Immersive Media Programming – Lab 5 – AR Plane Tracking, XR Interaction Toolkit and User Interface

In this lab we'll practice implementing AR plane tracking and XR Interaction Toolkit features, with a little bit UI building.

NOTE: This lab is probably more demanding than lab 4. Even though there is more time to finish than normally, please start early and ask for help if you get stuck!

The name of the lecture file related to this lab:

- Plane detection: "**2022/03/30 Immersive Media Programming**" (Week 5)
- User Interface: "**2022/04/04 Immersive Media Programming**" (Week 5)
- XR Interaction Toolkit: "**2022/04/06 Immersive Media Programming**" (Week 6)

Submission instructions:

- Record a video of your mobile device screens for each task (Tasks 1, 2, 3). The video names should follow the task names. For example, if you take two videos for Task 2, name them as: Task2_1.mp4, Task2_2.mp4
- Place all videos in a separate folder OUTSIDE the project folder.
- Make a build and include the APK file in the project root.
- Close Unity.
- Make a backup copy of your project folder.
- Remove unnecessary folders from your project folder: Library, Logs, obj.
- Zip the project folder AND the video folder into a single ZIP file.
- Upload the ZIP file to your Google Drive shared folder.
- Submit a link to the shared folder in AjouBB.

Deadline: April 19 at 23:59

Maximum points: 19p

Task 1: Setting up plane tracking (2p)

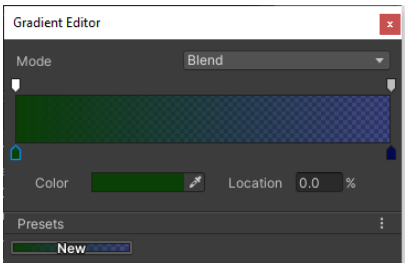
NOTE: you don't need a record a video of this task because it is included in Task 2.

Scene and default plane setup (1p)

- Create a new Unity 3D project, and complete the basic AR setup.
- Rename the scene to "Task1"
- Set up an AR Plane Manager with an AR Default Plane.
- Test to ensure that it works.

Create customized plane (1p)

- Follow the AR plane tracking lecture to create your own plane.
  - Use a gradient color
  - Create your own material for the plane
- Save the plane as a prefab, and assign it to AR PlaneManager.
- Test the scene.



Task 2: Tap to raycast (5p)

Demo: Lab5_Task2.mp4.

Duplicate Task1 scene and rename as "Task2".

- Import two of your favorite game objects from your previous labs (or from Asset Store / Sketchfab). (0.5p)
  - Add Rigidbody and collider components to the prefabs if they don't have them already.

- Then implement the following features:
  - When a plane is tapped, instantiate randomly one of the game objects above the plane (so that they fall down a bit). (1.5p)
    - Use ARRaycast for this!
    - Do not perform ARRaycast if Physics.Raycast() hits one of the instantiated gameobjects (see next).
  - When a gameobject is tapped, a special effect happens. (3p)
    - Use Physics.Raycast() for this.
    - Different effect for each prefab type.
    - The effects should be more complex than just changing some property of the gameobject.
    - Play an appropriate sound effect.

Task 3: Interactive AR museum (7p)

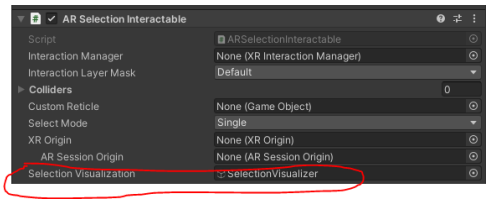
Demo: Lab5_Task3.mp4.

In this task you'll create some gameobjects with basic interaction and annotation features using the XR Interaction Toolkit. You will also build a simple UI to choose between the objects to instantiate.

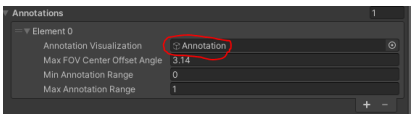
Create new scene "Task 3".

- Scene setup (1p)
  - Set up AR with AR plane manager and AR raycast manager.
    - Use your customized plane from Task1 as the plane prefab.
  - Import 2 prefabs from your previous labs
    - Alternatively you can make make copies of prefabs in Task2 and remove any unnecessary scripts.
    - Remove Rigidbodies.

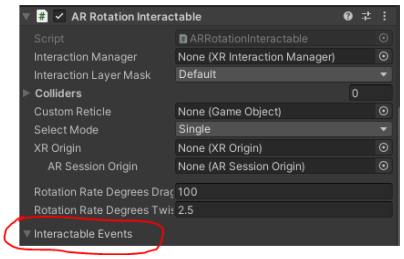
- For each prefab:
  - Create a different selection visualizer (see the XR Interaction Toolkit lecture) (1p)



  - Attach the following components: (0.5p)
    - AR Selection Interactable
      - Assign the selection visualizer
    - AR Rotation Interactable
    - AR Scale Interactable
    - AR Annotation Interactable



  - Design a nice annotation (s) (1p)
    - Create something different than in the lecture video :)
    - Different annotation for each prefab!
    - Assign the annotation to AR Annotation Interactable.



  - Play different sounds when the prefab is selected (use Interactable Events) (0.5p)

- Create a simple UI with a dropdown menu or buttons to choose the prefab to be instantiated. (1p)

- Write a script to instantiate the selected prefab at touch position. (2p)
  - Make sure that only one gameobject visible (i.e. destroy or hide the other one)!
  - See Tip 3 below about preventing instantiating when touching the UI.

NOTE: reasons why we are not using some interactables:
- AR Placement Interactable: because you are placing the objects manually in script depending on which prefab is selected.
- AR Translation Interactable: because this does not seem to work when instantiating a prefab manually.

Tip 1: Make sure to select "Both" for Active Input Handling under Project Settings > Player:

Tip 2: There is no AudioListener in the AR Camera, so you will need to add it to be able to hear the audio.

Tip 3: When the user interacts with a UI component, the selected prefab should NOT be instantiated. You can use this method to check if the user touched a UI element (parameter is the current touch position):

- if this returns true, don't instantiate a prefab because the user tapped on the UI!

```
// reference: https://www.youtube.com/watch?v=NdrvihZhVqs
private bool isOverUI(Vector2 position)
{
    // if the pointer is over a gameobject, it is not over a UI element.
    if (EventSystem.current.IsPointerOverGameObject())
        return false;

    // Check if "position" over one or more UI elements.
    // EventData contains information about mouse/touch events.
    // EventSystem.current is the active event system that is handling all UI events.
    PointerEventData eventData = new PointerEventData(EventSystem.current);
    eventData.position = new Vector2(position.x, position.y);

    // EventSystem's raycast methods do raycasts against UI elements (they don't have colliders
    // so we cannot use Physics.Raycast()).
    // RaycastAll() returns all hit information of UI elements that were hit.
    List<RaycastResult> results = new List<RaycastResult>();
    EventSystem.current.RaycastAll(eventData, results);

    // if hit count > 0 then we hit some UI elements
    return results.Count > 0;

}
```

Task 4: More UI building (3p)

Create new scene "Task4" that shows a main menu. The main menu has:

- Buttons for opening scenes Task1, Task2 and Task3.
- A Quit button to exit the application / stop the play mode in Unity.

In Tasks 1-3, add a Back button to return back to the main menu.

- Use the isOverUI() method (see above) in Task 2 to prevent other interactions when touching the UI!

Tip: Use SceneManager.LoadScene() to change the scene either by scene name or by scene index in Build Settings.

Task 5: Reflection (2p)

Write a short reflection text based on your lab experience (e.g. a text file or a Word file) using the questions below. The positivity or negativity of your answers does not affect scoring.

Please answer to these questions (at least):

- What did you learn? What did you know already?
- What was difficult in the assignment? How did you overcome it?
- What was good about the assignment?
- Do you have any suggestions to improve the assignment? Please tell us!

Save your reflections as Reflections.txt/docx to the root of your project.