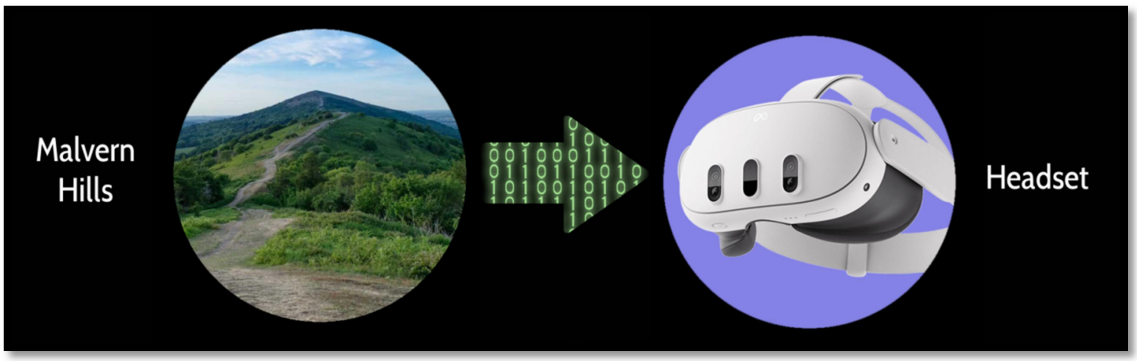
(1) Overview

We Make Malvern is an immersive map of the Malvern Hills, UK, for the Meta Quest3+ headset. It's free to use and adapt.



You access the map by being 'inside' a 7km-diameter landscape with realistic stereoscopic depth and 1:1 scale. Accurately presented high-detail environments offer the convincing expression of space-reality that's key to communicating the nature of somewhere. It's the next-best thing to actually being there. And in some ways, it's better.

Traditionally we've described our world with words and pictures, using technology to help encode and recall information. The immersive map is a step-change in how we can present place and possibility, a storytelling tool with which to inform and inspire.

We Make Malvern is a self-written application developed on a range of 3rd-party hardware and software tools. The application is in open Beta (please request access) for consultation prior to public release.

(1.1) Demonstration

A community energy transition plan acts as an introduction to the usefulness of a directed immersive experience, where people can visualize proposed change *in context to their own lives*. It's designed to promote understanding and help achieve consensus in green-lighting community benefit projects.

First-time users are offered a short tutorial on the mechanics of immersive maps.

In 2024, Labour committed to decarbonising the UK's electricity system by 2030. You can see thirteen counties from atop the Malvern Hills on a good day, but only one wind turbine.

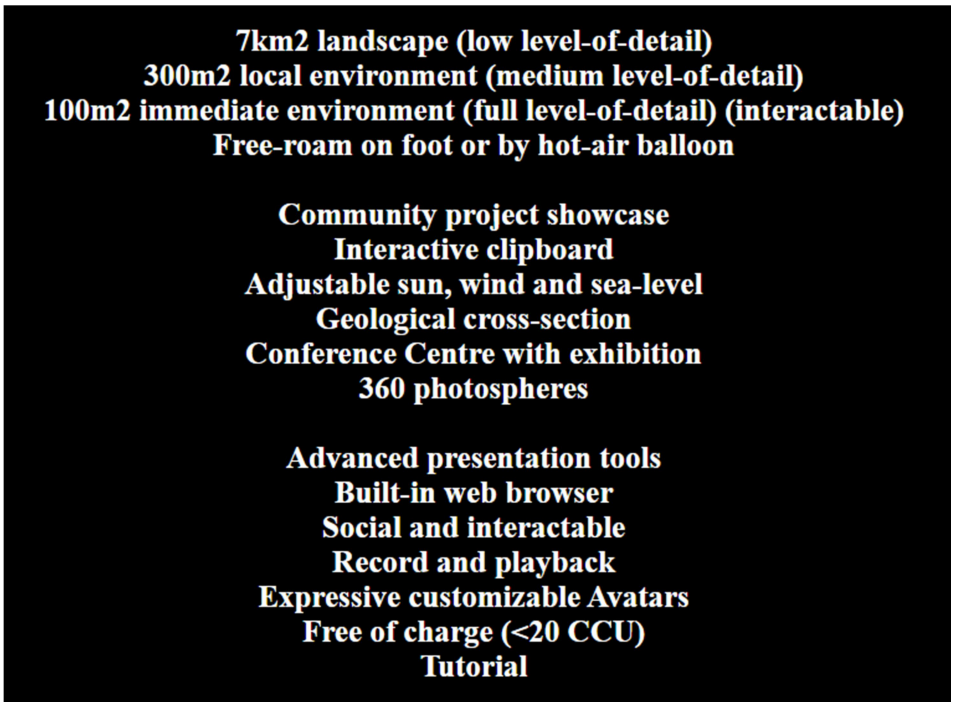
Transforming our personal energy use – how much we consume and how it's made – is an enormous challenge. We can do a lot as individuals in our own home, but are stronger if we work together on larger infrastructure projects. So when a community has a proposal for rapid energy transition, they need a presentation tool that can help achieve consensus and secure funding.

Local residents, land-owners and stakeholders can transparently see how possibilities for energy adaptations will affect them (eg. how the infrastructure will look from their bedroom window, how much they'll be paying for their electricity and what it does to their carbon budget). It's easier to achieve consensus when everybody is on the same page.

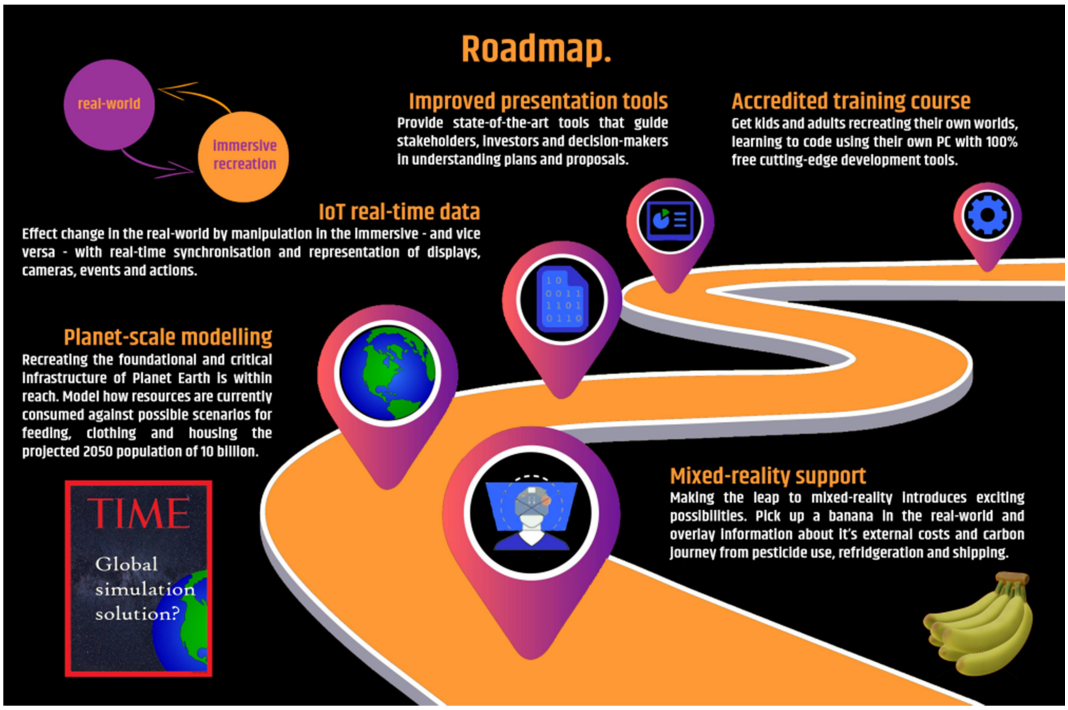
Present and future scenarios are visualized by making models of each and switching between them. One model might show energy resources as they're currently consumed, the other showing a proposal for local energy generation and distribution. Different phases of development are selected from an intuitive clipboard which shows solar gain with the relative cost to the householder, giving an indication of the economics of community benefit schemes.

Recreating the landscape of communities and the infrastructure of our core systems allows for a shared appreciation of space and resources. As a forum for debate on proposals, the audience meets *inside a recreation of those proposals*. Meetings and presentations can be recorded and replayed.

(1.2) Features



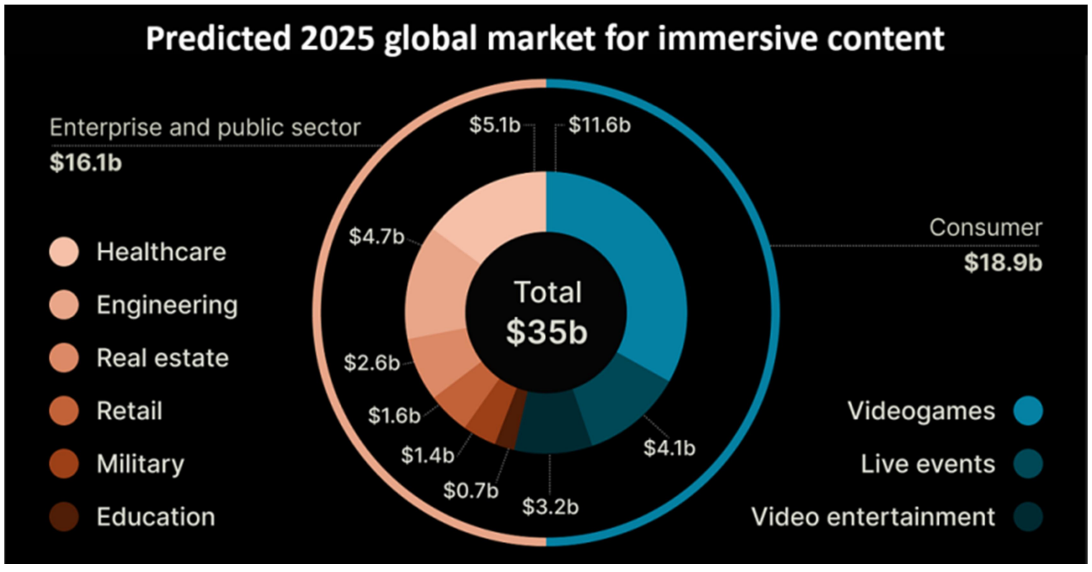
(1.3) Roadmap



(1.4) Commercial applications

Immersive recreations offer an entirely new way for people to discover and engage with commercial establishments.

For example, a hotel might consider offering (i) booking+payment via built-in web-browser *whilst standing in the room you want to book*. (ii) real-time human assistance. (iii) 7km^2 1:1 environment for familiarization with local amenities and landscape - drive from the motorway to the hotel - explore the local area from a hot-air balloon. (iv) windows that show an accurate representation of the actual view. (v) interactable room furnishings and contents.



(2) Hardware

Consumer-grade virtual-reality hardware has matured over the last decade thanks to advances in optics, processing power and 'free' software.

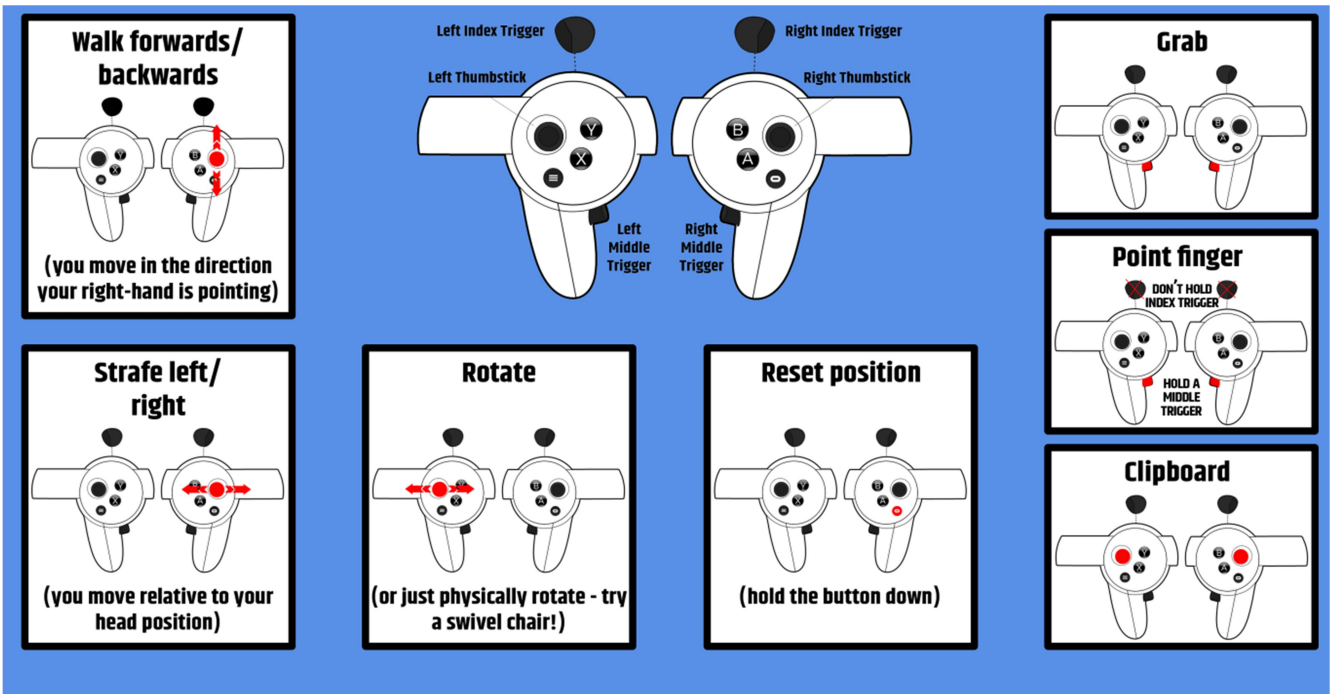
The Meta Quest3+ headset is cheap, fun-to-use and available everywhere. It's the runaway success of the last few years in delivering immersive content to the masses. Suitable for older kids and adults, it's sold millions of units and continues to impress with new features and functionality.

The headset is a box that places two small screens in front of your eyes. Normal vision is replicated by a built-in computer which continuously refreshes the display so that we see a stereoscopic 3D image *that changes as we move*. The effect is what we call immersion, and it comes free with any headset.

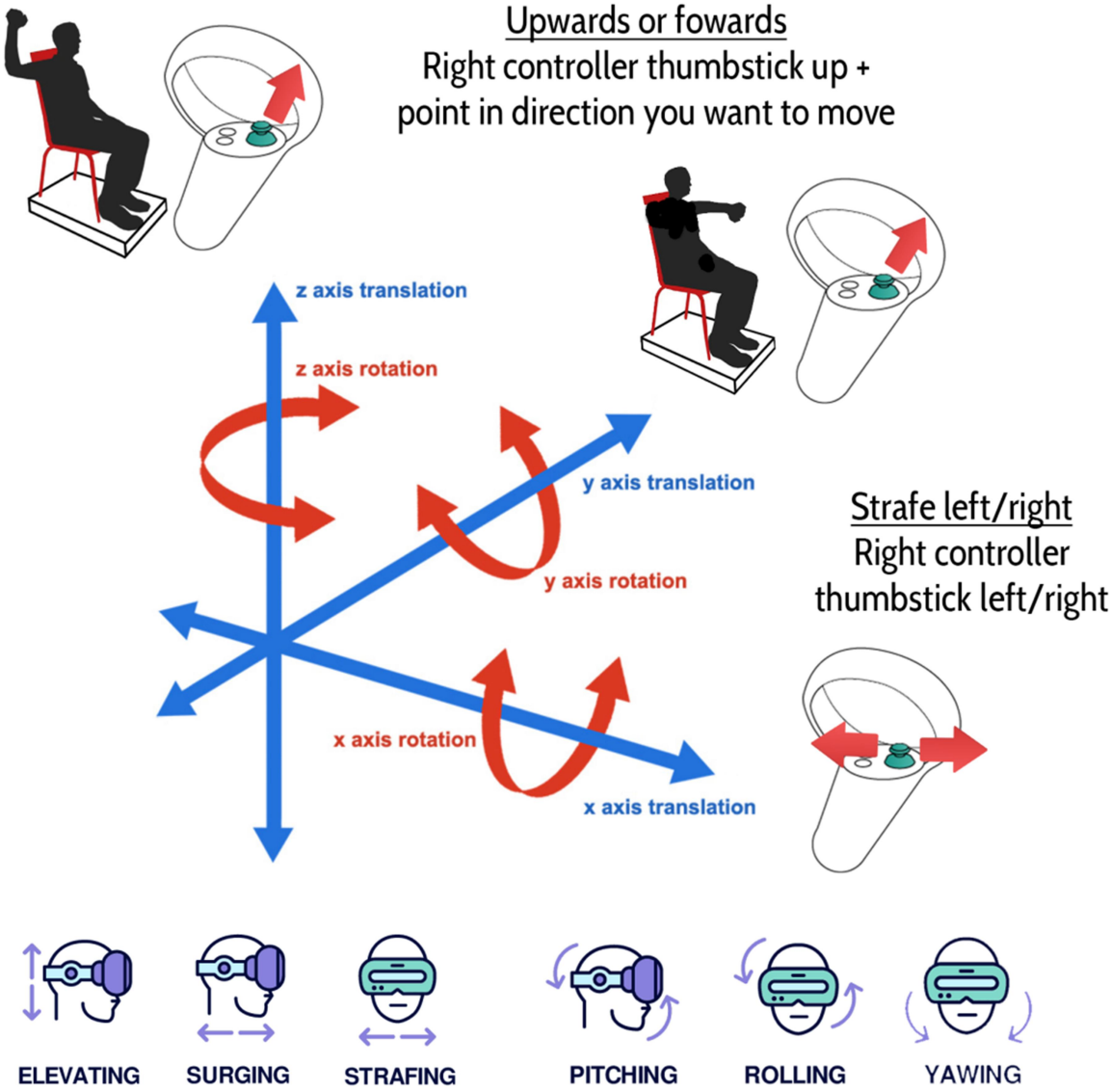
The trick, of course, is for the visitor to become actively engaged with the illusion and feel the pleasing level of sensory synthesis found in everyday life. This is achieved on multiple fronts, not least in the implementation of the movement mechanic.

(2.1) Movement and interaction

Moving around in an immersive map is achieved *by physically moving around* or, as you won't be able to 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*. This may seem complex on paper but quickly becomes intuitive in practise.



Moving about in 6-DOF (Degrees of Freedom) in the Malvern Demo



(2.2) First-time set-up

Apart from a Meta Quest3+ headset and an invitation, you'll need :

To complete set-up, you'll agree to and acknowledge many terms :

Safety and Warranty guide *full of legal warnings*
Parent Information guide
Account terms
Avatar terms

(3) Technical breakdown

As built in the Unity, the project has four Hierarchy elements :

(i) Networking.
(a) Network Manager : Connection to Photon servers and Room creation.
(b) Player Manager : Synchronization of visitor head+hand position; expressions.
(c) AvatarSdkManagerHorizon : Networks Meta Avatars v2.
(d) LipSynInput : simulates lip movement when talking.
(e) Voice : Networks visitor voice when talking.

(ii) Player.
(a) Movement : getting around in 6-DOF.
(b) CenterEye : stereoscopic vision.
(c) Left/Right Hand : interaction; pressing buttons and picking things up.

(iii) Pad.
(a) Pads : three pads are stackable, each with access to any PadScreen.
(b) PadScreens : minimap, LOD, solar, readout, sun, network and settings.

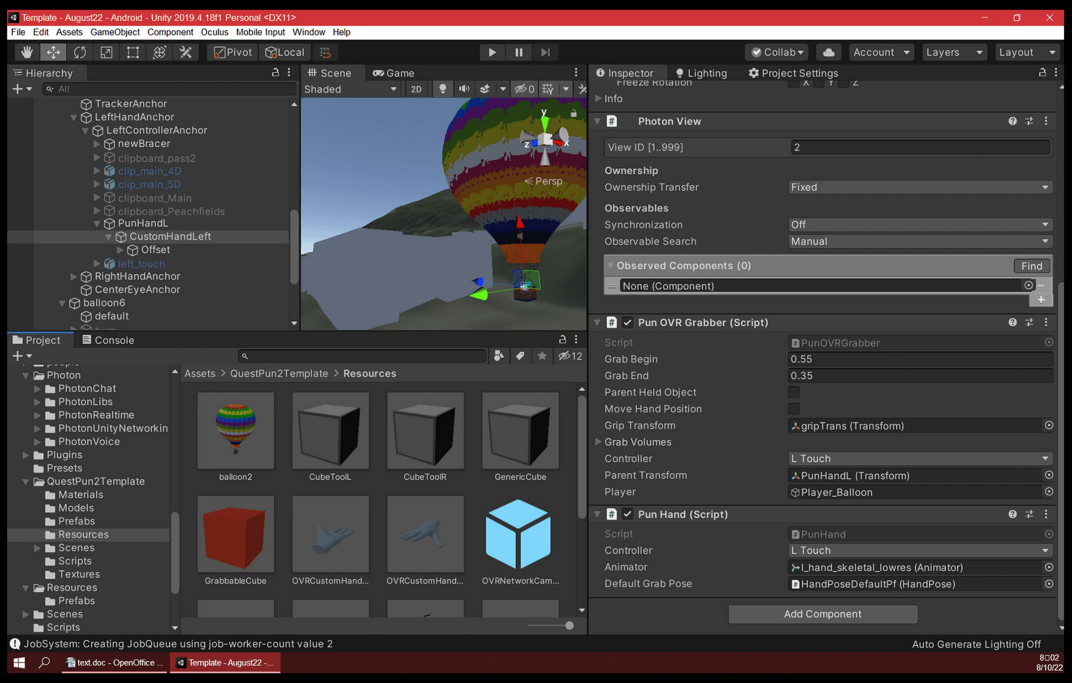
(iv) World.
(a) Sun : shows the correct position of the sun for any lat/lon and any time; simulates stars at night.
(b) LOD0 : high level-of-detail environment model.
(c) LOD1 : low level-of-detail environment model.
(d) Geology : geological cross-section model.
(e) Mechanics : interactive objects, birds+butterflies, trees etc.

Components attached to these elements work together to create the computer-generated display inside the headset that allows us to experience the digital world.

Dozens of scripts are employed, most of which are self-written. Notable exceptions are the core networking scripts (Meta and Photon SDK's), the template for integrating Meta Avatars, and the sun positioning code. Use of third-party (or 'off-the-shelf') solutions to specific problems cuts development time and avoids reinventing the wheel.

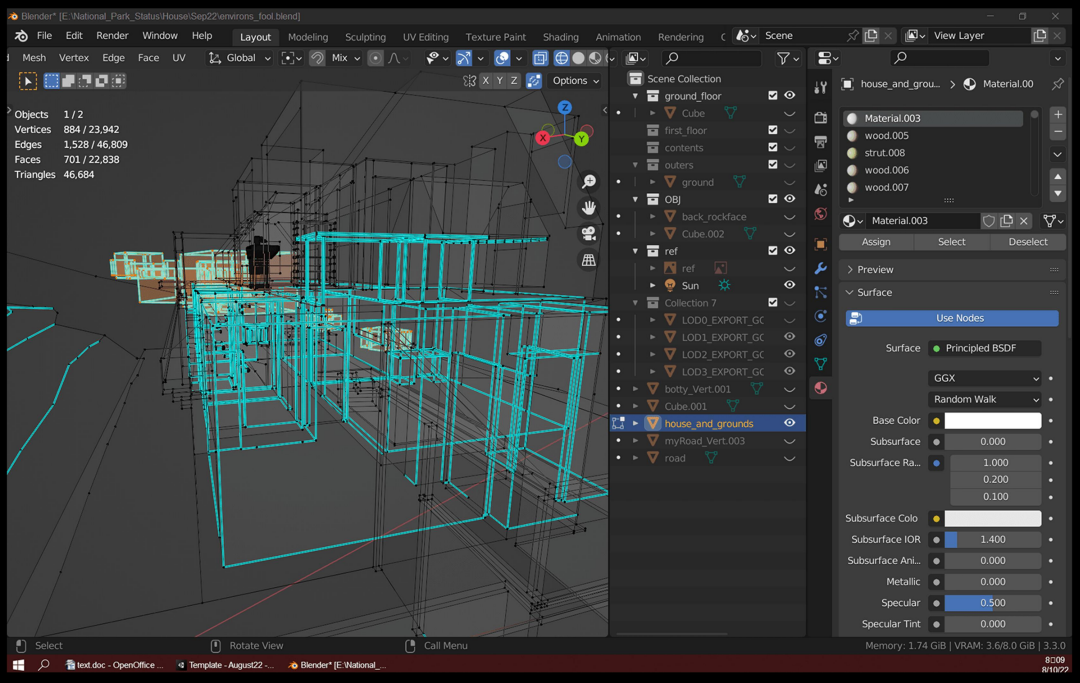
Landscape models were generated by integration with publicly available datasets. The large LOD1 environment, for instance, is created in the Blender 3D modelling program directly from a plugin, whilst the high-detail LOD0 has been entirely hand-crafted. The Conference Centre is a paid-for asset, heavily modified. The addition of a dedicated digital artist on any development team is strongly recommended.

(3.1) Software tools 1/3



unity

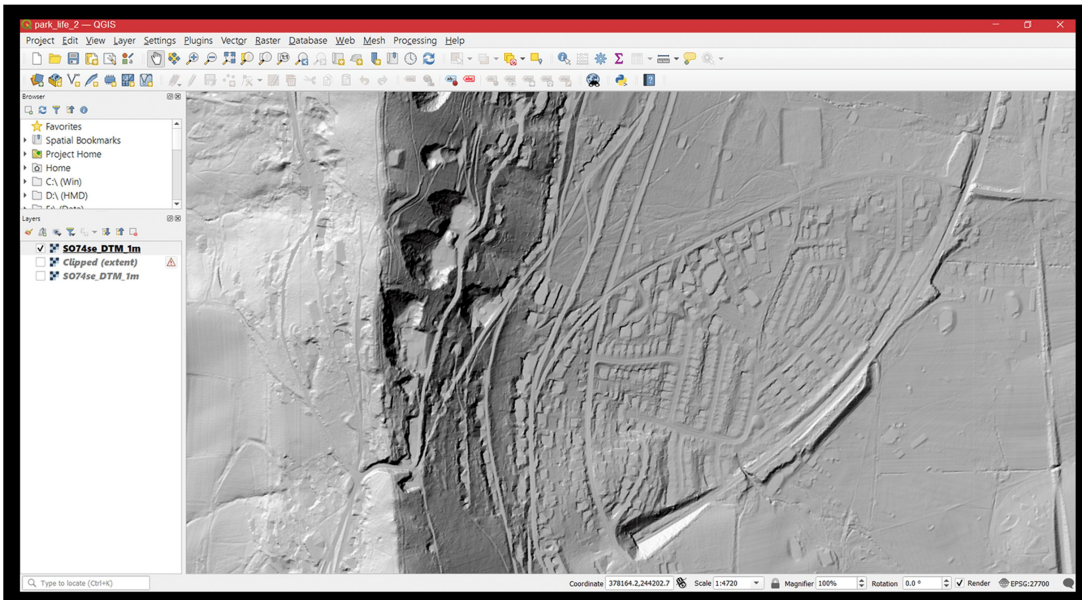
The Malvern demo was built inside the industry-leading Unity development environment. No license is required for projects with revenue or funding less than $100K in the last 12 months.



blender

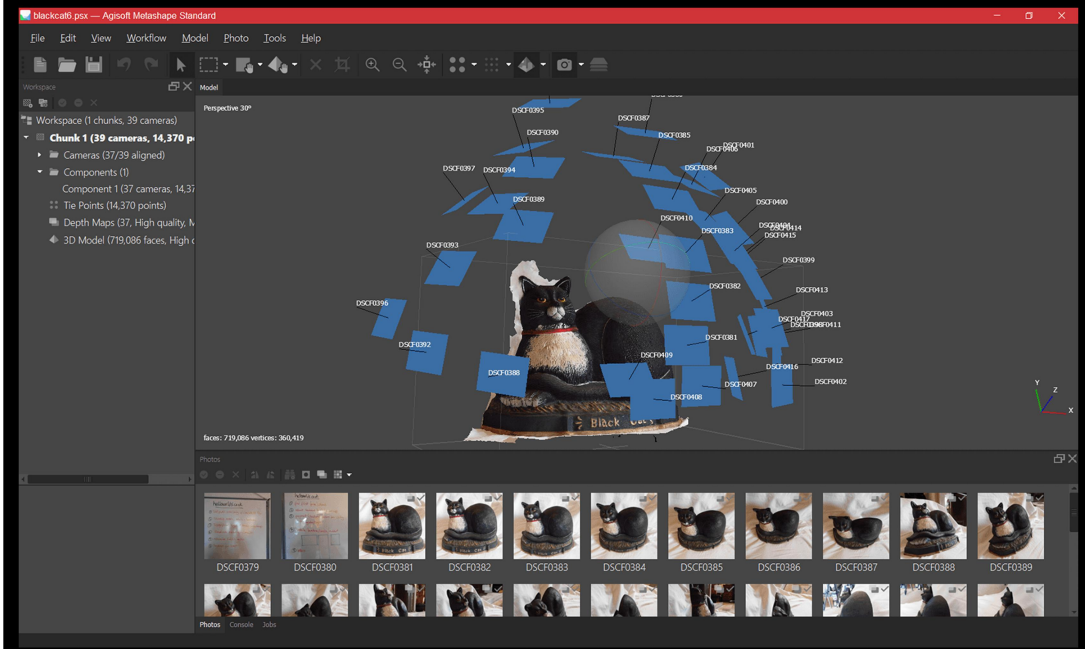
Assets and models were created using the open-source Blender 3D creation suite.

(3.1) Software tools 2/3



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.



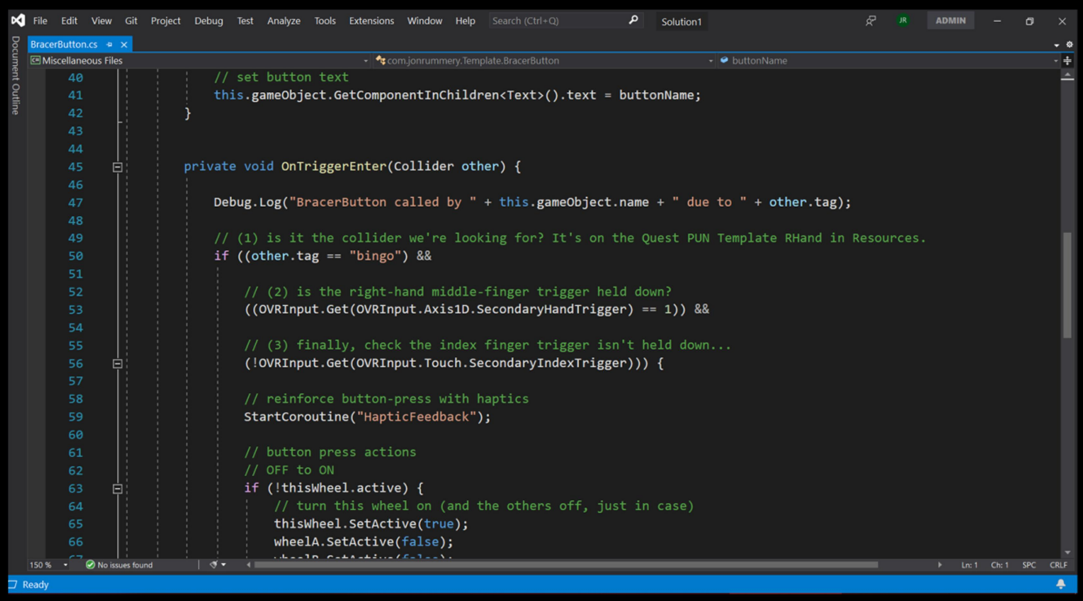
Scaniverse

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.

(3.1) Software tools 3/3



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

Android Debug Bridge

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

(3.2) Script (example) : MovePlayer 1/3

```
// <summary>
// MovePlayer.cs
// query OVRInput to move, strafe + rotate.
// </summary>

namespace wemakeworlds {

  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;

  public class MovePlayer : MonoBehaviour {

    public GameObject playerPivot;
    private GameObject _cam;
    [Tooltip("Lower the value for movement along stick y-axis to register when close to x-axis")]
    [Range(0, 1)] public float stickTolerance;
    [Header("Speed (right-stick up/down + left/right")] public int moveSpeed;
    [Header("Rotation speed (left-stick left/right")]
    public int rotationSpeed;
    [Header("Index trigger boost")] [Range(0, 10)]
    public int boostSpeed;
    [HideInInspector] public bool _isSliding;
    private float _newYMoveSpeed;
    private GameObject _rightHand;
    [HideInInspector] public Vector2 _leftStickInput;
    [HideInInspector] public Vector2 _rightStickInput;
    private float _rotation;
    private bool _isSnapping;
    private float _snapLeftSpeed;
    private float _snapRightSpeed;
    private Vector3 _movement;
    [HideInInspector] public bool _isFrozen = false;
    public GameObject frozenLight;
    [HideInInspector] public float _supermanTriggerInput;
    [HideInInspector] public float _boostTriggerInput;
    [HideInInspector] public enum RotationStyles {
      snap,
      smooth
    }
    [HideInInspector] public RotationStyles rotationStyle;

    private void Start() {

      // reference the headset position to strafe
      _cam = GameObject.FindGameObjectWithTag("MainCamera");
      Debug.Log("Camera found at " + _cam.name);

      // get a reference to the hand/controller used for forward direction
      _rightHand = GameObject.FindGameObjectWithTag("RightHand");

      // set initial movement style
      rotationStyle = RotationStyles.snap;

      // set the snap values
      _snapLeftSpeed
      _snapRightSpeed = rotationSpeed;
```

(3.2) Script (example) : MovePlayer 2/3

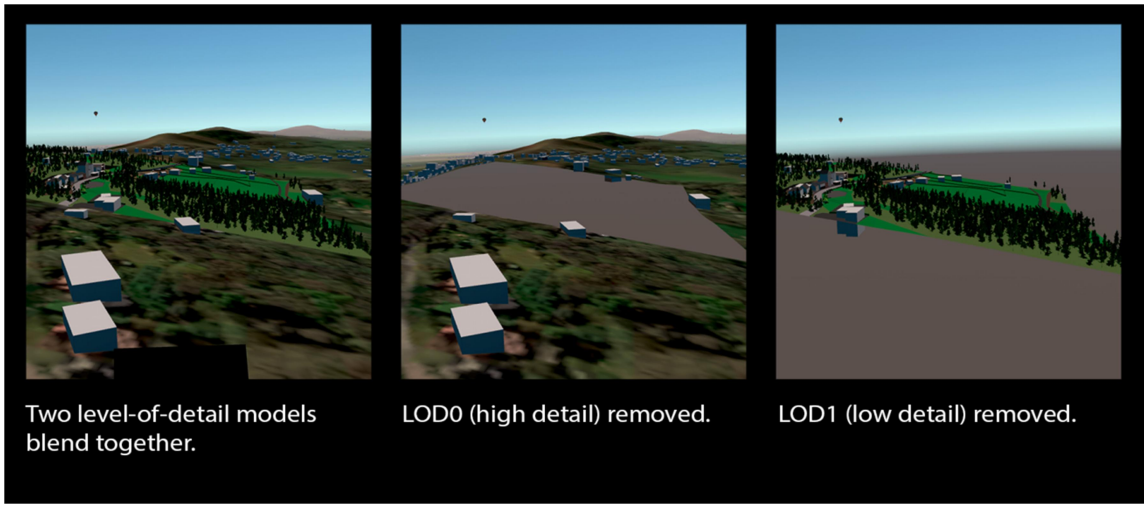
```
}

public void UpdateFrozenLight() {

   if (_isFrozen)
     frozenLight.SetActive(true);
   else
     frozenLight.SetActive(false);
}

void Update() {

   // check that a slider isn't being slid (no input wanted if so).
   if (!_isSliding) {

      GetOVRInput();
      ProcessOVRInput();
   }
}

void GetOVRInput() {

   // query the thumbsticks (Vector2 ((-1->1),(-1->1)) for (x,y))
   _leftStickInput = OVRInput.Get(OVRInput.Axis2D.PrimaryThumbstick);
   _rightStickInput = OVRInput.Get(OVRInput.Axis2D.SecondaryThumbstick);

   // ...and the two index-finger triggers for boost and superman modes (float)
   _supermanTriggerInput = OVRInput.Get(OVRInput.Axis1D.PrimaryIndexTrigger);
   _boostTriggerInput = OVRInput.Get(OVRInput.Axis1D.SecondaryIndexTrigger);
}

void ProcessOVRInput() {

   // ROTATION - two types :
   if (rotationStyle == RotationStyles.smooth) {

      // SMOOTH
      // left stick x-axis for rotation, but only if the left-stick is moved tolerably
      // smooth out the rotation
      _rotation = (rotationSpeed * _leftStickInput.x * Time.deltaTime);

      // rotate around the y-axis of player
      playerPivot.transform.Rotate(0f, _rotation, 0f, Space.Self);
   }

   else {

      // SNAP
      if (_leftStickInput.x == 0f)
         _isSnapping = false;

      if (!_isSnapping && (_leftStickInput.x != 0) && (_leftStickInput.y < stickTolerance)) {

         _isSnapping = true;

         // moving the stick to the left requires a -ve y rotation value
         if (_leftStickInput.x < 0)
```

(3.2) Script (example) : MovePlayer 3/3

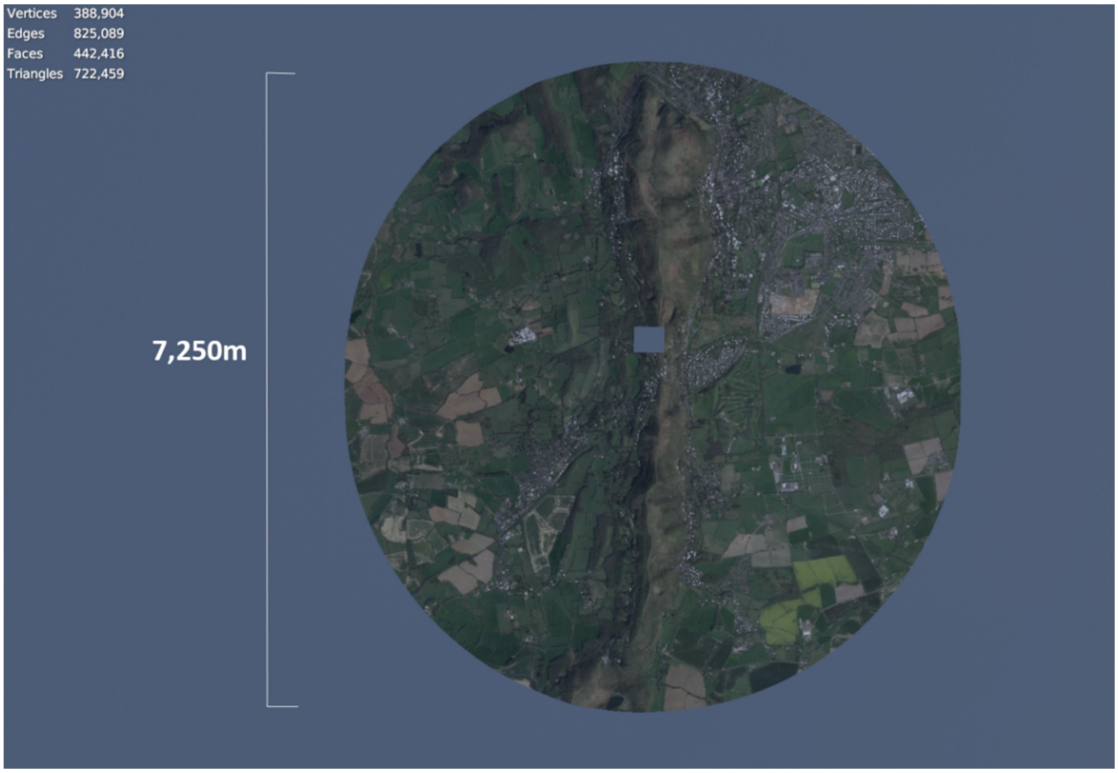
```
            _rotation = _snapLeftSpeed;
          else
            _rotation = _snapRightSpeed;

          playerPivot.transform.Rotate(0f, _rotation, 0f, Space.Self);
        }
      }

      // Player can be frozen by another Player (rotation is unaffected)
      if (!_isFrozen) {

        // STRAFE
        // right stick x-axis for lateral (side-to-side) movement
        if (_rightStickInput.x != 0 && _rightStickInput.y < stickTolerance) {

          // laterally move relative to headset position
          _movement = (_cam.transform.rotation * Vector3.right * PersistantData.moveSpeed * _rightStickInput.x
* Time.deltaTime);
          playerPivot.transform.position += _movement;
        }

        // FORWARD/BACKWARD
        // right stick y-axis for forward/backward movement
        if ((_rightStickInput.y != 0f) && (_rightStickInput.x < stickTolerance)) {

          // Yeah! we're gonna do some moving!
          _newYMoveSpeed = PersistantData.moveSpeed;

          // is Boost Mode enabled?
          if (_boostTriggerInput != 0f)
            _newYMoveSpeed += (boostSpeed * _boostTriggerInput);

          // is Superman Mode enabled?
          // NB this value has been deprecated : boost and superman speeds are the same
          if (_supermanTriggerInput != 0f)
            _newYMoveSpeed += (boostSpeed * _supermanTriggerInput);

          // is God Mode enabled?
          if (_boostTriggerInput != 0f && _supermanTriggerInput != 0f)
            _newYMoveSpeed += (boostSpeed * boostSpeed * _boostTriggerInput);

          // move forward/backward relative to right hand's position
          _movement = (_rightHand.transform.forward * _newYMoveSpeed * _rightStickInput.y *
Time.deltaTime);
          playerPivot.transform.position += _movement;
        }
      }
    }
  }
}
```

(3.3) Models

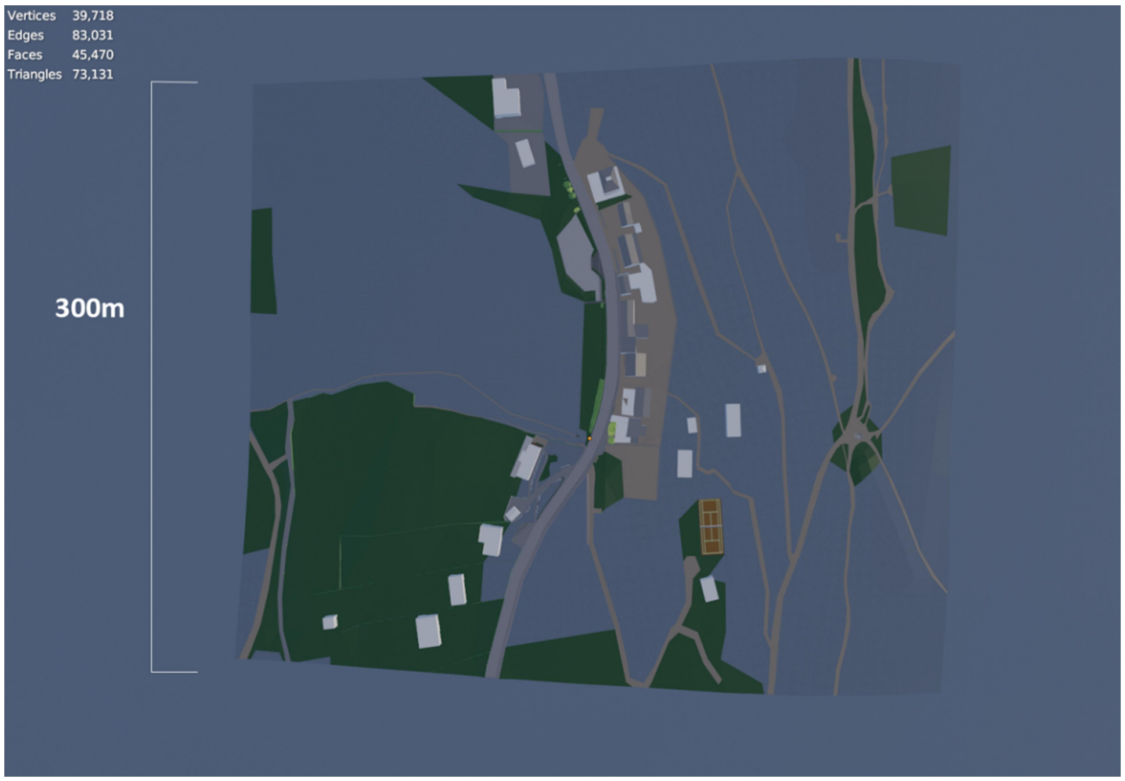
Digital models, to the computer, are all about triangles. The Meta Quest3+ headset can theoretically shift 1.8M triangles @ 120Hz, but to ensure a smooth frame-rate consider capping at 1M @ 90Hz.



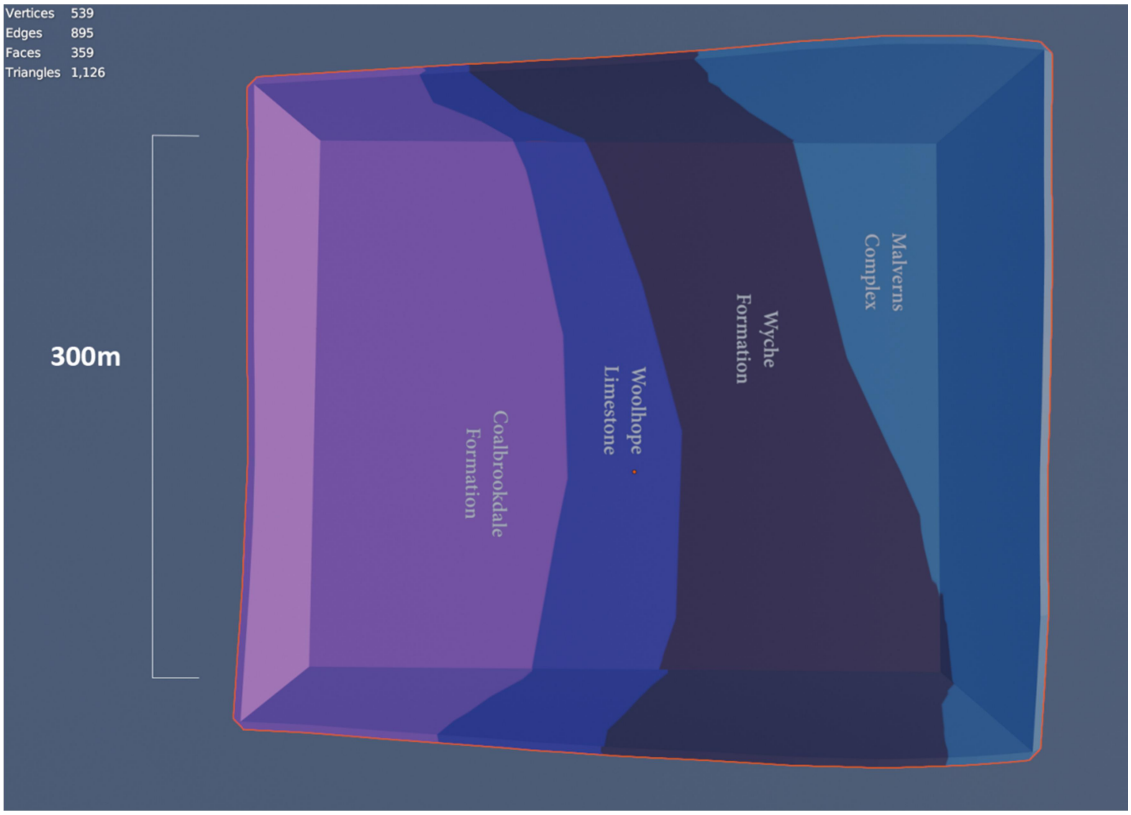
LOD1



LOD0



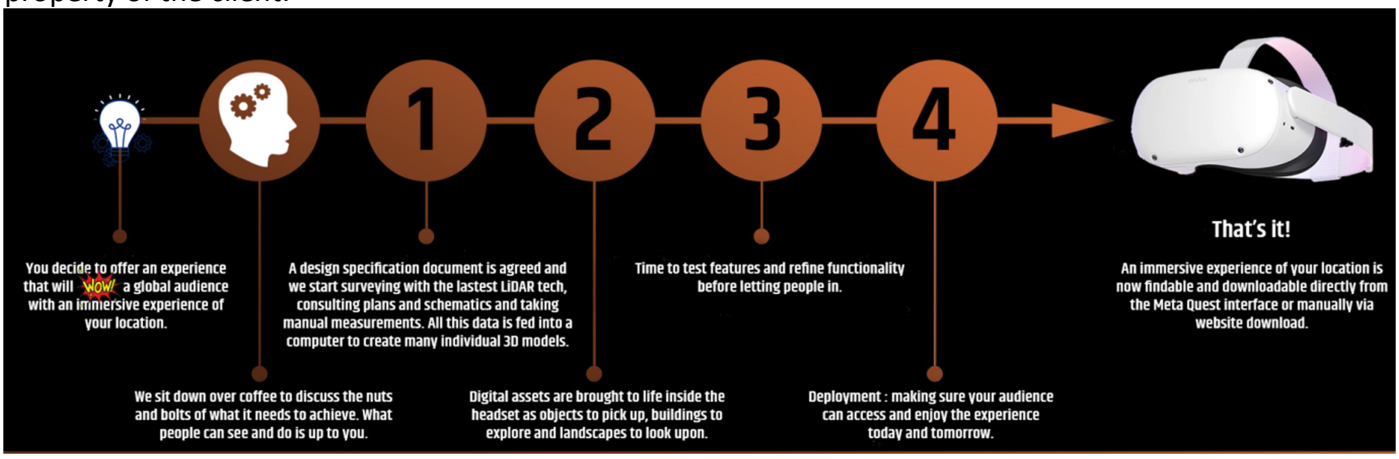
Geological cross-section



(4) Commissions

We Make Worlds accepts commissions based on an agreed Software Development Agreement (SDA).

A 25% non-refundable deposit is due on signing, the balance on delivery. On dispute, a nominated third-party adjudicates on the work being as good as (or better) than the benchmark Immersive Malvern demo in terms of quality and functionality. If found lacking, no balance is payable and the work remains the property of the client.



(4.1) Development methodology

(i) Analysis : Initial meeting and site visit.

(ii) Planning : Creative development of ideas.

(iii) Design : Software Development Agreement with Specifications signed.

(iv) Development : Alpha Build -> Beta Build.

(v) Testing and refinement : Production-ready Build -> Final Build.

(vi) Deployment : Opening the door to virtual visitors.

(vii) Maintenance : Managing a real-time world.

(4.2) Software Development Agreement (draft) 1/5

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

(4.2) Software Development Agreement (draft) 2/5

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							

(4.2) Software Development Agreement (draft) 3/5

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.

(4.2) Software Development Agreement (draft) 4/5

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

(4.2) Software Development Agreement (draft) 5/5

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

(4.3) Specifications (example)

[Client.FirstName]
Any

[Client.LastName]
Person

[Client.Company]
Made Up Company

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

[Short Description of Software]
An immersive experience to showcase a community energy transition scheme on the Malvern Hills, UK.

[Delivery Date]
13th September 2025

[Balance]
£1,500

[Long Description of Software]
Walk, talk and interact within a 1:1 scale recreation of the Malvern Hills, UK. 3 levels of detail (0.1km/2 immediate environment, 1km/2 distance, 10km/2 landscape). Visualization of a community energy infrastructure proposal with interactive controls for resource use/cost for different energy sources. User can move sun's position by programming time of day/year. Day/night cycle with moving stars. Player can control sea-level. Player can control wind speed. Max 20CCU. Tutorial. Presentation tools. Expressive, customizable Avatars.

[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, Linux.

[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 and British Camp should be visible but not navigable. Users can see each other and talk to each other, using their hands to manipulate the environment.

(2) *2 levels of detail (LOD0 : 0.1km/2 immediate environment, LOD1 : 7km/2 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 showcase 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 AND as a CO2e emission to the planet. The input source AND percentage of grid AND local energy can be controlled by the user (coal, gas, nuclear, biomass, hydro, wind/sun). The user can turn 6 household appliances on/off. The user can view 4 stages of local solar and wind energy infrastructure.

(4) *User can move sun's position by programming time of day/year. Day/night cycle. Player can control sea-level. Player can control wind speed.*
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 AND changing the value of solar energy generation in real-time. The user should be able to simulate varying wind-speeds, with similar effects on wind power generation. The sun's position in the sky should be accurate for the observer's latitude and longitude. There should be stars in the sky at nightime that move. 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 whilst this 'free' version of the PUN license precludes public release, limiting this Software to private use, it may be upgraded in the future.

(6) *Tutorial.*
A short tutorial is offered to acclimatize the new user with the headset's features and the use of hand controllers to interact and move around.

(7) *Presentation tools.*
One user can be the 'Presenter'; this user can control what everyone else is seeing AND control whether they can move. This effectively turns the experience into a presentation tool (freeze users -> move them to a new position -> change scenery -> 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 Avatars body and clothing.

[Test Acceptance Criteria]

(1) Max 20CCU.
(a) Build 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 and Herefordshire Beacon from the Wyche Cutting.

(5) Informative '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 to showcase transition schemes.

[Milestones]

12th August	Analysis meeting
14th-18th August	On-site survey
18th August	Planning meeting
19th August	Design meeting
20th August	Development---Programming---networking
22th August	Development---Programming---UI
24th August	Model---LOD1
28th August	Model---LOD0
1st September	Testing---first meeting
3rd September	Development---Programming---alpha build meeting
5rd September	Model---2xLOD completed
5th September	Testing---second meeting
22th September	Delivery
29th September	Deployment
->	Maintenance

(5) Making your world

Technology is a tool we can use together to everyone's advantage. Developing an immersive recreation of your location exposes many opportunities. Learning a modern programming language, 3D modelling and hacking cutting-edge technology are all versatile educational activities. Using industry-leading integrated development environments, you'll mix creativity with mathematics and logic with design to recreate your world and get visitors in.

Any PC with a reasonably modern GPU will run the hardware and software necessary to build extended reality applications.

Training can be delivered by workshops, demonstrations and presentations in person or via Discord screen-share, where we can both see what's on each other's screen. No prior experience necessary other than basic computer skills. There's a steep learning curve for complete beginners but rapid progress is rooted in delivering practical results that are measurable and fun to achieve.

(5.1) Training (sample)

This sample training script represents the first six hours of an online training session delivered via Discord with screen/keyboard/mouse share. It takes a complete beginner through the excruciating process of familiarization with the Unity configuration for virtual-reality application development and introduces a basic environment for working in 3D space. It will serve well to introduce the key concepts and terminology.

Install Unity (latest LTS version). Then, either follow this guide to set up your own development environment from scratch or import the project's source files directly from the public GitHub (https://www.github.com/nukkuepuss/malvern_demo.git). *Ensure that your Meta Quest3 headset can pair with it's desktop application and is in Developer Mode via the Meta smartphone application.*

Part 1 : The development environment

Unity is our real-time 3D development environment, where everything comes together. We can deploy the same project to 27 different devices with Unity, including the Meta Quest3 headset.

(i) Unity -> File -> Build Settings -> Platform -> Android -> Switch Platform -> Android

Unity needs to output a release-ready package in Android format (.apk file - this has the compiled source-code alongside resources, assets, a manifest file and expansion data). This is called a 'build'.

(ii) Unity -> Edit -> Project Settings -> Player

-> Company Name : [who you are]
-> Product Name : [can use a codename]

-> Version : [set here to auto-update elsewhere]

The build system uses package ID attributes to uniquely identify the app in the Meta Quest ecosystem, for building a .gradle file and as a value for the Android manifest file.

(iii) Unity -> Edit -> Project Settings -> Player -> Other

-> Identification -> Package Name : [must conform to Unity Package Manager]
Version : [will auto-update if set above]
Minimum API : [29]
Target : [Highest]

-> Configuration ->Scripting Backend : [IL2CPP]
ARM v7 (x)
ARMv64 (tick)

(iv) Unity -> Oculus -> Tools -> OpenXR on Quest

XR Plugin : Meta has made the OpenXR runtime as default backend.

(v) Unity -> Oculus -> Tools -> Oculus Platform Tool

-> Oculus App ID : [from dashboard]
-> Oculus App Token : [from dashboard]

(vi) Unity -> Project Settings -> Player -> Other -> Rendering

-> Color Space : [Linear]
-> Auto Graphics API : (x) [OpenGLES3.0]
-> Multithread (tick)
-> Low Overhead (x)

(vii) Unity -> Edit -> Project Settings-> Quality

-> Pixel Light Count : [1]
-> Texture Quality : [full res]
-> Anisotropic : [per texture]
-> Anti-aliasing : [4x]
-> Soft Particles : (x)
-> Realtime Reflection Probes : (tick)
-> Billboards Face Camera : (tick)

(viii) Unity -> Oculus -> Tools

Create and understand an Android manifest .xml file.

(ix) Centre Eye Anchor

-> Clip Near : 0.05
-> Clip Far : 7250

(x) Oculus Integration SDK or Meta XR SDK?

Meta recently announced that the Oculus Integration SDK (on which this project is built) is deprecated. Replacement Meta XR SDK are available but untested. Use Oculus Integration and ignore warning messages about deprecation.

OVRManager provides the main interface to the VR hardware.

OVRCamerRig provides the transform object to represent the Oculus tracking space.

OVRManager.cs -> Use Recommended MSAA : (tick)
Tracking Origin : [Floor]
Color Gamut : [Quest]
Skip Unneeded Shaders : (tick)

(xii) XR Plugin Management

Project Settings -> XR Plugin Management -> install plugin

Project Settings -> XR Plugin Management -> Android -> Oculus (tick)

Project Settings -> XR Plugin Management -> Windows -> Oculus (tick)

(xiii) Photon Unity Networking

Create an account with Photon (https://www.photonengine.com). Create a new PUN App (of type PUN) and Voice App (of type Voice). You'll now have an App ID and a Voice ID.

Unity -> PhotonServerSettings -> Server/Cloud Settings -> [PUN+Voice ID's]

(xiv) Meta account -> Developer

The Meta Quest Store is only available as a distribution avenue if you have been approved as a Meta Quest Developer. Meta Quest Developer Hub (MQDH). Requires 'Developer Non-Disclosure Agreement'.

Meta Virtual Reality Check (VRC) : AppLab and Meta Quest Store apps must meet or exceed Virtual Reality Check(VRC) guidelines to be considered for distribution. An in-app reporting mechanism is mandatory.

In order to use Avatars, you need to enable them in the Oculus Developer Portal. Create a new app at https://developer.oculus.com/manage with 'Quest' as the Platform.

Data Use Checkup -> request UserID, User Profile and Avatars. You'll need to enter some reasons why you need this, and a valid link to a Privacy Policy.

You can now find Oculus App ID's from the Oculus Portal for insertion into Unity.

Unity -> Oculus -> Platform -> Edit Settings -> App ID -> [Oculus Rift + Quest]

Part 2 : Introducing 3D space

Follow this guide to take your first steps towards working in 3D space on a 2D screen.

There's four main parts to the Unity interface :

- Hierarchy
- Project
- Inspector
- Scene

Resist becoming overwhelmed by the shock of seeing the complexity of the Unity. 80/20 rule : 80% of the time we'll only use 20% of the functionality. Nothing is on the screen without a good reason. Don't panic.

Resize windows to suit using two-headed horizontal/vertical arrows. Save Layout to suit.

GameObjects : fundamental containers with a Transform. Can have Components attached to give functionality.

Create -> 3DObject -> Cube

Three things happened (i) a cube appeared in Scene, (ii) info appeared in Inspector and (iii) a Cube entry appeared in Hierarchy (and was automatically highlighted blue to show object is selected).

Cube Components in Inspector : Mesh Filter (stores data about shape), Mesh Renderer (draws polygons), Box Collider (invisible; for collisions).

Have a look at the three arrows on the cube in Scene View.

red	x	left/right
yellow	y	up/down
blue	z	forward/backward

We're simulating 3D space in a 2D environment.

Hover over the blue (z) axis and, holding down the LMB, move mouse L/R : the cube moves! See how the Inspector -> Transform -> Position -> z value changes.

Hover over the 'z' and the mouse pointer turns to a vertically to-headed arrow. Hold down LMB and move mouse L/R - voila!

But why are the 'x' and 'y' values all over the place? It's a Unity thing - the first thing to do with a new GameObject is to reset the Transform using Reset.

So, we've got a Unity Scene with a cube… let's introduce the fundamental concepts of World Space vs Local Space, and Parenting.

The Inspector always shows the GameObject's Position in Local Space.

Local Space is the position (and rotation+scale, but let's concentrate on position) relative to any GameObject that it's tethered to, like a watch to a wrist. If you had a watch model and a wrist model, you'd 'parent' the watch to the wrist model. The watch would become a 'child' of the wrist and move with the wrist object without us doing any calculations… nifty and useful stuff.

When an object isn't Parented (as our cube isn't at the moment), Local Space is the same as World Space.

Let's create another cube, move it a bit and Parent the first cube as a Child of the second.

Rename the cube 'cube1' (or similar… it really makes no difference to Unity) and move it along the x-axis a bit. See how the Transform position changes in the Inspector. Make a note of these x,y,z values.

Now create another cube and rename it 'cube2'.

--→ make the cube's different colors

Now, in the Hierarchy, drag'n'drop cube1 on to cube2.

Two things happened… cube1 is now indented underneath cube1 in Hierarchy to show that it's a Child object, and in the Inspector the Transform position has changed to 0,0,0.

That's because cube1 is now at position 0,0,0 relative to it's Parent, cube2. Moving cube2 will not change the Local position of cube1, but it will change the World position. But if an object is Parented, the World position doesn't interest us much, as the Parent is doing all the work (we can still expose World position of Parented objects in code, but that's for the next tutorial).

(5.2) GitHub

The project uses GIT (git-scm.com) distributed source control. This gives each contributor their own complete copy of the entire repository. Developers can add and switch branches and Commit changes whenever they want (a maintainer will merge changes).

Use GitHub (github.com) for access to the Git repository. GitHub provides free hosting for open-source projects as well as a web-based Git repository browser and an issue tracker.

If you are familiar with Unity/C# and you have a bug fix or a feature that you would like to add, the best way to do so it to use the GitHub Pull Request feature:

Pull Requests
When submitting pull requests:
Clearly describe the problem you're solving.
Don't introduce regressions that will make it hard for systems administrators to update.
If adding a major feature, rebase your changes on Master and get to a single Commit.
Include test cases (see below).
Include sample logs (if relevant).
Include a change to the relevant section of the ChangeLog.
Include yourself in THANKS if not already there.

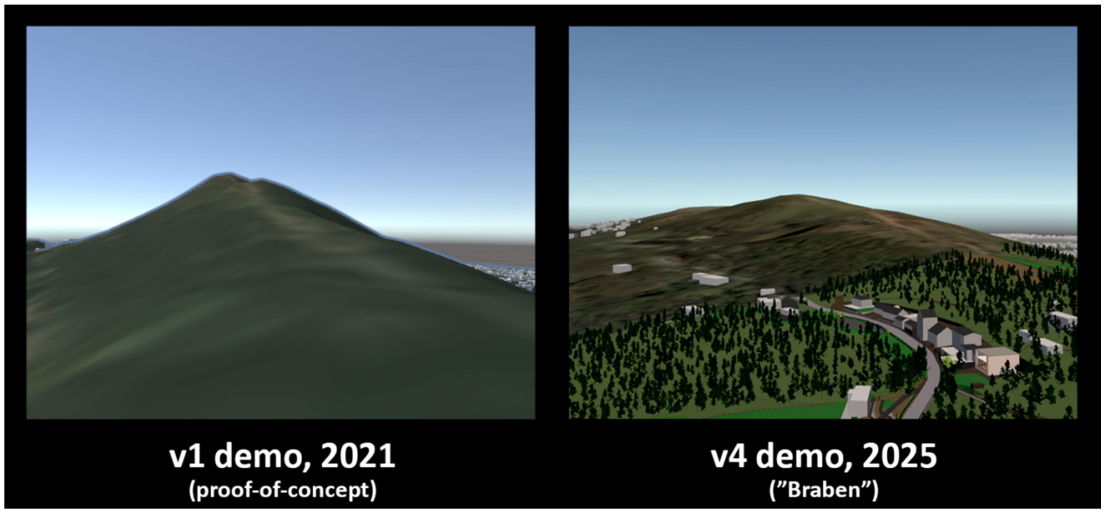
Style
Please use tabs. Keep to 80 columns, at least for readable text. Curly braces are opened on the same line and followed by an empty line. Pascal formatting.

Tests
Add tests. They should test all the code you add in a meaningful way.

Coverage
Test coverage should always increase as you add code.

(6) Credits

Thanks to the giants who made the software and assets that helped turn this idea into content (please contact me if I've left you out), and apologies to those who's ears I chewed off for many years.



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

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

-end-