<AR NAVIGATION GAME IN MUSEUM >

ABSTRACT

Museum nav is a museum exploration game using on Unity's AR Foundation components, as well as two basic theories: "how to create the surrounding environment in which we may hold the device" and "integration of entities with the surrounding content."

Combined with user testing, we will investigate the possibility of incorporating 3D sceneries into AR design, from Augmented Reality to Augmented Virtuality [1].

TECHNICAL PART

The idea behind the design is to use real exhibits as clues, then cover the surrounding scenes with 3D models (walls, characters, and buildings with the same theme as the exhibits) to give users an immersive experience, allowing them to click on the screen and connect to other exhibits in the same historical context (figure 1).

1.1 NAVIGATION PART

1.1.1 PROCEDURES FOR THE OPERATION

The user opens the app, navigates to the closest exhibit based on their location, and then creates an immersive experience by overlaying the 3D model on surrounding objects. A red navigation line shows when the user clicks on the screen (middle part in left figure 1) to take them to the exhibit (blue cube in right).

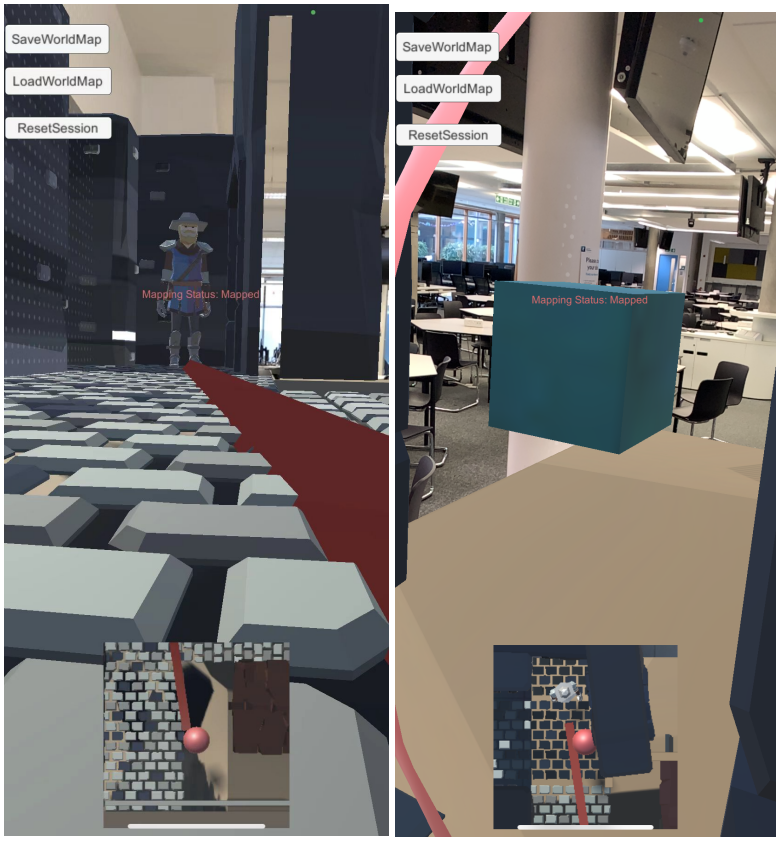

Figure 1. Scene and blue cubes

1.1.2 INDICATOR, TOP-DOWN MAP, LINE RENDER

We integrate camera and render texture to generate a top-down map, then apply position constraints to update the position changes on the Raw picture in real-time.

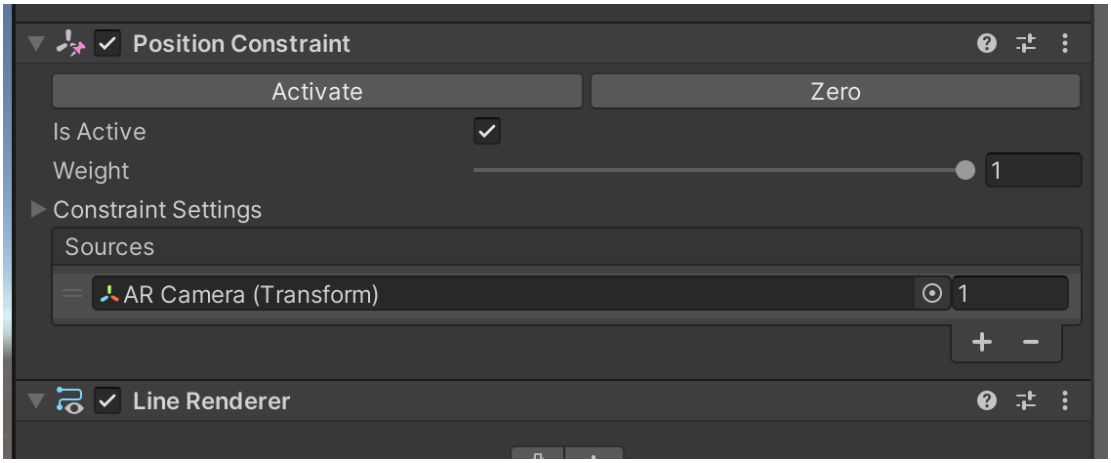

Figure 2. Position Constraint

Calculate Path: File **Navigation.cs**, line 35 utilises Navmesh, the blue region in the figure 5, to collect the navigation data between the target and camera, and the result is provided to line render, which generates the appropriate navigation line.

```
7   public class Navigation : MonoBehaviour
8   {
9       [SerializeField]
10      private Camera topDownCamera;
11      [SerializeField]
12      private GameObject target;
13
14      private NavMeshPath path;
15      private LineRenderer lineRenderer;
16
17      private bool lineToggle = false;
18
19      void Start()
20      {
21          path = new NavMeshPath();
22          lineRenderer = transform.GetComponent<LineRenderer>();
23      }
24
25      // Update is called once per frame
26      void Update()
27      {
28          if((Input.touchCount >0)&&(Input.GetTouch(0).phase == TouchPhase.Began))
29          {
30              lineToggle = !lineToggle;
31          }
32
33          if (lineToggle)
34          {
35              NavMesh.CalculatePath(transform.position, target.transform.position, NavMesh.AllAreas, path);
36              lineRenderer.positionCount = path.corners.Length;
37              lineRenderer.SetPositions(path.corners);
38              lineRenderer.enabled = true;
39          }
40
41      }
42  }
43
```

Figure 3. Navigation.cs

1.2 CONSTRUCTION OF THE SCENE

A "transparent wall" is employed to separate different areas; characters, including AI for increased realism and small blue figures that replicate the influence of other tourists moving around the scene; and, lastly, the blue cube, which is used to show the position of each exhibit.

1.2.1 TRANSPARENT WALL

Create a simple model based on a side view of a building (figure 4 right), using Occlusion "invisible" materials for the walls and floors (figure 4 left), which can be located under Material -> shader-> VR -> spatial mapping -> Occlusion.

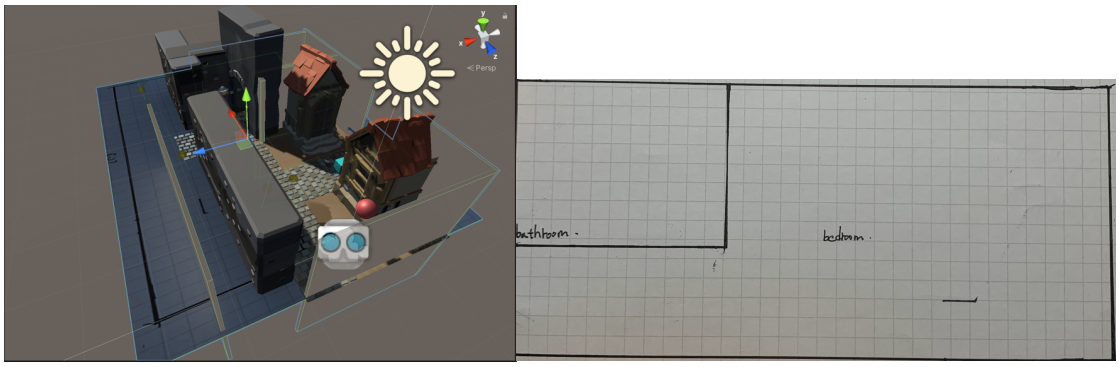

Figure 4. Side view of a building

1.2.2 NAVIGATION DATA

Calculate navigation data in this region and determine the distance between the camera and the objects. Because the size of all models must be scaled proportionally to fit the real world, **Agent Radius = 0.2** is specified here to acquire a greater calculation range (the blue mesh should cover the entire scene.)

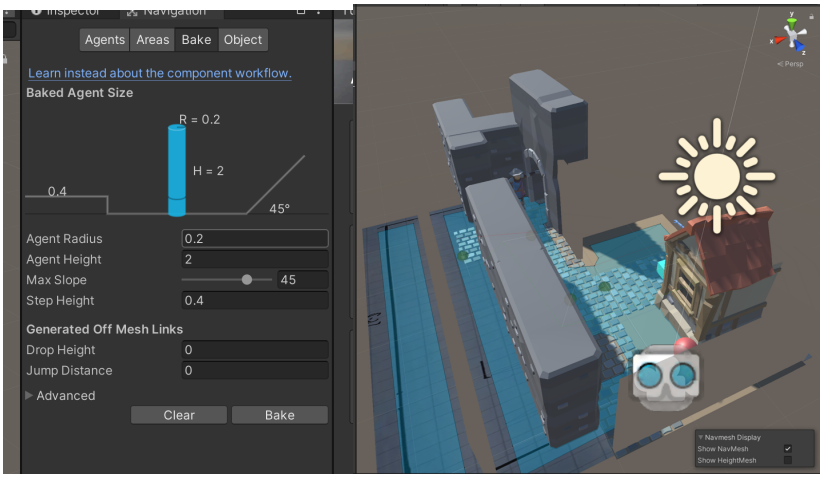

Figure 5. Navmesh and settings

1.2.3 CHARACTER

Figure A, the hat-wearing soldier is a character model with a speaking animation in loops. There are no parameters for this character's actions to be triggered.

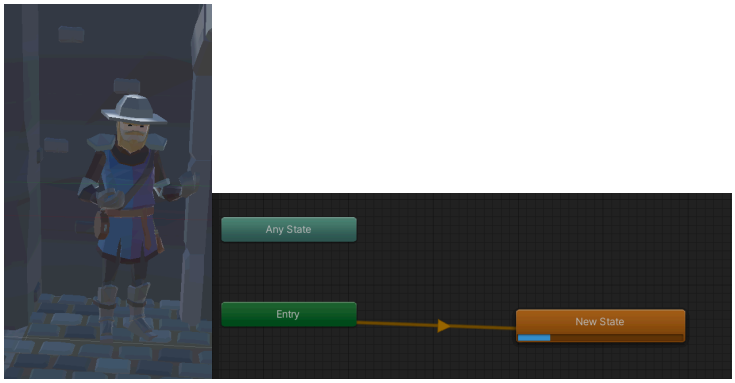

Figure 6. Character model and animation control

Figure B - A blue character moves along a path that has been specified. Her walking action is based on a blend tree, and the **velocityX** and **velocityZ** parameters represent the movement coordinates on the X and Z axes (The Z-axis points to us and the Y-axis points to the sky).

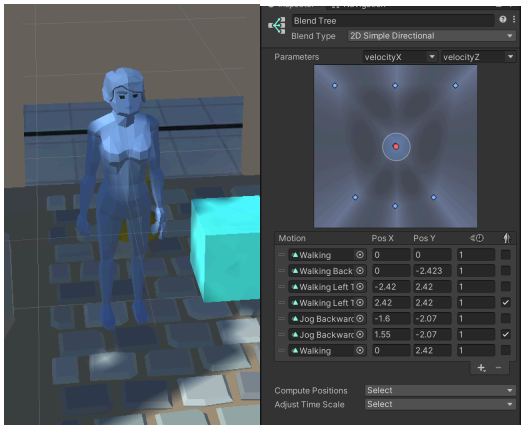

Figure 7. Character model and blend tree setting

Line35 -50 in PedstainController.cs. It is used to figure out how fast **velocityX** and **Z**. As a result of its impact, the character's mobility is proportionate to his or her walking pace. Update the animation settings on line 53-55.

```
35
36          Vector3 worldDeltaPosition = agent.nextPosition - transform.position;
37          // Map 'worldDeltaPosition' to local space
38          float dx = Vector3.Dot(transform.right, worldDeltaPosition);
39          float dz = Vector3.Dot(transform.forward, worldDeltaPosition);
40          Vector2 deltaPosition = new Vector2(dx, dz);
41
42          // Low-pass filter the deltaMove
43          float smooth = Mathf.Min(1.0f, Time.deltaTime / 0.15f);
44          smoothDeltaPosition = Vector2.Lerp(smoothDeltaPosition, deltaPosition, smooth);
45
46          // Update velocity if time advances
47          if (Time.deltaTime > 1e-5f)
48              velocity = smoothDeltaPosition / Time.deltaTime;
49
50          bool shouldMove = velocity.magnitude > 0.5f && agent.remainingDistance > agent.radius;
51
52          // Update animation parameters
53          animator.SetBool("move", shouldMove);
54          animator.SetFloat("velocityX", velocity.x);
55          animator.SetFloat("velocityZ", velocity.y);
```

Figure 8. Animation setting code

Line 23 in **State Walk.cs**, provide the agent with first waypoint location and enable it to travel, **line 30 - 34**, give the character the next coordinate point and enter the waypoint's coordinate loop, **line 40**, switch the state to **State_SayHello** when the character sees the target.

```
22
23          agent.destination = waypoint.transform.position;
24          agent.isStopped = false;
25
26      }
27
28      public void Execute()
29      {
30          if (!agent.pathPending && agent.remainingDistance < 0.5f)
31          {
32              Waypoint nextWaypoint = waypoint.nextWaypoint;
33              waypoint = nextWaypoint;
34              agent.destination = waypoint.transform.position;
35
36          }
37
38          if (owner.seenTarget)
39          {
40              owner.stateMachine.ChangeState(new State_SayHello(owner));
41          }
```

Figure 9. Navmesh agent setting

State_SayHello.cs, line 32 -46, Stop and transition from a moving motion to a greeting motion.

```
30      public void Execute()
31      {
32          if (!agent.pathPending && agent.remainingDistance < 5.0f)
33          {
34              agent.isStopped = true;
35              animator.SetBool("move", false);
36              animator.SetBool("seen", true);
37
38          }
39
40          if (owner.seenTarget != true)
41          {
42              Debug.Log("lost sight");
43              animator.SetBool("seen", false);
44              animator.SetBool("move", true);
45              owner.stateMachine.ChangeState(new State_Walk(owner));
46          }
47
48      }
49
```

Figure 10. State_SayHello.cs

1.3 ARWorldMAP and ARCollaborationData

The unity sample code[2] will be referenced in this section. These two components will deal with the high dislocation of the placement model as well as the interaction of diverse users in the unified scene.

ARWorldMap is an **ARKit feature [3]** that can save unknown scene information and then reload the scan file to improve tracking. White dots will appear on the vertical and horizontal flat surfaces[3] when the programme initially starts, as illustrated in the right picture. The image in the centre displays the current state of the map collection as well as the location of the stored file.

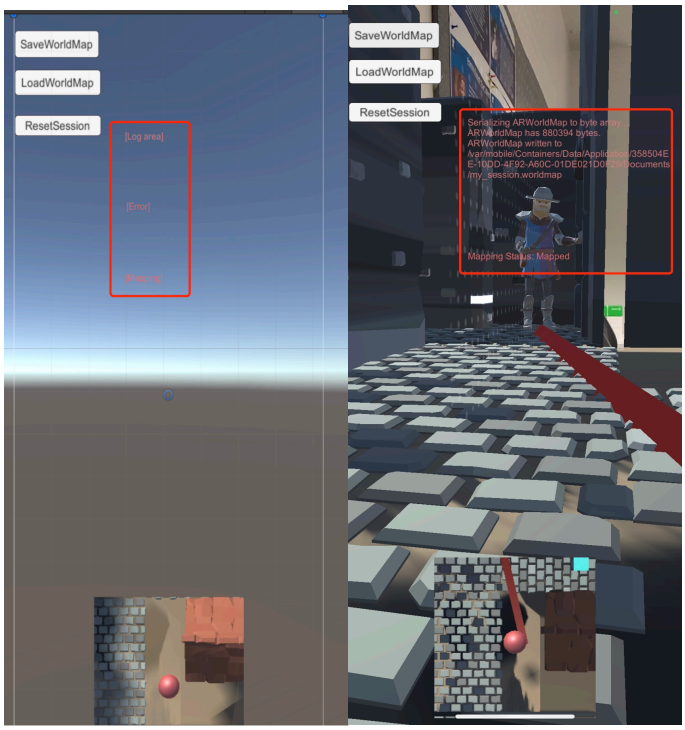

Figure 11. Storyboard

ARCollaborationData is a feature of ARkit [3] that is used to record multi-device collaboration sessions and is used to solve the problem of data transfer between users. This solution has been left out of the system since it requires the use of a unified server for testing.

EVALUATION

We'll utilise user-testing and edge-testing in this section to evaluate the program's usability.

2.1 USER TESTING

This evaluation part looked at two users who used the same equipment and the same version of the application but in separate areas. The users' genuine experiences, word description, and opinions were captured, and all data recordings were agreed upon and carried out with the participants' consent. Below, we'll go over the two users' experiences and experimental designs separately.

2.1.1 CASE A

The experiment is carried out in a normal size field. The room's side perspective is identical to the original design (figure 4 right). **User A** holds a mobile phone and aims to find the blue square.

When the A first turned it on, she was completely perplexed and had no idea how to use it. Then she moved to the target after clicking on the screen and seeing the navigation line emerge. When the user spins the phone to the right, it practically scrapes the wall since part of the field is rather narrow. The experiment comes to an end as the user gets close to the target.

After using it, the user indicated confusion regarding the program's function, and while the surrounding 3D environment was supposed to be visually appealing, it produced some line-of-sight interference.

Figure 12. User A testing

2.1.2 CASE B

The experiment took place in a broader field, and some individuals moved about the venue at the same time. **User B** held a mobile phone to look for the blue cube.

Rather than relaxing as one might assume, the user is tight, moves slowly, and frequently does not move. When following the navigation line, the user mistakenly thinks the 3D model in the scene is the target and directly passes the blue cube target. After reminding, the user has made a correction, and while the situation in **Case A** does not appear when the phone is shifted, but still have dangers when pedestrians pass by.

Users said the navigation feature was extremely good, and the surrounding scenes were quite appealing, but it might be deceptive if there was no hint. When combined with the user's personal museum experience, navigation is an excellent tool for locating a specific sculpture in a collection of exhibits;

however, the surrounding 3D scene is unnecessary because its presence encourages him to be cautious. The movement of people around him also aggravates the situation.

Figure 13. User B testing

"And I think this kind of expression is more appropriate in VR since I have to pay attention to the people around me while holding the phone, and this design is not reasonable." Said by him.

2.2 Edge case testing

2.2.1 Light

With the scattered light and bright light [4], the recognition efficiency is noticeably reduced because there is no prior storage when applying ARWorldMap (in section 1.3), and the elements may be obtained by moving the screen for a distance time. Light has relatively little effect on the program scan while there are files loaded.

2.2.2 Test of standing position

When we modify the location[5] and direction of the program, the entire scene may change. This is because the whole scene position, including height, is based on the camera, thus we add a reset button to reset the entire scene.

REFLECTION

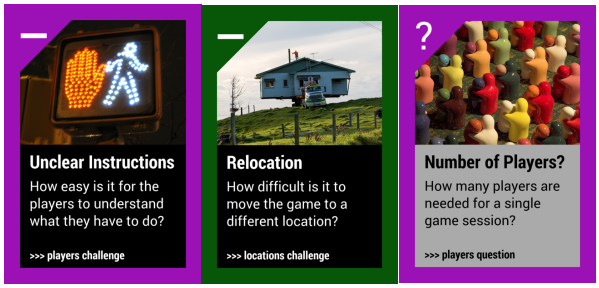

Figure 14. Ideation card

3.1 DIFFERENCES AND PROBLEMS

The design in CW1 relied on virtual reality to connect disparate exhibits and place them in the same game scenario, and we're doing the same thing here with augmented reality. Unlike other AR designs, we use 3D objects to hide most of the surrounding objects, leaving exhibits as "true" aspect of AR.

3.1.1 PROBLEM: SECURITY RISKS

The user may always hold the phone to engage with the scene, as illustrated in the user experiment above. There will be some risks when turning the body. It will unavoidably influence other visitors while users are there. This technology, like smartphones, has the potential to cause mishaps or drive users to unintentionally miss true dangers[5].

The problem is addressed in the threshold [6] by restricting the number of participants and employing a transparent approach to identify adjacent players. In another article[5], the experimenter used voice warnings to inform users of getting too close to the showcase or dangerous. And here we utilize the blue characters to simulate other users in real-time, but because only people who use this application are taken into account, this issue may still happen to both users and other visitors.

3.1.2 PROBLEM: IMMERSION

The physical and VR glasses are used to improve immersion in the threshold[6], so here we increase the sense of immersion by adding 3D objects to the program scene. However, both users were afraid to move too much in the trial due to too many influencing elements, and the crowd's movement aggravated the problem.

We improved by adding navigation lines and buttons , similar to the way in paper[5], however, these features are frequently overlooked by the user, leading to the user believing that the set scene is a distraction (Section 2.1.2).

3.2 HCI CHALLENGE

We focus on two HCI challenges: "*How to design the surrounding environment in which we may hold the device*." and "*integration of entities with the surrounding content*." The analysis is conducted, and the following are the findings.

To provide users with a sufficiently immersive experience, the application must have clear instructions and design. As an event in the context of visiting a museum, it will be more closely linked to the location.

The user experience is influenced by the site's size. Most users will be apprehensive if "there are no barriers on all sides," thus the design should be carried out in places with reference objects.

The site's size in relation to the number of attendees will have an effect on the experience. The experiment in section 2.1 reveals that the experience effect is greater when a user is a single person and the venue is small; otherwise, the experience barriers will grow, and thus the design must address the specific location features. To get the optimum experience, several elements modify the proportional connection between the virtual scene and the real situation.

CITE LIST

1. *AR Foundation Samples*. AR Foundation Samples 2021; Available from: https://github.com/Unity-Technologies/arfoundation-samples.
2. Polygon knights: https://assetstore.unity.com/packages/3d/environments/fantasy/polygon-knights-low-poly-3d-art-by-synty-83694#description
3. Navigation feature: https://www.youtube.com/watch?v=fuHFrMZ4q_s&t=1098s

REFERENCES

1. Milgram, P. and F. Kishino, *A taxonomy of mixed reality visual displays.* IEICE TRANSACTIONS on Information and Systems, 1994. **77**(12): p. 1321-1329.
2. *AR Foundation Samples*. AR Foundation Samples 2021; Available from: https://github.com/Unity-Technologies/arfoundation-samples.
3. Oufqir, Z., A. El Abderrahmani, and K. Satori. *ARKit and ARCore in serve to augmented reality*. IEEE.
4. Amin, D. and S. Govilkar, *Comparative study of augmented reality SDKs.* International Journal on Computational Science & Applications, 2015. **5**(1): p. 11-26.
5. Hunsucker, A.J., E. Baumgartner, and K. McClinton, *Evaluating an AR-based museum experience.* Interactions, 2018. **25**(4): p. 66-68.
6. Tennent, P., et al., *Thresholds.* Journal on Computing and Cultural Heritage, 2020. **13**(2): p. 1-35.