Practical experiments

Jon Higgins

Virtual and Extended Realities, MA
UWE, Bristol

Eat right with AR

Motivation and concept

Why AR?
Having never worked with AR I was keen to experiment with it. There is a wide variety of tools and production methods so I wanted a project that would allow me to understand the range of options for future projects, rather than focus on a specific platform or AR feature. I also wanted to understand how to prototype AR projects.

Concept ideation
I didn't spend a lot of time thinking about the concept, but I liked how nutritional advice in a running magazine I read was tailored to runners and their goals. I wondered if a similar approach could work for the wider population and their individual health/ wellness goals, and if AR could be used to achieve the concept.

Concept
We all know we should eat at least 5 fruit and vegetables a day, but only around half the UK population manage to meet this target*.

How food choices affect us can be hard to understand. What does choosing an apple over crisps do for me? If we could make the health and wellness benefits of individual foods clearer and more relevant to individuals, would this make people more likely to buy and eat them?

Could AR and personalisation during the shopping experience help people make better food choices?

Inspiration
Inspiration for the concept comes from Candide Labels, now defunct iOS app in which users identified plants via a computer vision machine learning model. The user then placed AR plant labels in their space.

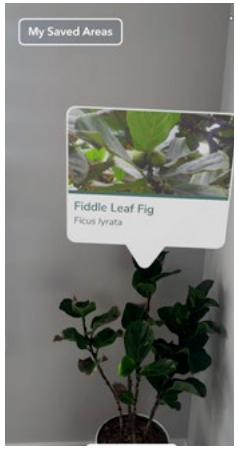


Image credit: AR/VR tips

Prototyping

Prototyping can be used develop and test the concept, indicating the efficacy of the solution and identifying next steps to test.

Paper prototyping
Lo-fi prototyping gives fast feedback, so I started with paper prototyping to sketch out the user journey.

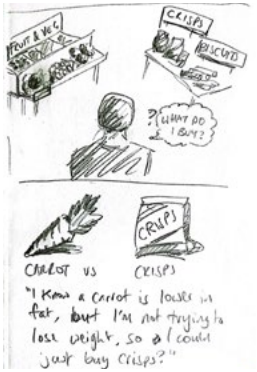
Rapid user testing
Using physical objects was a fast method of prototyping the 3D space. The test proved the basic user flow could work as it was understood by the user. It also highlighted issues

- How would the recognition of foods as objects in AR work? E.g. image recognition or a printed image target on the food label.
- How will the AR overlays be anchored to food objects?
- Would clicking the overlays be hard (the user in the video clicked the carrots themselves rather than the button on the overlay)

Figma for UI design
Figma is great for designing UI, here shown are the AR food overlays. Combining this Photoshop allows placing the UI in faux 3D space for testing out how the UI would render on a phone.

Gravity Sketch for 3D prototyping
I enjoyed designing a food store environment in Gravity Sketch to test the concept. After learning the controls I found it and more effective than sketching out a 3D scene in Unity:

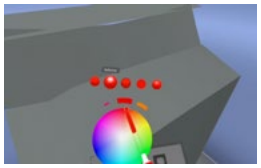
- You are within the 3D space you are designing, meaning: no need to zoom in/out or pan
- Drawing tools allow sketching out basic 3D objects



User is faced with the question of what buy at a supermarket, with an internal monologue showing how non-personalised nutritional information can lead to bad choices.

Sketch of app UI that would enable personalised nutritional information


Stills from user test video. See a video of the user test


See a video of Gravity Sketch prototype

* 54.8% of adults aged 16 and over had 5 or more portions of fruit and vegetables a day ('5 a day') in 2017/18
https://www.ethnicity-facts-figures.service.gov.uk/health/diet-and-exercise/healthy-eating-of-5-a-day-among-adults/latest

Eat right with AR

Testing key mechanic

XR experiences usually rely on one or more key mechanics, it's worth identifying and testing these early. Some parts of the concept are more known to me (the app UI for selecting the users goals) and while the AR interaction of identifying a food and displaying an AR overlay above it is less known to me, therefore I need to prototype this. I thought I'd try this with something that would be quick to test.

8th Wall
Niantic 8th Wall is a platform for building Web AR experiences, so avoids needing to build a mobile app. It's aimed at creative, brand and marketing projects. Features include:

- **Sky target:** use any sky as canvas for your AR experience
- **Image target:** make any image the entry point to your AR experience
- **Location target:** start an AR experience when user visits a specific location, get a 3D model of that location from Lightship VPS to position your experience in
- **SLAM/World tracking**: automatically recognises users' physical space; enabling object placement and physics effects

8th Wall target Options
The first part of the mechanic relies on the user scanning the food object with their phone. Once that object has been recognised correctly, its position will need to be tracked via world or SLAM tracking so that the marker is positioned above the object. To recognise something in the user's viewport, 8th Wall provides the following options:
1. Image target (e.g. a photo, label or QR code)
2. A public location scanned with Niantic VPS
3. Sky effects
4. Face effects

Both 1 and 2 were relevant options. Looking in to #2 I would have needed to visit a food store and scan it to Niantic Wayfarer myself, I was unsure how this worked so it seemed a bit long for the prototype. #1 seemed feasible to test the idea, so I took photos of food objects and adapted an 8th Wall example to place markers above them

8th Wall for testing key mechanic
The process I used for 8th wall was working within their web dashboard, I setup some image targets and modified some example HTML/JS/A-Frame code then published.

Learnings from 8th Wall Prototype

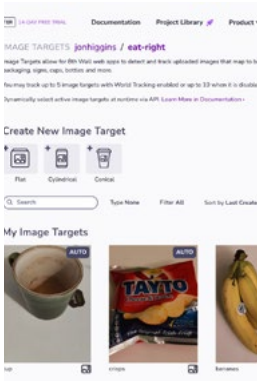
- Image targets not suitable for things that vary visually
- Image targeting works ok-ish in this prototype. Positioning of the marker needs improving
- I need to either adjust what is targeted or method of targeting
- Target method is a core element of AR experience design

8th Wall platform evaluation
Good

- Web-based: easy to distribute and update
- Powerful features
- Low pricing
- Slack community and lots of creative example projects with code

Not so good

- Requires technical skills (HTML/JS/A-Frame)
- Poor layout options
- Accessibility issues



Vuforia Engine/Unity for Cloud Recognition
Due to the issues with 8th Wall's image target I looked at other AR platforms to see if they could achieve the semantic identification of bananas I was looking for. I was also interested in comparing another framework in general. Vuforia is a popular AR platform aimed at large enterprise applications across a variety of industries. There are a lot of Vuforia products and it can be hard to figure out which is relevant, I went with Vuforia Engine marketed as "most widely used platform for AR development, with support for the majority of phones, tablets, and eyewear"

Vuforia Engine supported AR targets
Images and objects

- Image – 2D photos and images
- Cylinder – 3D cylindrical object with image targets
- Multi Targets - 3D cube object with image targets
- Model Targets - 3D object from CAD model
- Cloud Recognition – Same as image target but supports 10k+ images
- Barcode Scanner – Barcodes and QR codes
- VuMarks – QR code type data in more visually appealing format

Environments

- Area Targets - track and augment areas and space of a 3D scanned environment
- Ground Plane - horizontal surfaces in users' environment, such as floors and tabletops

Eat right with AR

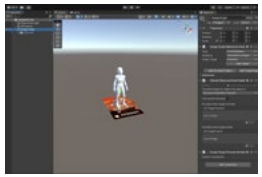
Vuforia Engine/Unity process
Creating a simple example (my version on the right) using Vuforia's sample assets is relatively straightforward if you have some Unity experience: it involves signing up, adding the Vuforia asset to a Unity Android/iOS project and then adding a couple of Vuforia game objects.

1. Sign-up on website and create a license key
2. Setup Unity project to build to Android/iOS
3. Add Vuforia Engine asset to Unity project via Asset Store / Package Manager
4. Add Vuforia AR camera, configure with license key and remove existing camera
5. Add Vuforia Image Target, configure with image asset
6. Add a child 3D object to the Image Target that is shown when target is triggered

Image target issues
When using image targets, Vuforia's computer vision find some images much easier to recognise than others. It helpfully provides feedback in the web dashboard on uploaded images rating them 0-5 stars and showing "features" it identifies in the images. The more features in an image, the easier to identify. The best images are: rich in detail, high in contrast and have no repetitive patterns. My fruit images proved problematic and scored 0 stars.

Good images:
- Rich in detail
- High in contrast
- Not repetitive patterns

Facepalm time / Unity process
I really struggled to get Vuforia to recognise anything other than its example images, even after spending



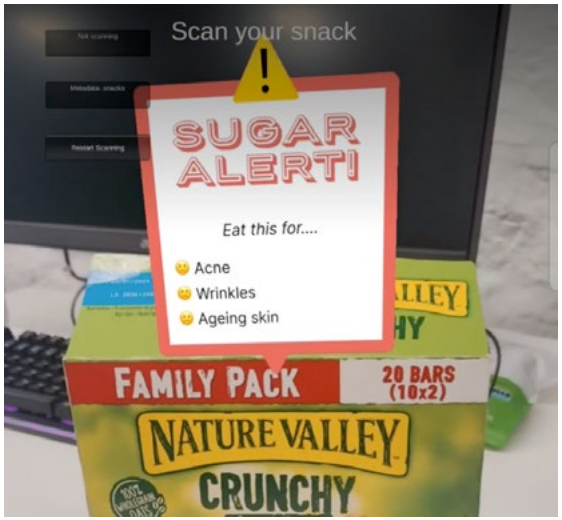
Unity scene with Vuforia Image Target

Built Android app screenshot (using laptop screen to display target)

Image targets in Vuforia, features are indicated in yellow on right-hand side. Bad indicates it wil be hard to recognise this image

time optimising my target images to 5 stars. I tried lots of things for many hours... Until I realised in Unity I was building the Vuforia sample scene to my phone instead of the scene I was editing. Following that I adopted a more methodical Unity approach:
1. Create Unity project
2. Add a 3D cube or primitive to the scene and save it
3. Set build target (e.g. Android)
4. Check it builds to target OK (e.g. test on mobile device)
5. Commit and push to Git

This provides a stable base to make additional changes, repeating steps 4 & 5 after significant changes in the Unity project. This way if there are build errors they can be isolated to the last change.



Vuforia Engine sketch
1. I updated the example to test my concept:
2. Uploaded fruit and snack images to Vuforia target manager. Tagged them with metadata of "banana" or "snack"
3. Switched AR camera to Cloud Recognition
4. Added UI image overlays I made earlier in Figma
5. Added a C# script that displays the UI image overlay if an image target with "banana" or "snack" metadata is found

Unity project: https://github.com/jonjhiggins/eat-right-ar and screencast of prototype

Vuforia Learnings
- An Image Target is not suited to targeting a 3D object, its meant for a 2D image/photo
- Vuforia Cloud Recognition does not have a machine learning aspect (e.g. it cannot semantically understand what a banana is after seeing many photos of one)
- Combining ML image recognition with 3D object recognition is possibly an unsolved problem?

Next steps

If AR and 3D space is important and experience can be limited to a specific store: use Area Targets to place AR overlays above different food sections in the food store

If recognition of food objects is important and it should work in any shop: use 2D image recognition and use 2D UI elements

Although this project did not result in a realisation of the concept, I did improve my knowledge of AR platforms and design. Prototyping allowed me to "fail fast" and also reminded me that some projects do not need 3D/AR to succeed. I would use both 8th Wall and Vuforia Engine again if they suited the project, I am also keen to try ARFoundation for Unity in the future.

Peppa's Park Adventure in Unity and A-Frame

Motivation and concept

Why create a game in Unity?

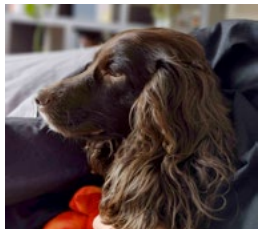
I learnt enough Unity back in 2018 to build a simple VR experience. I really liked the creative possibilities it offered, how it could be adapted to so many uses and how it made complex things like 3D space, physics, spatial sound and FX simple to implement. The mix of a visual editor that could be extended with code suited my preference for visual technical projects. I wanted to use this course as an opportunity to work with Unity again and improve my technical skills, so I had been doing the Unity Learn Junior Programmer Pathway. The first interesting challenge in that pathway was to design your own simple game using the skills you had recently learnt.

Concept

The project design doc (available on Unity Learn) was really useful in helping think about different elements of gameplay, it is somewhat prescriptive which is useful at this stage to limit the scope of the idea, but still allows some creative freedom. It would also be useful as a framework to critique other games so you can understand which parts contribute to it and consider how well each element works. The document makes you decide on:

- Player control
- Basic gameplay
- Sound and effects
- Gameplay mechanics
- User interface

For my project I decided based it around a dog called Peppa, who we had recently been looking after, and how she loves the park, chasing balls and sticks and is totally disinterested in other dogs. A simple concept with easy to build gameplay and aesthetics, to gain a broad but shallow knowledge of Unity games dev that can later be built on.



Project timeline

The design doc makes you define project milestones. This can be hard to do when you are unsure how long things will take, but if you don't get too caught up on the timings it is a useful exercise in understanding the priorities of the project and a guide during the project to understand how you are progressing - invaluable for making sure you don't get stuck or too focussed on one element.

Unity vs web projects

Having worked on web apps for a number of years I have a good understanding of how a project or new feature might be planned. This experiment helped me begin to understand the differences between approaching a Unity versus a web project (e.g. camera angle, gameplay, game mechanics, sound).

Design doc missing elements

Unity Learn's design document is for a short project so is missing elements such as:

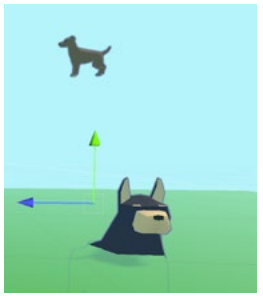
- Visual and sound aesthetic
- Target audience
- Target platform (VR / AR / WebGL

Building the project

Milestone #1

Prototype assets in place

poly.pizza has a large collection of free, simple 3D assets, where I was quickly able to find the dog and ball assets I was looking for. I was unfamilar with 3D asset formats, but .obj file imported to Unity fine as long as the supplied .mtl material file was included. When placing them in the scene the assets came in at different sizes and more problematically, when setting them to the same position values in Unity they appear at totally different positions. I didn't understand about how to best import 3D assets into Unity but I decide to manually resize and position the objects and move on.



Move dog left to right, balls flying, bag dogs moving

When testing out the experience I decide I want to make the bad dogs run towards our Player (Peppa the dog), which is simple using Unity's "transform. LookAt", however my manually transformed and positioned assets make a mess of this, with the dogs running in the wrong direction. I can't figure out why, but if I switch out the assets for primitive cubes it works fine, which suggests its worth sketching out movement with primitives first.

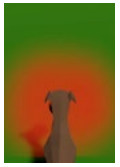
Display score at top

I didn't know how to achieve this so I researched via Google. I started using an unofficial YouTube video, but eventually found it to be out of date. After looking again I found a better tutorial on Unity Learn. For something that would require a tutorial my first point of call is now Unity Learn as I find these are the best resources if they are up-to-date.

Peppa's Park Adventure in Unity and A-Frame

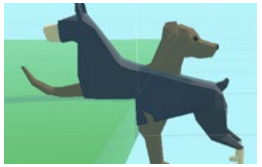
Milestone #2

Catching a ball increments score, hides the ball, particle effect and sound



The "catching" is just a collision between two GameObjects (ball and Player), so they need a Rigidbody adding to them, then the ball has a "OnCollisionEnter" method that checks if has collided with our Player GameObject. Hiding the ball is easy by destroying the game object.

Tangent #1: Fixing imported 3D assets dimensions



Looking to fix my earlier issues with imported 3D asset sizes I couldn't find resources online on the best way to deal with the issue so I came up with my own approach which seemed to work. I used Blender to resize the models to a predictable $1m^3$, rotating them the same way and reseting their origins.

Adding particle effects

Particle effects can produce some very impressive effects. While I found them quick to add in a collision, I struggled to create anything visually compelling. There are a lot of controls in the particle effects and this warrants more time than I have for this experiment so I move on

Adding audio

I follow a Unity Learn tutorial on audio and applied it to my scene, adding a mixer to control sound groups (atmos, music and FX) then I added sounds from freesound.org and set certain sounds to play on collision via a C# script. After understanding the process it was straightforward. Adding spatial audio to the bad dogs really improved realism as their barks become louder as they approach.

Add an intro screen

I realised at this point I didn't have much of aesthetic, while I didn't want to go too far with this I decided it would be good to quickly research one and design something outside of Unity. While I could have designed in Unity, stepping out of it into Photoshop made me feel less constrained. Unity feels more like a layout than a design to me currently. For the next project I would spend time outside of Unity moodboarding and sketching with other tools.

Publish to WebGL

A Unity WebGL build will work in modern web browsers, including mobile, making it easy to distibute. You can also publish freely and instantly to Unity Play to host your experience, which is a useful tool for sending links to other people and devices for testing. I planned to publish to WebGL/Unity Play at the end of the project. On reflection I would now do this step much earlier and more frequently (e.g. at least every milestone), that way you know your project is building successfully rather than having a lot of issues to sort out at the end of the project.

Disappointingly it didn't work on mobile - Unity Play blocks mobile usage. I later found out this is a control you set on the Unity Play website. As there is no keyboard input on mobile it wasn't possible to control the player. If I wanted to support mobile I could try mapping the accelerometer or touch to the controls. It was at this point I realised the interactions wouldn't easy port over to VR, so there is a learning about deciding on target platforms early on. Unity Play also starts with audio off, and requires the user to turn it on, which may be an issue for some projects.

Peppa's Park Adventure on Unity Play
Video of gameplay
Unity project on Github

Build to Meta Quest 2

To change the interactions to work in VR I needed to build to a VR headset. I had a frustrating time trying to do this; my Mac had slow builds and the university PCs were not setup to build VR. So I abandoned this in favour of A-Frame VR. As I had been working on a Git branch for the VR work I was easily able to switch back to the main branch for a working version.

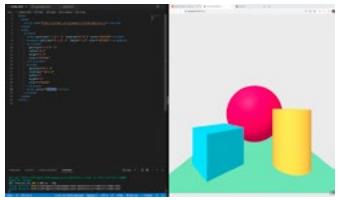
Evaluation

This experiment fulfilled the goal of getting a broad but shallow understanding of Unity games dev. While there were some frustrations, it was fun to build and I still see great potential of Unity as a tool and plan to continue working with it. The unfinished game was very poor in terms of gameplay and aesthetic, which shows it takes significant time to produce a game because of all the various elements involved.

Peppa's Park Adventure in Unity and A-Frame

A-Frame

A-frame is a web framework for building 3D/AR/VR experiences. It is a HTML entity-component system (similar concept to Unity Game Object and components) which has simple building blocks for XR experiences. It powers the 3D functionality in Mozilla Hubs.



Device and platform support
A-Frame extends three.js which means the experiences can be run in any web browser that supports WebGL (all modern browser), it also supports popular VR hardware such as Meta Quest 2. Being web-based avoids the distribution and development friction associated with native apps. It doesn't have a dedicated publishing solution like Unity Play, but similar generic web services exist (e.g. Netlify, Vercel).

Editing experience
While a visual inspector exists for editing scenes, A-Frame requires HTML skills. This code is simpler than Unity as it is based around simple declarative attributes, e.g. I want a red box, 1m^3: <a-box color="red" size="1 1 1"></a-box>. Going beyond basic functionality requires technical knowledge in understanding the documentation and custom functonality requires more complex Javascript and three.js knowledge.

Documentation and support
The A-Frame website has reasonably detailed documentation. However, much of it appears to be an autogenerated API spec that lacks detail of what various aspects do. The tutorial is very basic, particularly compared to Unity Learn. The website feels dated, looking at GitHub it appears the project is still in active development, but releases over the last few years are for supporting new hardware and bug fixes, rather than improving functionality. Industry support is unclear, it previously being a Mozilla project but now seems to be supported by agency Supermedium. The future for WebXR seems unclear, with no big-tech backing following Meta's withdrawl of React360.

Limitations
Unity builds to native apps that have low-level access to device features not available to WebXR, such as sensors, bluetooth and access to the file system. Unity also offers higher performance graphics and much higher quality particle effects and physics.

Rebuilding Peppa's Park Adventure in A-Frame

To learn about A-Frame and compare it to Unity I tried rebuilding my Unity game in A-Frame. Getting a development environment was quick, it just involves including the A-Frame JS file. I completed the short tutorial to get an idea how A-Frame worked and then began sketching out the main objects in the scene by taking their position, scale and rotation values from Unity.

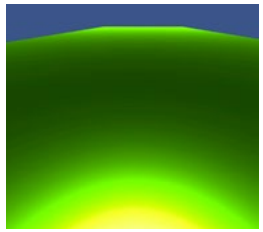
Some of A-Frame's built-in functionality makes creating faster than Unity, for example the camera primitive comes with built in user keyboard controls, whereas in Unity this requires custom scripting. The built-in functionality can get in the way at times, for example I couldn't control the camera position in JS without turning off the default WASD/look controls behaviour which took me a while to figure out.

Same, same but different
Adding elements to the scene is quick, but differences between the platforms consumed much of my time. Materials and lighting behave differently to Unity, after experimentation and I still unable able to match the colours across the experiences. Unity uses a left-handed co-ordinate system while A-Frame's is right-handed. As such I have to negate position values on the Z axis, so Z of 3 in Unity would be Z of -3 in A-Frame. When rotating I have to negate the X axis to achieve the same effect, I'm unsure why, a better knowledge of 3D geometry would be useful .

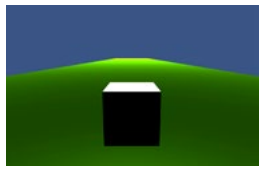
Customising beyond A-Frame out of box features
The entity-component system is easy to extend (add a custom attribute to an element and define it as a component in a JS file) and similar to Unity. So bringing over features such as camera tracking and bad dogs running towards player was just a case of rewriting C# to JS. However, trying to modify the keyboard controls /player movement felt much harder in A-Frame, the mix of A-Frame/three.js further complicating things. I had to keep the standard WASD controls, which feel very clunky as the player is not rotated in a direction. I also tried implementing physics via cannon.js, which felt harder than Unity's inbuilt physics.

A-Frame could be a good prototyping tool or suited to projects that match its out of the box features. It would also be good for creators who find Unity too bloated or want to advance from Mozilla Hubs/Spoke. As ever, sketching out key mechanics early would give an indication of the platform's suitability to a particular project.

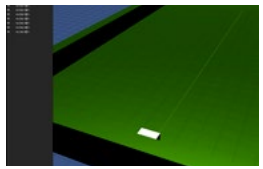
A-Frame experience
A-Frame project on GitHub
Video of A-Frame gameplay



Colours and lights render differently in A-Frame



Understandng the different co-ordinate system in order to position accurately



The inspector is useful for checking and modifying properties

A-Frame supports .obj and .glb 3D models. Complex models may need simplifying for performance

AI text and image generation

Motivation and concept

As AI tooling has recently become much more powerful and accessible I wanted to see how some of these tools could be applied to XR projects. As AI impacts more of our lives I wanted gain a better understanding and form a critical opinion on their usage.

Text content generation via large language models

A large language model (LLM) is a neural network with billions of weights that is trained unsupervised on vast amounts on unlabelled texts that can be used for computational linguistic tasks. Put more simply, it is a self-trained, intelligent computer "brain" that has read and understood a huge amount of text from a wide range of sources and is able to use that to complete language-based tasks such as answering questions, summarising text, sentiment analysis and translating between languages. While we are only beginning to understand the many applications of such technology, I thought I would try out a few possible use-cases in XR using the most popular LLM: OpenAI ChatGPT.

Project idea generation

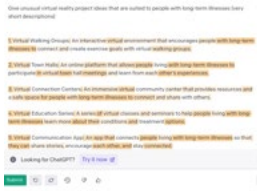
Using the prompt "Give me 3 virtual reality project ideas" ChatGPT responds with 3 good suggestions of "VR Travel Guide", "VR Training Simulator", "VR Therapy" with well written descriptions. The ideas are obvious applications of VR with generic descriptions; unspecific prompt tends to lead to popular or generic answers. ChatGPT allows refinement of answers using additional prompts, so I could follow up with "Suggest more unusual applications, that would be suited to people with long-term illnesses". The result of that prompt was still fairly generic. OpenAI's developer Playground allows some more fine-grained controls of the model, for example the "Temperature" can be changed to effect the randomness of the result and "Show probabilities" can help understand which words the model was most likely to return, which could be used to debug or adjust your prompt. The results are still too generic to use, however it is useful for researching elements that might make up a project idea, e.g. "Identify technology challenges for people with long-term illnesses."

Script or narrative generation

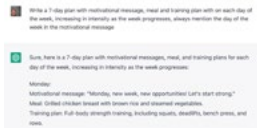
In the Adaptive Podcasting workshop I needed to quickly get some content for my idea of an exercise and nutrition training plan. Using the prompt "Write a 7-day plan with motivational message, meal and training plan with on each day of the week, increasing in intensity as the week progresses, always mention the day of the week in the motivational message" gave a good but generic training plan, quickly giving me the content I needed to prototype with. I found the response more interesting when I added "in the style of a 16th century playwright", which is example of how extra prompts can result in more creative responses. It could also be used to take the same content and personalise it for different audiences. While I have been using OpenAI via its web dashboard it is also available via API so can be integrated into other apps or XR experiences.

Image generation

Deep learning models can generate images, such as OpenAI DALL-E, Craiyon and Midjourney. I worked with Midjourney as it had the best image quality. It works through Discord (chat app), which initially added friction to the process but the social aspect of seeing other's prompts and images inspired me to try different things with the images I generated.



Higher probabilities of words occurring highlighted in OpenAI playground



No style specified

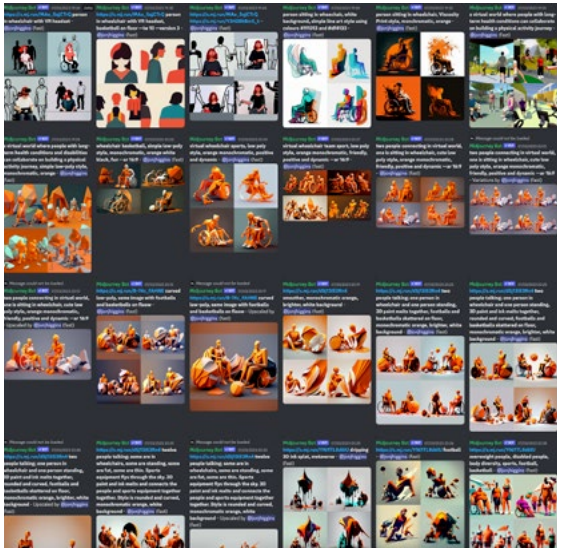
Rewritten with style specified

Final image edited in Photoshop

Concept art

I generated concept art for a social VR experience to encourage people with long-term conditions become more physically active. I had a hand-drawn aesthetic in mind, so I supplied an existing image. Midjourney wasn't able to copy this style, so I tried different aesthetics before settling on low-poly. The characters were interesting but angular and unfriendly, Midjourney can remix earlier images so I used this to smooth out the figures. It gave some interesting results at the end but it used the wrong colour palette, so I reworked the image in Photoshop. Although it required manual work at the end Midjourney produced better results than if I was to draw it or use stock images and provided creative options.

I supplied this image to Midjourney to copy the style of. Image credit: Nifty Fox Creative



Developing the concept art using Midjourney

AI 3D asset generation

While Midjourney could be used for 2D images that could make up a 3D object (e.g. textures) I was interested to see if 3D assets could be generated via AI.

Using ChatGPT to generate 3D object code
I'd previously had good results using AI code generation for JS and C# scripts, so I tried prompting a LLM for 3D asset code, so I started with "Generate geometry definition in OBJ format for a cube" which produced a good result and explained the file format. Building on that I retried the same prompt but asked for a small dog, which returned valid OBJ code but looked nothing like a small dog, suggesting this is not the right platform for the job. This was confirmed by asking ChatGPT how well it could generate OBJ geometries from text prompts:

"As an AI language model, I can generate OBJ geometries from text prompts to some extent, but the quality and accuracy of the resulting geometries will depend on the complexity and specificity of the prompt, as well as the limitations of my training data. [..] for more complex shapes or models, I may require more detailed and specific information about the shape, topology, and connectivity of the model."

AI tools for generating 3D objects
There doesn't appear to be an established AI tool for generating 3D objects, but there are several research projects which show promise: Google's DreamFusion, NVIDIA's GET3D and OpenAI's Point-E. These aren't as easy to access as ChatGPT but can be run either on local or cloud computers with high-powered GPUs via Python command line or notebook app. I decide to look at Point-E, which takes text and image prompts and returns 3D point clouds. Point clouds can be converted into meshes, saved as .ply files and imported into Blender. The examples of a corgi dog and red motorcycle look pretty good. I find a Google Colab project that uses Point-E, copy it into my workspace, fix some path errors and change the text prompt to "a small brown dog". The result is not very compelling, an odd creature of sorts. The text prompt version seems to use a smaller, lower quality model (4 million parameter), whereas the image prompt can be used with a larger model (1 billion parameter), so I try that instead. This takes much longer to render, the point cloud looks promising but unfortunately the meshes are very poor in Blender. There are many steps to this process, which I have little knowledge of, so its hard to debug. Point-E could be used to generate some strange creatures, but ultimately it was a disappointing experiment. However, I learnt some new ways to represent and work with 3D geometry that may be useful in the future, as well as work with AI models.

ChatGPT successfully generated a cube in .obj format

A very abstract interpretation of a small dog from ChatGPT

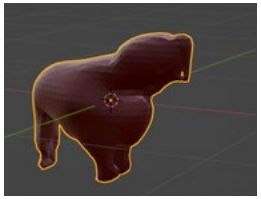
AI issues and ethics

AI thought process is not usually transparent, so it can be hard to understand why AI has given a certain result. AI models need to be trained on large amounts of data, making it hard to have oversight of its contents. Issues with training data can include bias, discrimination and infringements on privacy and copyright. Where AI is using personal data, privacy laws need to be adhered to. These issues should be considered when using AI tools in future projects.

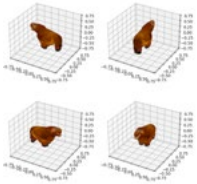
AI tools evaluation

While there are some incredibly impressive tools in this space my experiments show some are still in early development while others can improve productivity but should be used with care. Currently I find AI is best used as a muse to develop ideas, rather than produce final outcomes.

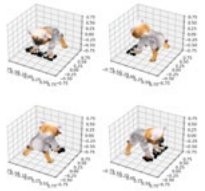
Point-E generated Corgi mesh from their example image of a cartoon Corgi



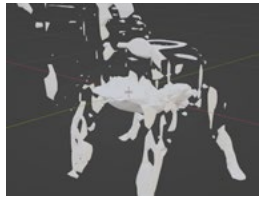
I asked Point-E to generate "a small brown dog"



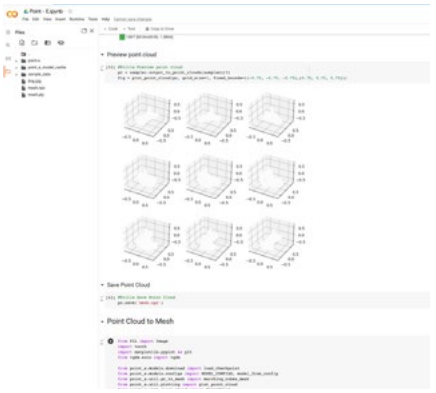
The point cloud of "a small brown dog"



I provided Point-E with an image of a pug dog, the point cloud was not too bad



Unfortunately the pug's mesh was very poor when imported to Blender



Google Colab workbook running Point-E

Web audio API

The Web Audio API allows web browsers to control audio, including:

- choosing audio source (e.g. sound file, microphone or oscillator that creates sound)
- add effects (e.g. echo or spatial effects such as panning) to audio
- create visualisations from audio

It has good browser support and has been around for nearly 10 years.

I wanted to experiment with audio as I have not worked with it before and I think XR provides some interesting opportunities to use audio as an input and an output.

How does it work?
The API handles audio operations within an **audio context** which contains **audio nodes** linked together in an **audio routing graph**.

I created a basic example that creates a sound using an oscillator audio node (https://codepen.io/jonjhiggins/pen/XWPvWYG) which I found relatively easy to do, but customising the sound involved low-level customisation of the sound - e.g. the musical note played is defined by a numeric frequency.

Tone.js

Tone.js is a framework that simplifies working with the web audio API, I used it to make a basic keyboard with option to change the instrument and drum machine loop (https://codepen.io/jonjhiggins/pen/poOMvgY).

This involved much less code than using the Web Audio API directly and did not require me to find out how to create the various instrument sounds, which are a mix of different oscillators and effects (detailed in the Tone.js documentation). It also simplified changing the musical note (you can just pass it a text value, e.g. "C4") and handling accurate time. I would likely use Tone.js over the Web Audio API directly as its faster and easier to use, with good documentation, community examples and add on effects and instruments. The trade-off is the additional size of the asset (75kb GZIP), maybe taking an additional 1.5s to download on 3G network, and may be slow to parse on low end devices in a complex web app, but this would likely be fine for most projects.

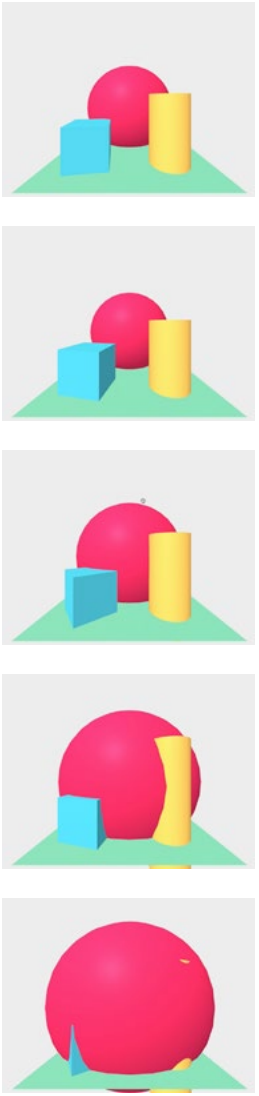
I also experimented with Scribbletune which allows generative creation of music via patterns, chord progressions and arpeggiators: https://codepen.io/jonjhiggins/pen/yLxmbWo

Using movement to control audio
Next I looked at using movement data to control audio. For movement input I used face recognition from a camera. I accessed the users camera as a video source using the web MediaSource API then used ml5.js, which is built on TensorNet, to get the user's face position from the video stream. ml5.js returns a bounding box relative the video source size which might be hard to use predictably as input values to control audio. So I took the X and Y centre points of the users face and normalised them in relation to the video stream's height and width, meaning when the user's face is on the far left of the screen it will return a value of 0, and on the far right it will return a value of 1. Normalising is a useful technique when converting between inputs/outputs with different bounds. I then added Tone.js to play a synth, and connected to the video so that the user's head movement on the X-axis controlled the musical note played, while the Y-axis controlled a distortion effect. Testing a 4:3 webcam the X-axis worked well, but the Y-axis is harder to move on, I tried adding easing to amplify the effect, but it didn't work. It might work better moving the head backwards and forwards, using the face height value. Testing on a phone's selfie camera the X-axis is very narrow. https://codepen.io/jonjhiggins/pen/MWqWrRr

Video of face tracking

Using audio to control visuals
Finally, I used Tone.js to connect to the user's microphone and get the current level (volume) as a normalised value. I then added in a basic A-Frame 3D scene and used the microphone level value to control properties of the 3D objects in the A-Frame scene. I like the playfulness of this sketch, when playing music through the microphone the sphere expands and engulfs the other objects. Tone.js has more advanced audio analysis tools that could be used in visualisation, while there are many possibilities for how these audio properties could be rendered in A-Frame. https://codepen.io/jonjhiggins/pen/JjmjMoV

I enjoyed this experiment and like how audio and 3D objects are represented by values that can be mapped to each other, creating interesting effects.



Video of microphone to A-Frame

Links

Eat right with AR

8th Wall web app prototype video
https://uweacuk-my.sharepoint.com/:v:/g/personal/jon_higgins_live_uwe_ac_uk/Eb-TOe4uX-SZKjoCx7HqkCncBQXg_ufQKhxnh1HIgO79pPw?e=G4706m

Gravity Sketch prototype video
https://uweacuk-my.sharepoint.com/:v:/g/personal/jon_higgins_live_uwe_ac_uk/Ea1BBJtg2WhCsgo0FZsWd4ABL6EpMfsWsRcE_sptntCcdw?e=fpIWtq

Vuforia Engine Unity prototype code
https://github.com/jonjhiggins/eat-right-ar

Vuforia Engine Android app prototype video
https://uweacuk-my.sharepoint.com/:v:/g/personal/jon_higgins_live_uwe_ac_uk/EQFfQWpP7sxArQaT4tbOjQ4BXrrfmPLHP2Iyb2LKP5WVGg?e=CLJfhI

Peppa's Park Adventure in Unity and A-Frame

Peppa's Park Adventure on Unity Play
https://play.unity.com/mg/other/build-hz8

Video of gameplay
https://uweacuk-my.sharepoint.com/:v:/g/personal/jon_higgins_live_uwe_ac_uk/EZ8mbFVJ37ZNtGrkPku2OGsBd6mizPic9mN8tkj41_qvLA?e=qEeOfh

Unity Project on Github
https://github.com/jonjhiggins/peppas-park-adventure

A-Frame experience
https://peppas-park-adventure-a-frame.netlify.app/

A-Frame project on GitHub
https://github.com/jonjhiggins/peppas-park-adventure-a-frame

Video of A-Frame gameplay
https://uweacuk-my.sharepoint.com/:v:/g/personal/jon_higgins_live_uwe_ac_uk/EeX-bo82z3E1AkFsFu194fToBYLn0185psqUigVQByNDIg?e=21rtuP

Web Audio API

Basic web audio API example
https://codepen.io/jonjhiggins/pen/XWPvWYG

Tone.js keyboard / drum machine
https://codepen.io/jonjhiggins/pen/poOMvgY

Scribbletune chords and arpeggiator
https://codepen.io/jonjhiggins/pen/yLxmbWo

Using movement (face tracking using ml5.js) to control audio
Codepen: https://codepen.io/jonjhiggins/pen/MWqWrRr
Video: https://uweacuk-my.sharepoint.com/:v:/g/personal/jon_higgins_live_uwe_ac_uk/EXZXsStMYSpHsmYmGowHDaYBiiZVg-XCopMynpGRTPFsog?e=xQ0W1Y

Using audio to control A-Frame visuals
Codepen: https://codepen.io/jonjhiggins/pen/JjmjMoV
Video: https://uweacuk-my.sharepoint.com/:v:/g/personal/jon_higgins_live_uwe_ac_uk/EQcmbpujjWFLpLtKFf2X4-sBof2Mg2Y9_osrhZ_OkFzp-g?e=80fqTz