Contents

Introduction

AR Emergency Instructions and Evacuation Guidelines is an augmented reality (AR) application that helps users to learn where to go when there is an emergency and the right way to use a fire extinguisher, in a much interesting and more interactive way. It is very important to ensure everyone knows the assembly point they should go as this saves time and keep them away from dangers.

The main disaster management solution to minimize the impact of man-made and natural threats on building occupants is evacuation. AR is the novel technology that able to enhance evacuation by providing building occupants with virtual contents to improve their performance (Ruggiero, 2018). Asphyxiation or lung damage caused by smoke inhalation, building structure collapse on people, body injury caused by trampling of those caught in crowds of people trying to evacuate are the causes of loss of human life and injury during a building fire (Ahn & Han, 2012). Therefore, it is significant to staff and students to know what to do when there is an emergency.

Objectives

AR Emergency Instructions and Evacuation Guidelines allows users to check every assembly point daily and thus provides users detailed information of which assembly point to go according to their location during an emergency. This application provides respective audio explanations of assembly points and video regarding all assembly points in Universiti Malaysia Pahang. This application also shows the steps to use fire extinguisher by watching a short video.

Scope

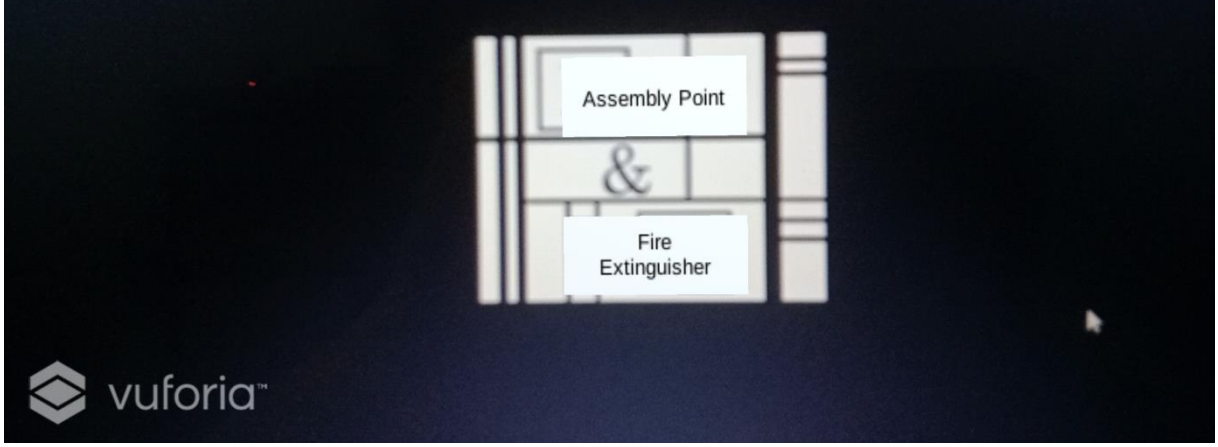
AR Emergency Instructions and Evacuation Guidelines only focuses on assembly points and fire extinguishers in Universiti Malaysia Pahang. Because of some limitations such as the

time to develop this application, the size of the application, and so on, there are only three assembly points being selected. The selected assembly points are assembly point 11, assembly point 12, and assembly point 13. Any fire extinguisher in Universiti Malaysia Pahang can be checked by using this application. Markers will be attached to every assembly point and fire extinguisher in order to allow users to use this application.

Project Results

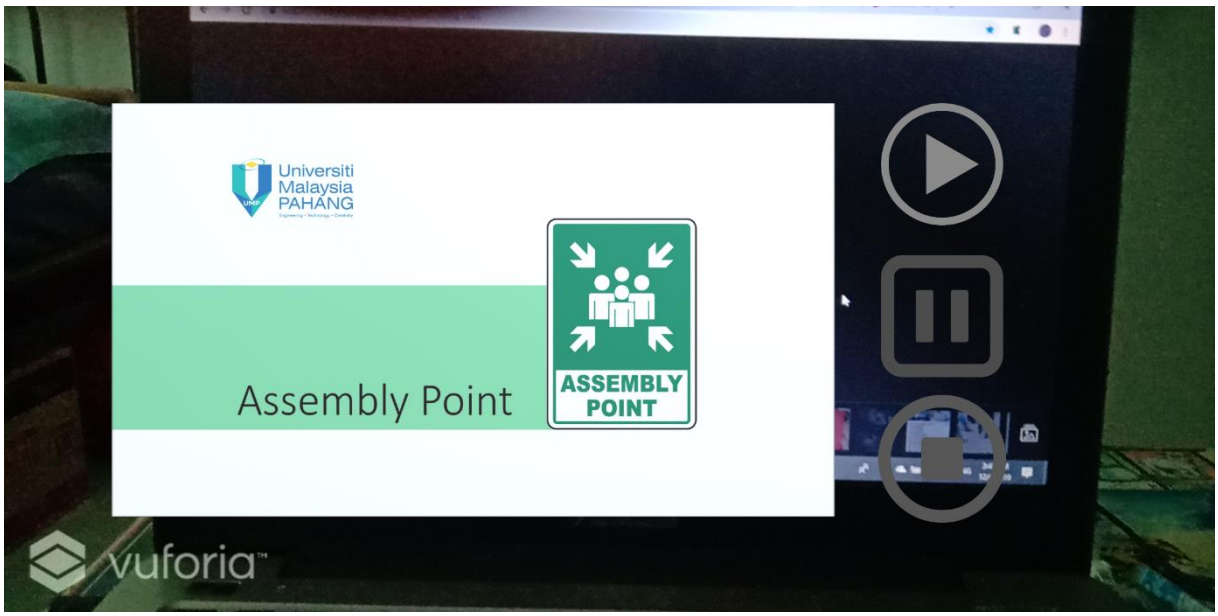
First of all, user opens AR Emergency Instructions and Evacuation Guidelines application. User will see two visual button, one is button 'Start' while the another is button 'Exit'. Button 'Start' is to switch on the AR camera while button 'Exit' is to close the application. User selects a visual button by pressing the button on mobile screen.

User need the specific marker in order to continue using this AR application. This is the marker of this application:

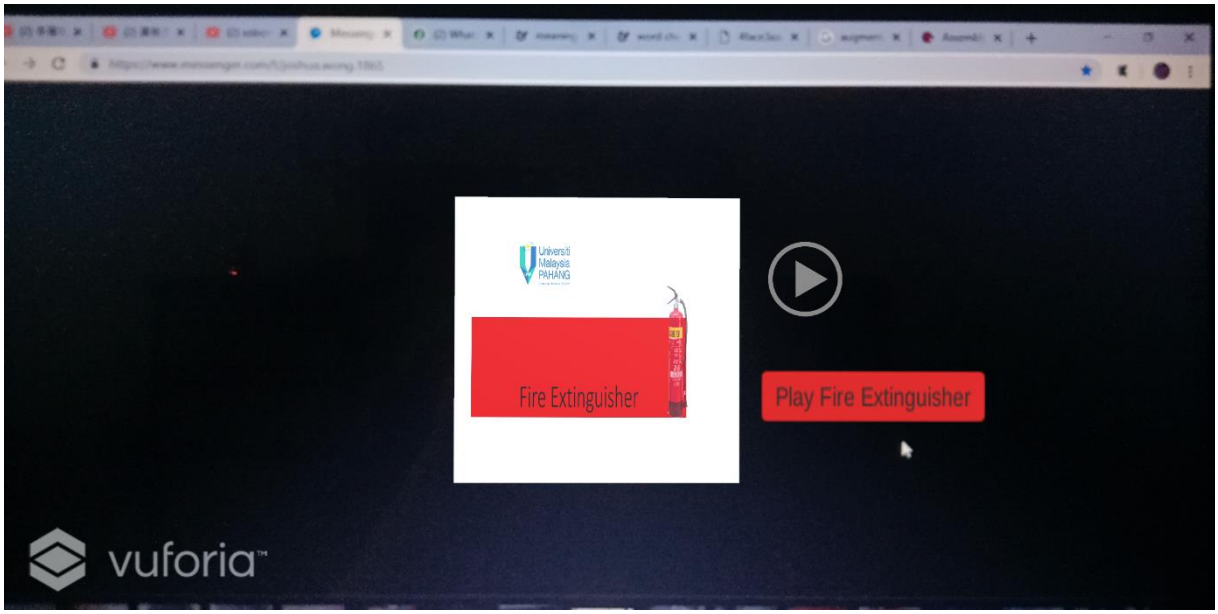


Once user selects visual button ‘Start’, there are two virtual buttons, which are button ‘Assembly Point’ and button ‘Fire Extinguisher’. User has to put finger behind the virtual button to select that button.

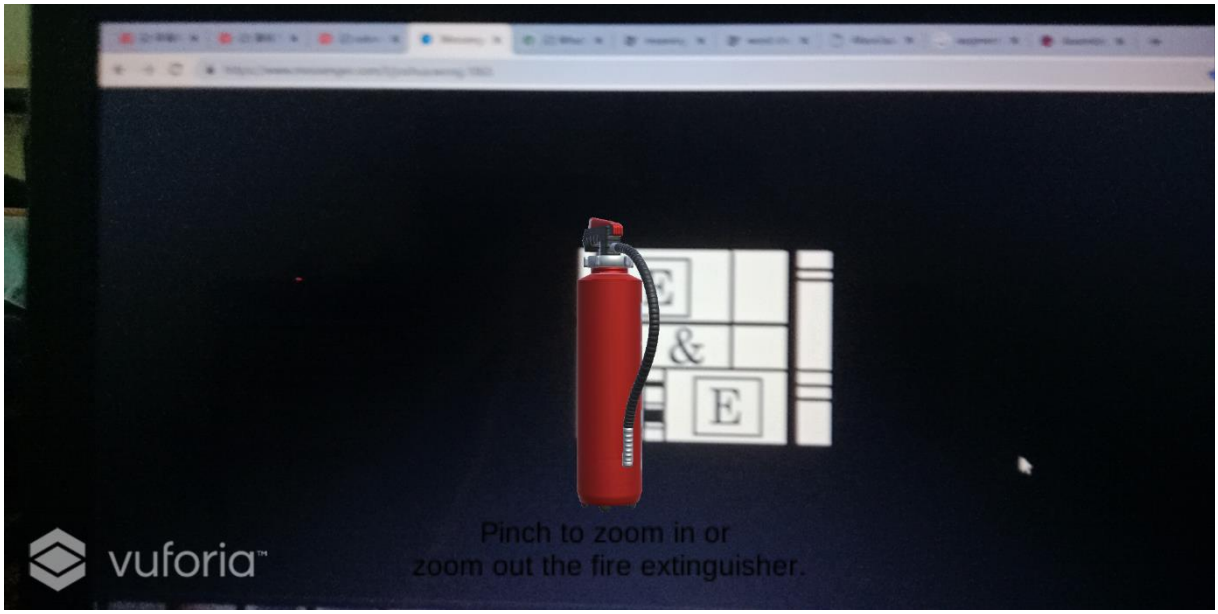
Once user selects virtual button ‘Assembly Point’, user will see an image showing the details of assembly points in area of KK1 and KK3 in Universiti Malaysia Pahang. There are four visual buttons beside the image, which are button ‘AP-11’, button ‘AP-12’, button ‘AP-13’, and button ‘Watch Video’. When user press button ‘AP-11’, button ‘AP-12’ or button ‘AP-13’, user can hear the audio explanation of that selected assembly point while button ‘Watch Video’ directs user to next scene to watch a video about assembly point.



This is the scene that consists video of assembly point. There are three visual buttons, button ‘Play’, button ‘Pause’, and button ‘Stop’. User can select it to play, pause, or stop video.



Once user select the virtual button ‘Fire Extinguisher’ (instead of virtual button ‘Assembly Point’), user can see a video of fire extinguisher. There are two visual buttons beside the video, which are button ‘Play’ and button ‘Play Fire Extinguisher’. User can select button ‘Play’ to play video.



Once user selects the visual button ‘Play Fire Extinguisher’, user will see a 3D model of fire extinguisher with an animation of rotate and user can use two finger to scale the 3D model, either larger or smaller by touching on mobile screen.

Appendix (Full Coding)

VirtualButtonAP.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;
using Vuforia;

public class VirtualButtonAP : MonoBehaviour, IVirtualButtonEventHandler
{
    public GameObject virtualButton;

    public void OnButtonPressed(VirtualButtonBehaviour vb)
    {
        if (vb.VirtualButtonName == "AP")
        {
            SceneManager.LoadScene("SceneAP");
        }
    }

    public void OnButtonReleased(VirtualButtonBehaviour vb)
```

```
    {


    }

    // Start is called before the first frame update
    void Start()
    {
        virtualButton.GetComponent<VirtualButtonBehaviour>().RegisterEventHandler(this);
    }

}
```

VirtualButtonFE.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;
using Vuforia;

public class VirtualButtonFE : MonoBehaviour, IVirtualButtonEventHandler
{
    public GameObject virtualButton;

    public void OnButtonPressed(VirtualButtonBehaviour vb)
    {
```

```
        if (vb.VirtualButtonName == "FE")
        {
            SceneManager.LoadScene("SceneFE");
        }
    }

    public void OnButtonReleased(VirtualButtonBehaviour vb)
    {

    }

    // Start is called before the first frame update
    void Start()
    {
        virtualButton.GetComponent<VirtualButtonBehaviour>().RegisterEventHandler(this);
    }

}
```

VideoScriptAP.cs

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

```
public class VideoScriptAP : MonoBehaviour
{
    public UnityEngine.Video.VideoClip videoClip;
        public AudioSource MusicSource;



    // Start is called before the first frame update
    void Start()
    {
        var videoPlayer = gameObject.AddComponent<UnityEngine.Video.VideoPlayer>();
        var audioSource = gameObject.AddComponent<AudioSource>();
        videoPlayer.playOnAwake = false;
        videoPlayer.clip = videoClip;


    }


    public void PlayVideo()
    {
        var vp = GetComponent<UnityEngine.Video.VideoPlayer>();
        vp.Play();
    }


    public void PauseVideo()
```

```
    {
        var vp = GetComponent<UnityEngine.Video.VideoPlayer>();
        vp.Pause();
    }

    public void StopVideo()
    {
        var vp = GetComponent<UnityEngine.Video.VideoPlayer>();
        vp.Stop();
    }

}
```

VideoScriptFE.cs

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

public class VideoScriptFE : MonoBehaviour
{
    public UnityEngine.Video.VideoClip videoClip;
    public AudioSource MusicSource;
```

```
    // Start is called before the first frame update
    void Start()
    {
        var videoPlayer = gameObject.AddComponent<UnityEngine.Video.VideoPlayer>();
        var audioSource = gameObject.AddComponent<AudioSource>();
        videoPlayer.playOnAwake = false;
        videoPlayer.clip = videoClip;
    }

    public void PlayVideo()
    {
        var vp = GetComponent<UnityEngine.Video.VideoPlayer>();
        vp.Play();
    }

}
```

SceneAPAudio.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneAPAudio : MonoBehaviour
{
```

```
public AudioSource MusicSource;
public AudioClip MusicClip;

// Start is called before the first frame update
void Start()
{
    MusicSource.clip = MusicClip;
}

public void AP11Audio()
{
    MusicSource.Play();
}

public void AP12Audio()
{
    MusicSource.Play();
}

public void AP13Audio()
{
    MusicSource.Play();
}
```

```
    public void WatchAPVideo()
    {
        SceneManager.LoadScene("SceneAPVideo");
    }
}
```

ButtonScript.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ButtonScript : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

   public void StartARCamera()
    {
        SceneManager.LoadScene("SceneSelection");
    }
```

```
    public void ExitApp()
    {
        Application.Quit();
    }

    public void WatchVideo()
    {
        SceneManager.LoadScene("SceneAPVideo");
    }

    public void PlayFE()
    {
        SceneManager.LoadScene("ScenePlayFE");
    }

}
```

BackSceneWelcome.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
```

```
public class BackSceneWelcome : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {


    }


    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            SceneManager.LoadScene("SceneWelcome");
        }
    }
}
```

BackSceneSelection.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
```

```
public class BackSceneSelection : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            SceneManager.LoadScene("SceneSelection");
        }
    }
}
```

BackSceneFE.cs

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

```
using UnityEngine.SceneManagement;


public class BackSceneFE : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {


    }


    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            SceneManager.LoadScene("SceneFE");
        }
    }
}
```

BackSceneAP.cs

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class BackSceneAP : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            SceneManager.LoadScene("SceneAP");
        }
    }
}
```

LeanScale.cs

```
using UnityEngine;
```

```
namespace Lean.Touch
{
	/// <summary>This script allows you to scale the current GameObject.</summary>
	[HelpURL(LeanTouch.HelpUrlPrefix + "LeanScale")]
	public class LeanScale : MonoBehaviour
	{
		[Tooltip("Ignore fingers with StartedOverGui?")]
		public bool IgnoreStartedOverGui = true;

		[Tooltip("Ignore fingers with IsOverGui?")]
		public bool IgnoreIsOverGui;

		[Tooltip("Allows you to force rotation with a specific amount of fingers (0 = any)")]
		public int RequiredFingerCount;

		[Tooltip("Does scaling require an object to be selected?")]
		public LeanSelectable RequiredSelectable;

		[Tooltip("The camera that will be used to calculate the zoom (None = MainCamera)")]
		public Camera Camera;
```

```
        [Tooltip("If you want the mouse wheel to simulate pinching then set the strength of it here")]
        [Range(-1.0f, 1.0f)]
        public float WheelSensitivity;

        [Tooltip("Should the scaling be performanced relative to the finger center?")]
        public bool Relative;

        [Tooltip("Should the scale value be clamped?")]
        public bool ScaleClamp;

        [Tooltip("The minimum scale value on all axes")]
        public Vector3 ScaleMin;

        [Tooltip("The maximum scale value on all axes")]
        public Vector3 ScaleMax;

#if UNITY_EDITOR
        protected virtual void Reset()
        {
            Start();
        }
#endif

        protected virtual void Start()
```

```
        {
            if (RequiredSelectable == null)
            {
                RequiredSelectable = GetComponent<LeanSelectable>();
            }
        }

        protected virtual void Update()
        {
            // Get the fingers we want to use
            var fingers = LeanSelectable.GetFingers(IgnoreStartedOverGui, IgnoreIsOverGui, RequiredFingerCount, RequiredSelectable);

            // Calculate pinch scale, and make sure it's valid
            var pinchScale = LeanGesture.GetPinchScale(fingers, WheelSensitivity);

            if (pinchScale != 1.0f)
            {
                // Perform the translation if this is a relative scale
                if (Relative == true)
                {
                    var pinchScreenCenter = LeanGesture.GetScreenCenter(fingers);
```

```
                    if (transform is RectTransform)
                    {
                        TranslateUI(pinchScale, pinchScreenCenter);
                    }
                    else
                    {
                        Translate(pinchScale, pinchScreenCenter);
                    }
                }

                // Perform the scaling
                Scale(transform.localScale * pinchScale);
            }
        }

        protected virtual void TranslateUI(float pinchScale, Vector2 pinchScreenCenter)
        {
            // Screen position of the transform
            var screenPoint = RectTransformUtility.WorldToScreenPoint(Camera, transform.position);

            // Push the screen position away from the reference point based on the scale
            screenPoint.x = pinchScreenCenter.x + (screenPoint.x - pinchScreenCenter.x) * pinchScale;
```

```
                screenPoint.y = pinchScreenCenter.y + (screenPoint.y - pinchScreenCenter.y) * pinchScale;

                // Convert back to world space
                var worldPoint = default(Vector3);

                if (RectTransformUtility.ScreenPointToWorldPointInRectangle(transform.parent as RectTransform, screenPoint, Camera, out worldPoint) == true)
                {
                    transform.position = worldPoint;
                }
            }

            protected virtual void Translate(float pinchScale, Vector2 screenCenter)
            {
                // Make sure the camera exists
                var camera = LeanTouch.GetCamera(Camera, gameObject);

                if (camera != null)
                {
                    // Screen position of the transform
                    var screenPosition = camera.WorldToScreenPoint(transform.position);
```

```
                        // Push the screen position away from the reference point based on the scale
                        screenPosition.x = screenCenter.x + (screenPosition.x - screenCenter.x) * pinchScale;
                        screenPosition.y = screenCenter.y + (screenPosition.y - screenCenter.y) * pinchScale;

                        // Convert back to world space
                        transform.position = camera.ScreenToWorldPoint(screenPosition);
                    }
                    else
                    {
                        Debug.LogError("Failed to find camera. Either tag your cameras MainCamera, or set one in this component.", this);
                    }
                }

                protected virtual void Scale(Vector3 scale)
                {
                    if (ScaleClamp == true)
                    {
                        scale.x = Mathf.Clamp(scale.x, ScaleMin.x, ScaleMax.x);
                        scale.y = Mathf.Clamp(scale.y, ScaleMin.y, ScaleMax.y);
                        scale.z = Mathf.Clamp(scale.z, ScaleMin.z, ScaleMax.z);
                    }
```

```
                transform.localScale = scale;
            }
        }
}
```

LeanTouch.cs (imported directly from Lean Touch)

LeanSelectable.cs (imported directly from Lean Touch)

LeanSnapshot.cs (imported directly from Lean Touch)

LeanFinger.cs (imported directly from Lean Touch)

References

1. ARWAY. (2017, September 05). IMAREC - Augmented Reality Emergency Evacuation & Communication App. Retrieved from https://www.youtube.com/watch?v=p8xKVWfYr-w
2. Ismail, K. A. (n.d.). Retrieved from http://oshmo.ump.edu.my/index.php/en/info/assembly-point
3. Fire Extinguisher 3D Model. (n.d.). Retrieved from https://free3d.com/3d-model/fire-extinguisher-60547.html
4. Ahn, J., & Han, R. (2012). An indoor augmented-reality evacuation system for the Smartphone using personalized Pedometry. *Human-Centric Computing and Information Sciences*, *2*(1), 1–23. https://doi.org/10.1186/2192-1962-2-18
5. Ruggiero, L. (2018). A Review of Augmented Reality Applications for Building Evacuation. Retrieved from http://arxiv.org/abs/1804.04186