
GUIDE DE DÉVELOPPEMENT

Groupe 2

AL BAKALI Roudaina
EL MOUDNI Zakaria
KOC Damian
ROBALINO ROBLES Jeremy
YAZBECK John



05 MAI 2025
HELB IODA B2 2024-2025
Cours : Gestion de projets 1

Guide de Développement

Table des matières

1	Introduction.....	2
2	Outils	2
3	Installation	2
4	Explications techniques sur le développement.....	2
4.1	Développement du Front-End	2
4.1.1	Création de scènes	2
4.1.2	Mise en place d'un canvas.....	4
4.1.3	Options de traduction de l'application.....	4
4.2	Développement de la carte et options.....	5
4.2.1	Affichage de la position GPS sur la carte Mapbox.....	6
4.2.2	Chargement et affichage dynamique de planètes depuis un fichier JSON	6
4.2.3	Contrôles tactiles pour zoom et déplacement sur la carte	7
4.3	Développement de l'AR et options	8
4.3.1	Affichage dynamique des planètes selon localisation GPS	8
4.3.2	Interaction tactile pour supprimer les planètes.....	9
4.3.3	Rotation continue des planètes.....	9
5	Comment tester ?.....	10

Guide de Développement

1 Introduction

Ce guide détaille le processus de développement d'une application mobile intégrant de la réalité augmentée (AR), une carte interactive avec positionnement GPS, ainsi qu'une interface utilisateur multilingue. Le but est de permettre une installation facile, une compréhension technique du développement et un test fluide du projet.

2 Outils

1. Logiciels et plateformes :
 - Unity : version 2022.3.58f1
 - Unity Version Control pour la gestion du code
 - Visual Studio avec support Unity et C#
 - Visual Studio Code (pour éditer les pages HTML/CSS)
2. Environnement :
 - Systèmes d'exploitation : Windows / macOS / Linux.
 - Modules : AR Foundation (avec ARCore pour Android).
 - Capteurs utilisés : GPS, Boussole, Caméra.

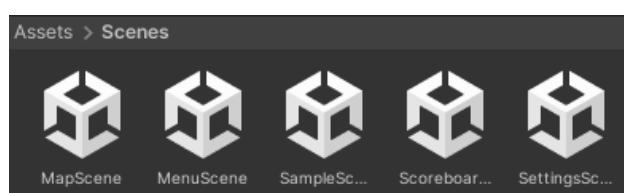
3 Installation

1. Ouvrir Unity Hub, importer le projet avec la bonne version (2022.3.58f1).
2. Configurer Unity Version Control (connexion requise).
3. Synchroniser les dernières modifications du dépôt.
4. Ouvrir la scène principale (MenuScene).
5. Lancer la scène en mode Play ou builder pour appareil mobile.

4 Explications techniques sur le développement

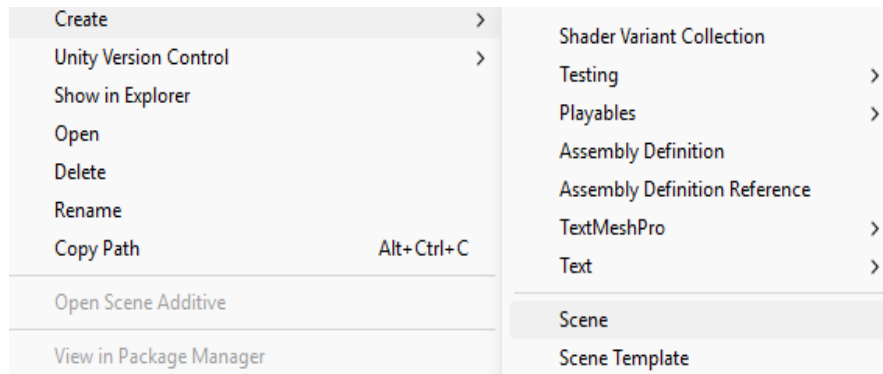
4.1 Développement du Front-End

4.1.1 Création de scènes

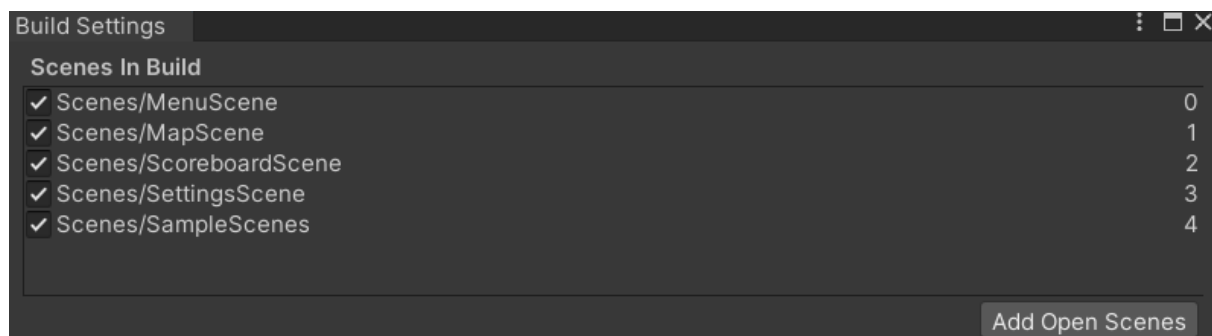


Les scènes permettent à l'utilisateur de changer de vue entre les différentes actions. Dans notre exemple, nous avons des scènes pour le Menu, dont le but principal est de rediriger l'utilisateur vers d'autres scènes comme Map (qui redirige vers SampleScene pour la gestion de la caméra et de l'AR), ScoreboardScene et SettingsScene.

La création se fait par clic :



Enfin, pour que la scène soit récupérée et utilisée dans le build, elle doit être prise en compte :

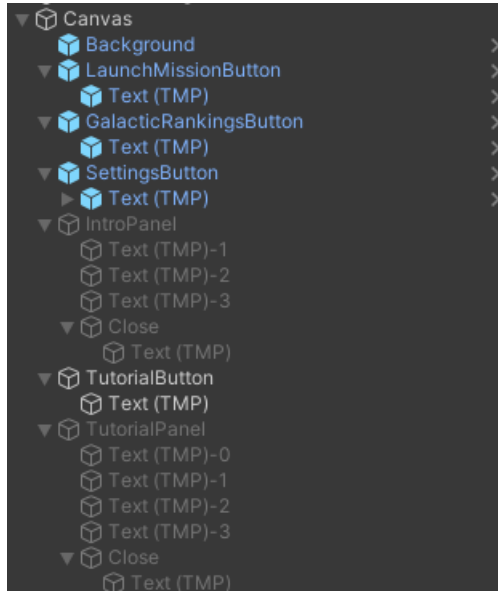


Création d'un script pour le passage entre scènes :

```
1  using UnityEngine;
2  using UnityEngine.SceneManagement;
3
4  public class MenuController : MonoBehaviour
5  {
6      public void LaunchMission()
7      {
8          SceneManager.LoadScene("MapScene");
9      }
10
11     public void GalacticRankings()
12     {
13         SceneManager.LoadScene("ScoreboardScene");
14     }
15
16     public void OpenSettings()
17     {
18         SceneManager.LoadScene("SettingsScene");
19     }
20 }
```

4.1.2 Mise en place d'un canvas

Canvas UI est utilisé lorsqu'on souhaite conserver une structure composée de boutons, pannels, etc. Dans le projet, l'un des canvas se présente de la façon suivante (exemple dans MenuScene) :



4.1.3 Options de traduction de l'application



L'application est capable d'adapter le langage de l'app entière avec le script suivant :

```
// Gestion de la sélection de la langue et de la mise à jour de l'interface
void SelectLanguage(string language)
{
    Debug.Log($"Langue sélectionnée : {language}");

    // Sauvegarder la langue choisie
    PlayerPrefs.SetString(LANGUAGE_KEY, language);
    PlayerPrefs.Save();

    switch (language)
    {
        case "French":
            TranslateToFrench();
            backgroundImage.sprite = backgroundFrench;
            break;

        case "Dutch":
            TranslateToDutch();
            backgroundImage.sprite = backgroundDutch;
            break;

        case "English":
            TranslateToEnglish();
            backgroundImage.sprite = backgroundEnglish;
            break;
    }

    // Retour aux boutons principaux après la sélection
    HideLanguagePanel();
}
```

```
// Méthodes de traduction
void TranslateToFrench()
{
    // Traduction des boutons dans toutes les scènes
    if (languageButtonText) languageButtonText.text = "Langues";
    if (soundButtonText) soundButtonText.text = "Son";
    if (creditsButtonText) creditsButtonText.text = "Crédits";
    if (gotomenuButtonText) gotomenuButtonText.text = "Aller au menu";
    if (creditsText) creditsText.text = "Équipe de développeurs:\r\n - AL BAKALI Roudaina\r\n - EL MOUDNI Zakaria\r\n - HOC Damian\r\n - ROBALINO ROBLES Jeremy\r\n - YAZBECK John\r\n";

    if (playButtonText) playButtonText.text = "Jouer";
    if (leaderboardButtonText) leaderboardButtonText.text = "Classement";
    if (settingsButtonText) settingsButtonText.text = "Paramètres";

    if (scoreboardGoToMenuButtonText) scoreboardGoToMenuButtonText.text = "Aller au menu";
}

void TranslateToDutch()
{
    if (languageButtonText) languageButtonText.text = "Talen";
    if (soundButtonText) soundButtonText.text = "Geluid";
    if (creditsButtonText) creditsButtonText.text = "Credits";
    if (gotomenuButtonText) gotomenuButtonText.text = "Ga naar menu";
    if (creditsText) creditsText.text = "Team van ontwikkelaars:\r\n - AL BAKALI Roudaina\r\n - EL MOUDNI Zakaria\r\n - HOC Damian\r\n - ROBALINO ROBLES Jeremy\r\n - YAZBECK John\r\n";

    if (playButtonText) playButtonText.text = "Spelen";
    if (leaderboardButtonText) leaderboardButtonText.text = "Ranglijst";
    if (settingsButtonText) settingsButtonText.text = "Instellingen";

    if (scoreboardGoToMenuButtonText) scoreboardGoToMenuButtonText.text = "Ga naar menu";
}

void TranslateToEnglish()
{
    if (languageButtonText) languageButtonText.text = "Languages";
    if (soundButtonText) soundButtonText.text = "Sound";
    if (creditsButtonText) creditsButtonText.text = "Credits";
    if (gotomenuButtonText) gotomenuButtonText.text = "Go to menu";
    if (creditsText) creditsText.text = "Team of developers:\r\n - AL BAKALI Roudaina\r\n - EL MOUDNI Zakaria\r\n - HOC Damian\r\n - ROBALINO ROBLES Jeremy\r\n - YAZBECK John\r\n";

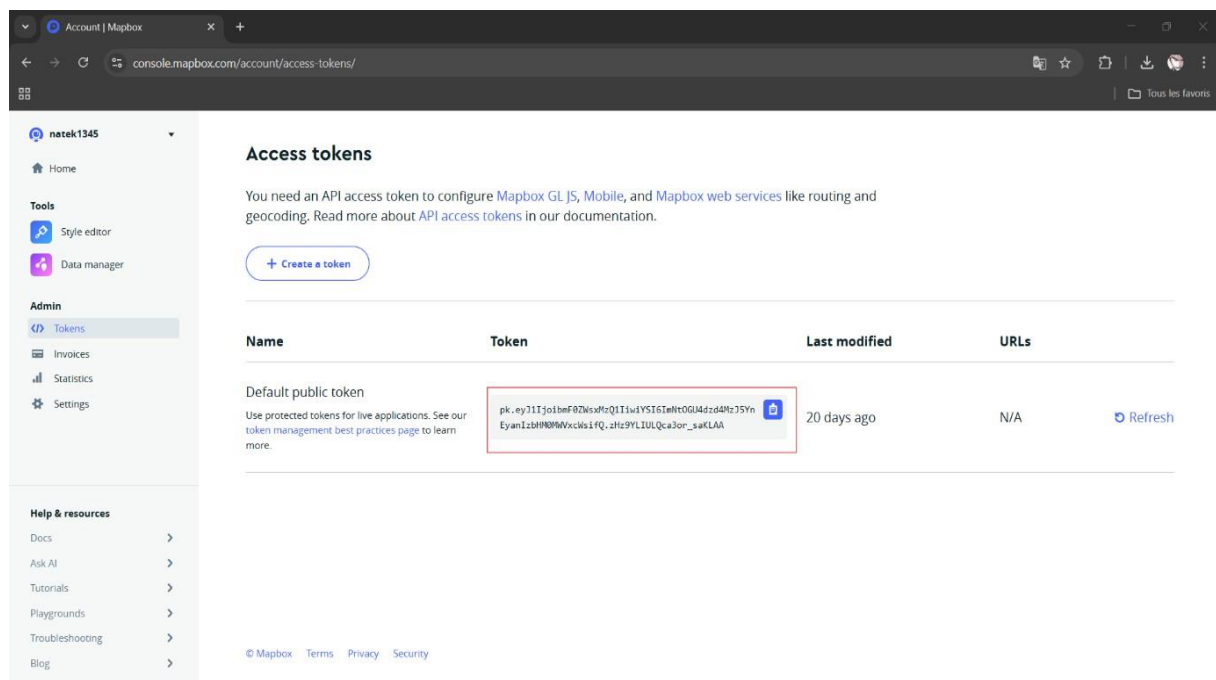
    if (playButtonText) playButtonText.text = "Play";
    if (leaderboardButtonText) leaderboardButtonText.text = "Scoreboard";
    if (settingsButtonText) settingsButtonText.text = "Settings";

    if (scoreboardGoToMenuButtonText) scoreboardGoToMenuButtonText.text = "Go to menu";
}
```

Pour ajouter une nouvelle langue, l'utilisateur doit simplement ajouter un nouveau bouton, faire la liaison entre tout les message TMP et d'ajouter la traduction.

4.2 Développement de la carte et options

Création d'un compte Mapbox pour avoir accès au tokens qui vont permettre d'implémenter la carte sur Unity.



The screenshot shows the Mapbox account page for user 'natek1345'. The 'Access tokens' section is active, displaying a table with one token: 'Default public token'. The token value is partially visible: 'pk.eyJ1IjoibF07WesVhQ1IiwidY5tG1wtoGJAdz04Mz05YnEyanIzBHMWVxcld5fQ.2Hz9YL1ULQca3or_sakLAA'. The token was created 20 days ago and has no URLs associated with it. A 'Refresh' button is available next to the token.

Name	Token	Last modified	URLs
Default public token	pk.eyJ1IjoibF07WesVhQ1IiwidY5tG1wtoGJAdz04Mz05YnEyanIzBHMWVxcld5fQ.2Hz9YL1ULQca3or_sakLAA	20 days ago	N/A

4.2.1 Affichage de la position GPS sur la carte Mapbox

Le script utilise le GPS du téléphone (via Input.location) pour :

- Démarrer le service de localisation (StartLocationService()),
- Récupérer les coordonnées actuelles de l'utilisateur,
- Afficher un marqueur de position (si locationMarkerPrefab est défini) à l'endroit correspondant sur la carte Mapbox.

```
1 référence
IEnumerator StartLocationService()
{
    if (!Input.location.isEnabledByUser)
    {
        Debug.LogWarning("Localisation désactivée.");
        yield break;
    }

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

    if (Input.location.status != LocationServiceStatus.Running)
    {
        Debug.LogWarning("Impossible d'obtenir la position.");
        yield break;
    }

    Debug.Log("GPS prêt.");
}
```

```
1 référence
void UpdateUserLocation()
{
    if (locationMarkerInstance != null && Input.location.status == LocationServiceStatus.Running)
    {
        double lat = Input.location.lastData.latitude;
        double lon = Input.location.lastData.longitude;
        Vector2d userLatLon = new Vector2d(lat, lon);
        Vector3 worldPos = map.GeoToWorldPosition(userLatLon, true);
        locationMarkerInstance.transform.position = worldPos;
    }
}
```

4.2.2 Chargement et affichage dynamique de planètes depuis un fichier JSON

Le script charge un fichier planets.json dans le dossier StreamingAssets, parse les données, puis place des marqueurs personnalisés (avec un prefab) sur la carte aux coordonnées fournies.


```
1 référence
IEnumerator LoadPlanetMarkers()
{
    string path = Path.Combine(Application.streamingAssetsPath, "JSON/planets.json");

    UNITY_ANDROID && !UNITY_EDITOR
    UnityWebRequest www = UnityWebRequest.Get(path);
    yield return www.SendWebRequest();

    if (www.result != UnityWebRequest.Result.Success)
    {
        Debug.LogError("Erreur lecture JSON : " + www.error);
        yield break;
    }

    string json = www.downloadHandler.text;

    if (!File.Exists(path))
    {
        Debug.LogError("Fichier JSON non trouvé : " + path);
        yield break;
    }

    string json = File.ReadAllText(path);

    PlanetDataList dataList = JsonUtility.FromJson<PlanetDataList>(json);
    if (dataList == null || dataList.planets == null)
    {
        Debug.LogError("Erreur parsing JSON.");
        yield break;
    }

    planetList = dataList.planets;
    PlaceMarkersOnMap();
}
```

4.2.3 Contrôles tactiles pour zoom et déplacement sur la carte

Le script permet d'interagir avec la carte en :

- Zoomant avec un pinch à deux doigts (HandleTouchZoom()),
- Faisant un pan (glisser) avec un seul doigt (HandleTouchPan()),
- Limitant le zoom entre minZoom et maxZoom.

```
1 référence
void HandleTouchZoom()
{
    if (Input.touchCount == 2)
    {
        Touch t0 = Input.GetTouch(0);
        Touch t1 = Input.GetTouch(1);

        Vector2 prevT0 = t0.position - t0.deltaPosition;
        Vector2 prevT1 = t1.position - t1.deltaPosition;

        float prevMag = (prevT0 - prevT1).magnitude;
        float currMag = (t0.position - t1.position).magnitude;
        float diff = currMag - prevMag;

        zoom += diff * 0.002f;
        zoom = Mathf.Clamp(zoom, minZoom, maxZoom);
        map.UpdateMap(map.CenterLatitudeLongitude, zoom);
    }
}
```



```
1 référence
void HandleTouchPan()
{
    if (Input.touchCount == 1)
    {
        Touch touch = Input.GetTouch(0);
        if (touch.phase == TouchPhase.Moved)
        {
            Vector2 delta = touch.deltaPosition;
            float zoomFactor = Mathf.Pow(2, zoom - minZoom);

            double latOffset = -delta.y * panSpeed / (zoomFactor * 3.0);
            double lonOffset = -delta.x * panSpeed / (zoomFactor * 3.0);

            Vector2d newCenter = new Vector2d(
                map.CenterLatitudeLongitude.x + latOffset,
                map.CenterLatitudeLongitude.y + lonOffset
            );

            map.UpdateMap(newCenter, zoom);
        }
    }
}
```

4.3 Développement de l'AR et options

4.3.1 Affichage dynamique des planètes selon localisation GPS

Cette fonctionnalité permet de charger et positionner dynamiquement des modèles 3D de planètes en réalité augmentée, selon la position GPS initiale de l'utilisateur.

Étapes d'installation

- Importer et configurer le SDK de géolocalisation natif d'Unity.
- Utiliser un fichier JSON pour stocker les coordonnées des planètes.
- Charger les modèles de planètes depuis le dossier Resources/Prefabs.

Script essentiel : LocationManager, Ce script est responsable de l'initialisation GPS, du chargement JSON, et du placement précis des planètes.

```
1 référence
void InstantiatePlanets()
{
    Vector3 origin = Camera.main.transform.position;
    Debug.Log($"📍 Origine (caméra) : {origin}");

    foreach (ARPlanet planet in planets)
    {
        double latDiff = planet.location.latitude - initialLocation.latitude;
        double lonDiff = planet.location.longitude - initialLocation.longitude;

        double offsetZ = latDiff * metersPerDegreeLat * distanceScale;
        double offsetX = lonDiff * metersPerDegreeLat * Math.Cos(initialLocation.latitude * Math.PI / 180.0) * distanceScale;

        Vector3 offset = new Vector3((float)offsetX, 0, (float)offsetZ);
        Vector3 planetPosition = origin + offset;

        if (planet.prefab != null)
        {
            GameObject planetInstance = Instantiate(planet.prefab, planetPosition, Quaternion.identity);
            instantiatedPlanets[planet.name] = planetInstance;
            Debug.Log($"✅ {planet.name} instanciée à {planetPosition} (offset: {offset})");
        }
        else
        {
            Debug.LogError($"❌ Préfab manquant pour {planet.name}");
        }
    }
}
```

4.3.2 Interaction tactile pour supprimer les planètes

Cette fonctionnalité permet à l'utilisateur de supprimer une planète en réalité augmentée en cliquant directement dessus.

Étapes d'installation

- Ajouter un Collider à chaque modèle de planète.
- Attacher le script PlanetClickHandler à chaque prefab de planète.

Script essentiel : PlanetClickHandler, Ce script détecte le clic ou la pression tactile sur une planète et gère sa suppression.

```
Message Unity | 0 références
void Update()
{
    if (Input.touchCount > 0 && Input.touches[0].phase == TouchPhase.Began)
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.touches[0].position);
        RaycastHit hit;
        if (Physics.Raycast(ray, out hit) && hit.transform == transform)
        {
            RemovePlanet();
        }
    }

    if (Input.GetMouseButtonDown(0))
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit hit;
        if (Physics.Raycast(ray, out hit) && hit.transform == transform)
        {
            RemovePlanet();
        }
    }
}

2 références
void RemovePlanet()
{
    Debug.Log($"D {gameObject.name} cliquée, suppression en cours...");
    Destroy(gameObject);
}
```

4.3.3 Rotation continue des planètes

Cette fonctionnalité ajoute une rotation continue réaliste à chaque planète en réalité augmentée pour un rendu visuel dynamique.

Étapes d'installation

- Attacher le script PlanetRotator aux prefabs des planètes.
- Régler la vitesse de rotation via l'inspecteur Unity.

Script essentiel : PlanetRotator, Ce script assure une rotation régulière autour de l'axe vertical (Y).

```
Script Unity (7 références de ressources) | 0 références
public class PlanetRotator : MonoBehaviour
{
    [Tooltip("Vitesse de rotation en degrés par seconde.")]
    public float rotationSpeed = 10f;

    Message Unity | 0 références
    void Update()
    {
        // Effectue une rotation autour de l'axe Y
        transform.Rotate(Vector3.up, rotationSpeed * Time.deltaTime);
    }
}
```

5 Comment tester ?

1. Play mode dans Unity pour tests initiaux.
2. Build mobile (Android conseillé) pour tests AR réels.
3. Tester les fonctionnalités suivantes :
 - Navigation entre scènes.
 - Affichage carte et position GPS.
 - Chargement des planètes depuis JSON.
 - Collecter des planètes et vérifier le résultat dans le classement.
 - Traduction de l'interface et autres gestions de paramètres.