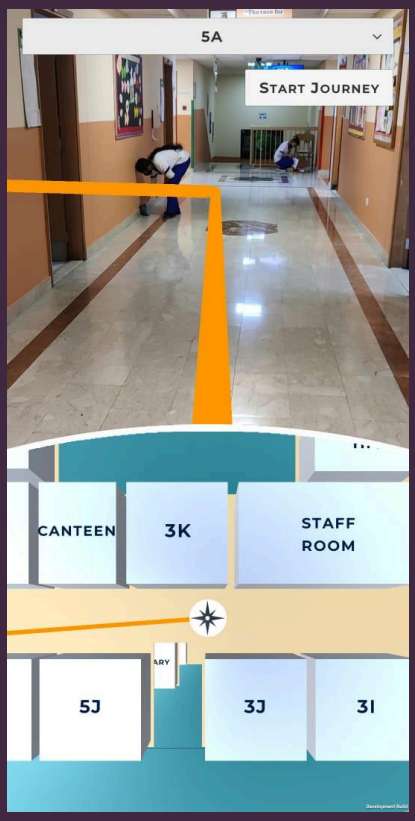
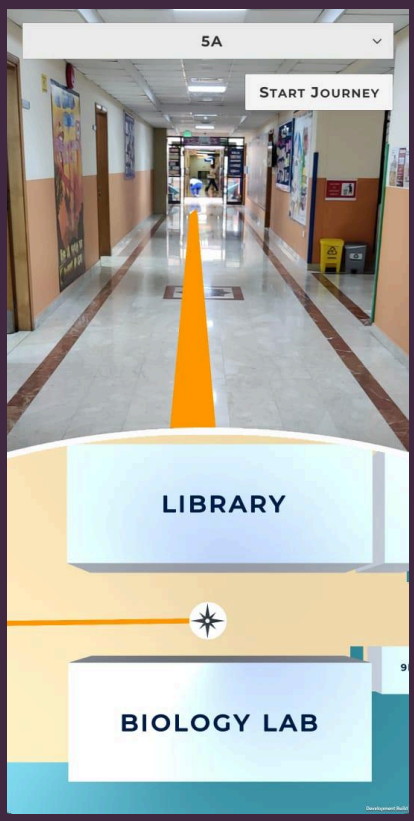
The Locus Navigator

What is Locus?

Locus is DPS Dubai's navigation application that will help you get to your selected destination from any point in the school.

The Locus' User Interface





The black box in the top half will be the image rendered from your phone camera.
The orange line is the path that directs you to your class. It is visible as a real-time rendering as well as a minimap projection.

Locus Navigation Manual

Functionality

- Find a QR code. There's one posted up at the entrance of every staircase.

- Do not scan the code from an angle. Stand directly in front at a distance of approximately 40cm.
- Ensure that the code is placed in the middle of the screen.
- After scanning, the AR session should reset.
- Rotate the tab vertically downward and hold it in a natural position.
- Enter your destination and click start journey.
- Do not rotate and move at the same time.

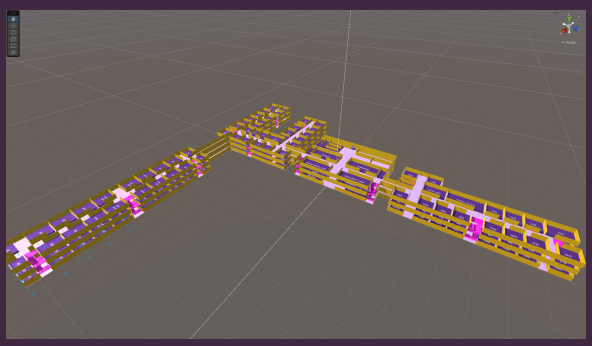
Precautions

- Try not to shake your phone abruptly as this can disorient the phone's accelerometer.

Troubleshooting

- If the minimap goes blank, move the device a bit lower.
- If the line points directly at the wall, rescan the code.

Development



Locus was built using the Unity Development Engine and was coded primarily in C#. The modules used for Android were AR Foundation and ARCore. After a scale model of the school was replicated in Unity, navigation targets were assigned to each class. After scanning the QR code, a line is rendered from the user's origin to the destination selected by them, thus enabling indoor pathfinding.

```
QrCodeRecenter
private void OnEnable()
{
    cameraManager.frameReceived += OnCameraFrameReceived;
}

private void OnDisable()
{
    cameraManager.frameReceived -= OnCameraFrameReceived;
}

private void OnCameraFrameReceived(ARCameraFrameEventArgs eventArgs)
{
    if (!cameraManager.TryAcquireLatestCpuImage(out XRCpuImage image))
    {
        return;
    }

    var conversionParams = new XRCpuImage.ConversionParams
    {
        inputRect = new RectInt(0, 0, image.width, image.height),
        outputDimensions = new Vector2Int(image.width / 2, image.height / 2),
        outputFormat = TextureFormat.RGBA32,
        transformation = XRCpuImage.Transformation.MirrorY
    };
    int size = image.GetConvertedDataSize(conversionParams);
    var buffer = new NativeArray<byte>(size, Allocator.Temp);
    image.Convert(conversionParams, buffer);
    image.Dispose();
    cameraImageTexture = new Texture2D(
        conversionParams.outputDimensions.x,
        conversionParams.outputDimensions.y,
        conversionParams.outputFormat,
        false);

    cameraImageTexture.LoadRawTextureData(buffer);
    cameraImageTexture.Apply();
    buffer.Dispose();

    var result = reader.Decode(cameraImageTexture.GetPixels32(), cameraImageTexture.width, cameraImageTexture.height);
    if (result != null)
    {
        SetQrCodeRecenterTarget(result.Text);
    }
}

private void SetQrCodeRecenterTarget(string targetText)
{
    Target currentTarget = navigationTargetObjects.Find(x => x.Name.ToLower().Equals(targetText.ToLower()));
    if (currentTarget != null)
    {
        session.Reset();

        sessionOrigin.transform.position = currentTarget.PositionObject.transform.position;
        sessionOrigin.transform.rotation = currentTarget.PositionObject.transform.rotation;
```

```
us Files
SetNavigationTarget
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.AI;

public class SetNavigationTarget : MonoBehaviour
{
    [SerializeField]
    private TMP_Dropdown navigationTargetDropDown;
    [SerializeField]
    private List<Target> navigationTargetObjects = new List<Target>();

    private NavMeshPath path;
    private LineRenderer line;
    private Vector3 targetPosition = Vector3.zero;

    private bool lineToggle = false;

    private void Start()
    {
        path = new NavMeshPath();
        line = transform.GetComponent<LineRenderer>();
        line.enabled = lineToggle;
    }

    private void Update()
    {
        if (lineToggle && targetPosition != Vector3.zero)
        {
            NavMesh.CalculatePath(transform.position, targetPosition, NavMesh.AllAreas, path);
            line.positionCount = path.corners.Length;
            line.SetPositions(path.corners);
        }
    }

    public void SetCurrentNavigationTarget(int selectedValue)
    {
        targetPosition = Vector3.zero;
        string selectedText = navigationTargetDropDown.options[selectedValue].text;
        Target currentTarget = navigationTargetObjects.Find(x => x.Name.Equals(selectedText));
        if (currentTarget != null)
        {
            targetPosition = currentTarget.PositionObject.transform.position;
        }
    }

    public void ToggleVisibility()
    {
        lineToggle = !lineToggle;
        line.enabled = lineToggle;
    }
}
```