ДОДАТОК А

ARManagerScript.cs (Скрипт з керування AR елементами)

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using UnityEngine.XR.ARFoundation;

public class ARManagerScript : MonoBehaviour
{
    public GameObject objectToPlace
    public ARRaycastManager raycastManager;
    public ARPlaneManager planeManager;

    public Button button;

    private static string objectToPlaceName;
    private static bool isObjectToPlaceExists;

    public void ClearScene()
    {
        if (isObjectToPlaceExists)
        {
            foreach (var trackable in planeManager.trackables)
            {
                trackable.gameObject.SetActive(false);
            }
            Destroy(GameObject.Find(objectToPlaceName));

            isObjectToPlaceExists = false;
            SceneManager.LoadScene(1);
        }
    }

    void Update()
    {
        if (Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Began)
        {
            List<ARRaycastHit> hits = new List<ARRaycastHit>();
```

```
            raycastManager.Raycast(Input.GetTouch(0).position, hits,
UnityEngine.XR.ARSubsystems.TrackableType.Planes);

            if (hits.Count > 0 && !isObjectToPlaceExists)
            {
                var createdObject = GameObject.Instantiate(objectToPlace, hits[0].pose.position,
hits[0].pose.rotation);

                if (createdObject.transform != null)
                {
                    isObjectToPlaceExists = true;
                }

                objectToPlaceName = createdObject.name;
            }
        }
    }
}
```

AndroidControl.cs (Скрипт, який керує кнопками Android та відображенням панелі сповіщень)

```
using System;
using UnityEngine;

public class AndroidControl : MonoBehaviour
{
    private void Start()
    {
        if (Application.platform == RuntimePlatform.Android)
        {
            Screen.fullScreen = false;

            AndroidUtility.ShowStatusBar(Color.black);
        }
    }

    void Update()
    {
        if (Application.platform == RuntimePlatform.Android)
        {
            if (Input.GetKey(KeyCode.Escape))
            {
                UnityEngine.SceneManagement.SceneManager.LoadScene(0);
            }
        }
    }
}
public static class AndroidUtility
{
    private const int MinStatusBarColorApi = 21;
    private const int SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN = 0x00000400;

    private static AndroidJavaObject activity;

    public static void ShowStatusBar(Color color)
    {
        int androidColor = ConvertColorToAndroidColor(color);
```

```
        RunOnUiThread(() =>
        {
            using (var window = Window)
            {
                window.Call("clearFlags", SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
                if (GetApi() >= MinStatusBarColorApi)
                {
                    window.Call("setStatusBarColor", androidColor);
                }
                else
                {
                    Debug.LogWarning("Changing the status bar color is not supported on Android API lower than
Lollipop.");
                }
            }
        });
    }
    private static void RunOnUiThread(Action action)
    {
        Activity.Call("runOnUiThread", new AndroidJavaRunnable(action));
    }
    private static AndroidJavaObject Activity
    {
        get
        {
            if (activity == null)
            {
                var unityPlayer = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
                activity = unityPlayer.GetStatic<AndroidJavaObject>("currentActivity");
            }
            return activity;
        }
    }
    private static AndroidJavaObject Window
    {
        get
        {
            return Activity.Call<AndroidJavaObject>("getWindow");
        }
    }
    private static int GetApi()
    {
        using (var version = new AndroidJavaClass("android.os.Build$VERSION"))
        {
            return version.GetStatic<int>("SDK_INT");
        }
    }
    private static int ConvertColorToAndroidColor(Color color)
    {
        Color32 color32 = color;
        int alpha = color32.a;
        int red = color32.r;
        int green = color32.g;
        int blue = color32.b;
        using (var colorClass = new AndroidJavaClass("android.graphics.Color"))
        {
            int androidColor = colorClass.CallStatic<int>("argb", alpha, red, green, blue);
            return androidColor;
```

```
        }
    }
}
```

ChangeScene.cs (Скрипт переходу між сценами)

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ChangeScene : MonoBehaviour
{
    public void MoveToScene(int sceneIndex)
    {
        SceneManager.LoadScene(sceneIndex);
    }

    public void MoveToScene(int sceneIndex, string siteName)
    {
        ScenesSharedData.SiteName = siteName;

        SceneManager.LoadScene(sceneIndex);
    }
}
```

LoadingImageRotation.cs (Скрипт повертання зображення під час завантаження карти)

```
using UnityEngine;
using UnityEngine.UI;

public class LoadingImageRotation : MonoBehaviour
{
    public RawImage rotationImage;

    void Update()
    {
        if (rotationImage.GetComponent<RawImage>().texture.name == "loadingCircle")
        {
            transform.Rotate(0f, 0f, -100f * Time.deltaTime);
        }
    }
}
```

ScenesSharedData.cs (Файл з важливою інформацією, яка спільна між всіма сценами)

```
using System;

public class ScenesSharedData
{
    public static SitesInfo[] Values; // All information about sites
    public static string SiteName; // Seleceted by user site short name
}
```

SitesData.cs (Файл для отримання всієї інформації про місця)

```
using System;

public class SitesData
{
```

```
    public static SitesInfo[] GetData()
    {
        SitesInfo[] values = new SitesInfo[3];

        values[0] = new SitesInfo(0,
            "Оперний театр",
            "\tЛьвівський національний академічний театр опери та балету імені Соломії " +
            "Крушельницької — театр опери і балету у Львові, в історичному центрі міста, на проспекті Свободи,
28. " +
            "\n\n\tЛьвівський оперний театр відомий у всій Європі. " +
            "Будинок побудований в стилі неоренесансу. На цьому наголошує характерний багатий скульптурний
декор, величні колони, " +
            "складні обманні ніші, ліпнина та витончені балюстради." +
            "\n\n\tОдна з найбільш примітних скульптурних композицій – «Радість і страждання життя». " +
            "Вона прикрашає собою центральний карниз, а прямо під нею розташувалися статуї муз-покровительок
мистецтва. ",
            "opera", true, 49.844167f, 24.026389f, "https://opera.lviv.ua/", "ChIJiWq8-XLdOkcRH8_-tOw_ePk");

        values[1] = new SitesInfo(1,
            "Площа Ринок",
            "\tПлоща Ринок — центральний майдан у Львові, історичне серце сучасного міста, " +
            "характерне для середньовічної архітектури європейських міст. " +
            "Центр історичної дільниці Середмістя." +
            "\n\n\tПлоща Ринок є однією з найдавніших і найцікавіших частин Львова. З часів Середньовіччя і
донині вона є важливим центром міського життя. " +
            "Площа виникла у XIII–XIV ст. як центральна частина “міста в мурах”, розпланованого тоді згідно із
засадами привілею Магдебурзького права. " +
            "\n\n\tЗабудова, що оточує площу сьогодні, формувалася протягом століть і зберігає елементи та
нашарування різних стилів і епох. " +
            "На її вирішення мали вплив загальноєвропейські архітектурні тенденції, місцеві традиції, мистецтво
різних етнічних громад, що проживали у Львові. ",
            "rynok", false, 49.841466f, 24.031265f, "https://lia.lvivcenter.org/uk/themes/rynok-overview/",
            "ChIJVVVlrm3dOkcRZE28KxteGqc");

        values[2] = new SitesInfo(2,
            "Високий замок",
            "\tВисокий Замок — замок, збудований під керівництвом Короля Русі Лева Даниловича, потім —
польського короля Казимира III на Замковій горі у Львові. "
             + "Майже повністю розібраний протягом XIX століття.\n\n\tЛьвів – дуже холмисте місто. " +
             "“Головний” пагорб міста називається “Замкова гора”, або ж частіше “Високий Замок”. " +
             "Однак туристи, які очікують тут побачити справжню фортецю і не знають історії будуть розчаровані "
+
             "– від замку тут залишилися лише невеликі руїни, та назва. " +
             "Сьогодні тут розмістився найвищий та найвідоміший оглядовий майданчик, " +
             "телевізійна вежа та доволі красивий парк." +
             "\n\n\tПарк був заснований в 1835 р. Назву він отримав завдяки фортеці, залишки якої збереглися на
території. " +
             "Структура парку орієнтована на модель ландшафтного парку. Має нижню та верхню тераси. " +
             "Над верхньою у другій половині XIX ст. насипали штучний курган Люблінської унії, з оглядовим
майданчиком на вершині. " +
             "Окраса парку — ясенева та каштанова алеї на нижній терасі.",
            "zamok", false, 49.848333f, 24.038889f, "https://lviv.travel/ua/news/netipovii-lviv-park-visokii-zamok",
"ChIJKSET2hPdOkcR-FIW8zwGF-Y");

        return values;
    }
}
```

SitesInfo.cs (Основний клас для тримання інформації про місця)

```
using System;
```

```
[Serializable]
public class SitesInfo
{
    public int Id; // Sites ID
    public string Name; // Official sites name
    public string Description; // Short Description of site
    public string ShortName; // Short version of name for work
    public bool IsArSupported;
    public float Latitude; // Sites latitude
    public float Longitude; // Sites lonitude
    public string URL; // URL to external site with info about site
    public string PlaceID; // Sites ID on Google Maps

    public SitesInfo() { }

    public SitesInfo(int id, string name, string description, string shortName, bool isArSupported, float latitude,
float longitude, string uRL, string placeID)
    {
        Id = id;
        Name = name;
        Description = description;
        ShortName = shortName;
        IsArSupported = isArSupported;
        Latitude = latitude;
        Longitude = longitude;
        URL = uRL;
        PlaceID = placeID;
    }
}
```

ChangeTogglePositionScript.cs (Скрипт для керування дозволами Android)

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

public class ChangeTogglePositionScript : MonoBehaviour
{
    public List<RectTransform> knobs = new List<RectTransform>(2);

    private void Start()
    {
        if (Application.platform == RuntimePlatform.Android)
        {
            if
(!UnityEngine.Android.Permission.HasUserAuthorizedPermission(UnityEngine.Android.Permission.Camera))
            {
                knobs[0].GetComponent<RectTransform>().localPosition = new
Vector3(knobs[0].anchoredPosition.x, knobs[0].anchoredPosition.y);
            }
            else
            {
                knobs[0].GetComponent<RectTransform>().localPosition = new Vector3(-
knobs[0].anchoredPosition.x, knobs[0].anchoredPosition.y);
            }
```

```
        if
(!UnityEngine.Android.Permission.HasUserAuthorizedPermission(UnityEngine.Android.Permission.FineLocati
on))
        {
            knobs[1].GetComponent<RectTransform>().localPosition = new
Vector3(knobs[1].anchoredPosition.x, knobs[1].anchoredPosition.y);
        }
        else
        {
            knobs[1].GetComponent<RectTransform>().localPosition = new Vector3(-
knobs[1].anchoredPosition.x, knobs[1].anchoredPosition.y);
        }
    }
  }

  public void ChangeKnobTransform()
  {
    if (EventSystem.current.currentSelectedGameObject.name == "SwitchCamera")
    {
      Permission.RequestUserPermission(Permission.Camera);

      knobs[0].GetComponent<RectTransform>().localPosition = new Vector3(-knobs[0].anchoredPosition.x,
knobs[0].anchoredPosition.y);
    }

    if (EventSystem.current.currentSelectedGameObject.name == "SwitchLocation")
    {
      Permission.RequestUserPermission(Permission.FineLocation);

      knobs[1].GetComponent<RectTransform>().localPosition = new Vector3(-knobs[1].anchoredPosition.x,
knobs[1].anchoredPosition.y);
    }
  }
}
```

GetMapScript.cs (Скрипт, який звертається до API для отримання зображення карти Google Maps)

```
using System.Linq;
using System.Collections;
using System.Globalization;
using System.Threading;
using TMPro;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

public class GetMapScript : MonoBehaviour
{
  public RawImage mapImage;

  public Texture2D internetAccess;
  public Texture2D locationAccess;
  public Texture2D loadingImage;

  public TMP_Dropdown dropdown;

  private CultureInfo culture;
```

```
private string url = "";

public string apiKey = "";

private readonly int zoom = 16;
private readonly string mapResolution = "low";
private readonly string mapType = "roadmap";

void Start()
{
   if (Application.platform == RuntimePlatform.Android)
   {
      if (Application.internetReachability == NetworkReachability.NotReachable)
      {
         mapImage.texture = internetAccess;
         return;
      }

      if (!UnityEngine.Input.location.isEnabledByUser)
      {
         mapImage.texture = locationAccess;
         return;
      }

      if (Thread.CurrentThread.CurrentCulture.Name == "uk-UA")
      {
         culture = CultureInfo.CreateSpecificCulture("en-US");
      }
      else
      {
         culture = CultureInfo.CreateSpecificCulture("en-US");
      }

      CultureInfo.DefaultThreadCurrentCulture = culture;
      CultureInfo.DefaultThreadCurrentUICulture = culture;
      Thread.CurrentThread.CurrentCulture = culture;
      Thread.CurrentThread.CurrentUICulture = culture;

      StartCoroutine(GetGoogleMap());
   }
}

IEnumerator GetGoogleMap()
{
   mapImage.GetComponent<RawImage>().texture = loadingImage;

   Input.location.Start();

   int maxWait = 20;
   while (Input.location.status == LocationServiceStatus.Initializing && maxWait > 0)
   {
      Debug.Log("Initialazing Failed");
      yield return new WaitForSeconds(1);
      maxWait--;
   }

   if (maxWait < 1)
```

```
        {
            Debug.Log("Time out");
            yield break;
        }

        if (Input.location.status == LocationServiceStatus.Failed)
        {
            Debug.Log("Location Failed");
            yield break;
        }

        Input.location.Stop();

        var currentSite = SitesData.GetData().Where(s => s.ShortName == ScenesSharedData.SiteName).ToList();

        url =
$"https://maps.googleapis.com/maps/api/staticmap?center={currentSite[0].Latitude},{currentSite[0].Longitude}
" +

$"&zoom={zoom}&size={mapImage.rectTransform.sizeDelta.x}x{mapImage.rectTransform.sizeDelta.y}" +
            $"&scale={mapResolution}&maptype={mapType}" +
            $"&markers=color:red%7C{currentSite[0].Latitude},{currentSite[0].Longitude}" +
            $"&markers=color:blue%7C{Input.location.lastData.latitude},{Input.location.lastData.longitude}" +
            $"&key={apiKey}";

        UnityWebRequest www = UnityWebRequestTexture.GetTexture(url);
        yield return www.SendWebRequest();

        if (www.result != UnityWebRequest.Result.Success)
        {
            Debug.Log("WWW ERROR: " + www.error);
        }
        else
        {
            mapImage.GetComponent<RawImage>().texture =
((DownloadHandlerTexture)www.downloadHandler).texture;
        }
    }
}
```

OpenExternalMapScript.cs (Скрипт для відкриття зовнішньої програми Google Maps)

```
using System.Linq;
using UnityEngine;
using UnityEngine.UI;

public class OpenExternalMapScript : MonoBehaviour
{
    public Button button;

    public void OpenExternalMap()
    {
        if (Application.internetReachability != NetworkReachability.NotReachable)
        {
            var currentSite = ScenesSharedData.Values.Where(x => x.ShortName ==
ScenesSharedData.SiteName).ToList();
```

```
Application.OpenURL($"https://www.google.com/maps/search/?api=1&query={currentSite[0].Latitude}%{curr
entSite[0].Longitude}&query_place_id={currentSite[0].PlaceID}");
        }
    }
}
```

ResetScript.cs (Скрипт для скидання основної сторінки)

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ResetScript : MonoBehaviour
{
    public void ResetScene()
    {
        ScenesSharedData.Values = null;
        SceneManager.LoadScene(0);
    }
}
```

SearchScript.cs (Скрипт для пошуку конкретних місць)

```
using System.Linq;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class SearchScript : MonoBehaviour
{
    public InputField input;
    public Button button;

    public void Search()
    {
        if (string.IsNullOrEmpty(input.text)) { }
        else
        {
            SitesInfo[] values = new SitesInfo[3];

            for (int i = 0; i < SitesData.GetData().Length; i++)
            {
                if (SitesData.GetData()[i].Name.ToLower().Contains(input.text.ToLower()))
                {
                    values[i] = SitesData.GetData()[i];
                }
            }

            if (values.Where(x => x != null).ToArray().Length == 0)
            {
                ScenesSharedData.Values = SitesData.GetData();
            }
            else
            {
                ScenesSharedData.Values = values.Where(x => x != null).ToArray();
                SceneManager.LoadScene(0);
            }
        }
```

```
    }
}
```

LoadTextIntoDropdown.cs (Скрипт для завантаження тексту для сторінки з AR)

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

public class LoadTextIntoDropdown : MonoBehaviour
{
    private static readonly Dictionary<int, string> data = new Dictionary<int, string>(2);

    public TMP_Dropdown dropdown;

    void Start()
    {
        data.Clear();
        data.Add(0, SitesData.GetData()
                .Where(x => x.ShortName == ScenesSharedData.SiteName).First().Description);
        data.Add(1, "Дізнатися більше...");

        dropdown.AddOptions(data.Values.ToList());
        dropdown.RefreshShownValue();
    }

    public void OpenExternalSite()
    {
        var dropdownValue = dropdown.value;

        if (dropdown.value != 0)
        {
            if (Application.internetReachability == NetworkReachability.NotReachable) { }
            else
            {
                Application.OpenURL($"{ScenesSharedData.Values.Where(x => x.ShortName ==
ScenesSharedData.SiteName).First().URL}");
            }
        }
    }
}
```

LoadTextScript.cs (Скрипт для завантаження тексту в сторінку без AR)

```
using TMPro;
using UnityEngine;
using System.Linq;

public class LoadTextScript : MonoBehaviour
{
    public TMP_Text text;

    void Start()
    {
        if (ScenesSharedData.Values is null || ScenesSharedData.SiteName is null) // видалити
        {
```

```
            ScenesSharedData.Values = SitesData.GetData();

            text.text = ScenesSharedData.Values[0].Description;
        }
        else
        {
            text.text = ScenesSharedData.Values.Where(x => x.ShortName ==
ScenesSharedData.SiteName).First().Description;
        }
    }
}
```

DropdownAddOptionsScript.cs (Скрипт для завантаження опцій в випадний список на основній сторінці)

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

public class DropdownAddOptionsScript : MonoBehaviour
{
    private static readonly Dictionary<int, string> data = new()
    {
        { 0, "" },
        { 1, "Налаштування" },
        { 2, "Про програму" }
    };

    public TMP_Dropdown dropdown;

    void Start()
    {
        dropdown.AddOptions(data.Values.ToList());
        dropdown.RefreshShownValue();
    }

    public void ChangeSceneOnClick()
    {
        SceneManager.LoadScene(3);
    }
}
```

LoadDataToScrollScript.cs (Скрипт для відображення інформації про місця на основній сторінці)

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

public class LoadDataToScrollScript : MonoBehaviour
{
    public List<Texture2D> textures;
    public Transform content;
    public Sprite buttonSprite;
```

```
    public InputField inputField;

    public void CreateObjects(SitesInfo[] values)
    {
        content.GetComponent<RectTransform>().sizeDelta = new Vector2(1080, 1000 * (values.Length));
        var different = SitesData.GetData().Length - values.Length;
        for (int i = 0; i < values.Length; i++)
        {
            var button = DefaultControls.CreateButton(new DefaultControls.Resources());
            var image = DefaultControls.CreateRawImage(new DefaultControls.Resources());
            var textTitle = DefaultControls.CreateText(new DefaultControls.Resources());
            var textDescription = DefaultControls.CreateText(new DefaultControls.Resources());
            var panel = DefaultControls.CreatePanel(new DefaultControls.Resources());

            button.transform.name = $"Button {i}";
            image.transform.name = $"Image {i}";
            textTitle.transform.name = $"Text Main {i}";
            textDescription.transform.name = $"Description {i}";
            panel.transform.name = $"Panel {i}";

            button.transform.tag = "GameController";
            image.transform.tag = "GameController";
            textTitle.transform.tag = "GameController";
            textDescription.transform.tag = "GameController";
            panel.transform.tag = "GameController";

            panel.GetComponent<RectTransform>().anchorMax = new Vector2(0.5f, 0.5f);
            panel.GetComponent<RectTransform>().anchorMin = new Vector2(0.5f, 0.5f);
            panel.GetComponent<RectTransform>().pivot = new Vector2(0.5f, 0.5f);
            panel.GetComponent<RectTransform>().sizeDelta = new Vector2(1080, 1000);
            panel.GetComponent<RectTransform>().localPosition = new Vector3(0, 0 + ((i + (values.Length - 1) *
500) - i * 1000));
            panel.GetComponent<Image>().color = UnityEngine.Color.black;
            panel.transform.SetParent(content, false);

            button.transform.SetParent(panel.transform);
            image.transform.SetParent(panel.transform);
            textTitle.transform.SetParent(panel.transform);
            textDescription.transform.SetParent(panel.transform);

            image.GetComponent<RectTransform>().localPosition = new Vector3(0, 250);
            image.GetComponent<RectTransform>().sizeDelta = new Vector2(1080, 500);

            button.GetComponent<RectTransform>().localPosition = new Vector3(350, -100);
            button.GetComponent<RectTransform>().sizeDelta = new Vector2(200, 100);

            textTitle.GetComponent<RectTransform>().localPosition = new Vector3(-150, -100);
            textTitle.GetComponent<RectTransform>().sizeDelta = new Vector2(600, 100);

            textDescription.GetComponent<RectTransform>().localPosition = new Vector3(0, -350);
            textDescription.GetComponent<RectTransform>().sizeDelta = new Vector2(900, 300);

            Texture2D texture = new Texture2D(2, 2);

            var getIndexOfImage = textures.Where(x => x.name == values[i].ShortName).First();
            bool isLoaded = texture.LoadImage(getIndexOfImage.EncodeToPNG());

            GameObject findImageByName = GameObject.Find(image.name);
```

```
if (isLoaded)
{
    findImageByName.GetComponent<RawImage>().texture = texture;
}

GameObject mainText = GameObject.Find(textTitle.transform.name);

mainText.GetComponent<Text>().fontSize = 72;
mainText.GetComponent<Text>().text = $"{values[i].Name}";
mainText.GetComponent<Text>().color = UnityEngine.Color.white;
mainText.GetComponent<Text>().alignment = TextAnchor.MiddleLeft;

GameObject descriptionText = GameObject.Find(textDescription.transform.name);

descriptionText.GetComponent<Text>().fontSize = 40;
descriptionText.GetComponent<Text>().text = $"{values[i].Description[0..150]}...";
descriptionText.GetComponent<Text>().color = UnityEngine.Color.white;

button.GetComponentInChildren<Text>().fontSize = 42;
button.GetComponentInChildren<Text>().text = "Більше";
button.GetComponentInChildren<Text>().color = UnityEngine.Color.white;
button.GetComponent<Image>().sprite = buttonSprite;

string shortName = values[i].ShortName;
if (values[i].IsArSupported)
{
    button.GetComponent<Button>().onClick
        .AddListener(delegate { new ChangeScene().MoveToScene(1, $"{shortName}"); });
}
else
{
    button.GetComponent<Button>().onClick
        .AddListener(delegate { new ChangeScene().MoveToScene(2, $"{shortName}"); });
}

if (values[i].IsArSupported)
{
    var arSupported = DefaultControls.CreateText(new DefaultControls.Resources());
    arSupported.transform.name = "ArSupportedText";
```

```
            arSupported.transform.tag = "GameController";
            arSupported.transform.SetParent(panel.transform);

            arSupported.GetComponent<RectTransform>().localPosition = new Vector3(470, 430);
            arSupported.GetComponent<RectTransform>().sizeDelta = new Vector2(75, 75);

            GameObject arSupportedLabel = GameObject.Find(arSupported.transform.name);

            arSupportedLabel.GetComponent<Text>().fontSize = 40;
            arSupportedLabel.GetComponent<Text>().text = $"AR";
            arSupportedLabel.GetComponent<Text>().color = UnityEngine.Color.white;
        }
    }
  }

  void Start()
  {
    if (ScenesSharedData.Values is null)
    {
      CreateObjects(SitesData.GetData());
      ScenesSharedData.Values = SitesData.GetData();
    }
    else
    {
      CreateObjects(ScenesSharedData.Values);
    }
  }
}
```