages/tools/particles-effects/quick-outline-115488

Avatars : The expressive and customizable avatars, including the hand models, are made by Meta.
https://assetstore.unity.com/packages/tools/integration/meta-avatars-sdk-271958

Avatar networking : Network programming for Meta Avatars adapted from Chilli Games' 'Meta Avatars Multiplayer Template'. Special thanks for providing outstanding support when I needed it most.
https://assetstore.unity.com/packages/tools/network/multiplayer-meta-avatars-vr-template-211918

Bird Flocks : The crow and butterfly assets are by Unlock Software in their 'Bird Flock Bundle'.
https://assetstore.unity.com/packages/3d/characters/animals/birds/bird-flock-bundle-25576

Sun positioning : Getting the sun to be at the correct position for all times of the day for the location is from Hessburg's 'SunLight – Location based Time of Day' scripts.
https://assetstore.unity.com/packages/tools/particles-effects/sunlight-location-based-time-of-day-66399

Boiler : model by NicoNicoNii.
https://www.cgtrader.com/free-3d-print-models/house/other/hot-water-tank-6974bf91-01ae-44a2-92f5-26ce9b1768ed

Television : (attribution pending)

Washing machine : (attribution pending)

EV Charger : (attribution pending)

Fridge : (attribution pending)

Tiles : the six tiles that are dispensed in the grid game (coal, gas, biomass, nuclear, hydro and solar) are adapted from Alex Reeves' 'Renewable and alternative energy low-poly' model. Useable under CC Attribution License. Special mention for specifically making the models for other people to use.

https://sketchfab.com/3d-models/renewable-alternative-energy-low-poly-943f1c168b4c47758ede9a961f523493

Tile dispenser : original model by supersmashbros113.

https://www.cgtrader.com/free-3d-models/interior/other/simple-low-poly-soda-dispenser

Earth globe : model by gabriel-saraiva-cesar.

https://www.cgtrader.com/3d-models/space/planet/low-poly-earth-15849d65-f1aa-43af-aa5a-92e75aada4a0

Solar Panels : model by Marc Mons.

https://www.turbosquid.com/3d-models/solar-panel-1171717

Wind turbine : model by romanejaquez.

https://www.cgtrader.com/free-3d-models/industrial/other/windmill-02b0572a-106b-4a76-99a5-44272e7a9a43