Immersive Media Programming – Lab 4 – AR Image Tracking

In this lab, we will begin working with AR and practice the following:

- Setting up AR project
- Using ARTrackedImageManager
- Subscribing to the trackedImagesChanged event to run code when images are detected.
- Raycasting to add some interaction.

Submission instructions (UPDATED):

- 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.

IMPORTANT: Make sure that you don't have any spaces or special characters (e.g. 한글) in the project path or file names!

Deadline: April 8 at 23:59

Maximum points: 13p

Task 1: Image tracking setup and test (3p)

a) Create a new Unity project using the AR project template. Then do the following: (0.5p)

- Rename the sample scene to Task1.
- Remove the following components from AR Session Origin
  - AnchorCreator
  - ARRaycastManager
  - ARAnchorManager
  - ARPlaneManager
- File > Build Settings… > switch platform to Android
- Edit > Project Settings > XR Plug-in Management > on Android tab, check ARCore

b) Export and import prefab(s) (1p)

- Open your Lab 1 project and save two game objects from Lab 1 Task 2 as prefabs (if they are not a prefab already).
- Then, export the prefabs by Assets > Export Package > check the prefabs and all other assets that are related to them (scripts, materials, textures etc) > Export
- Then in the Lab 4 project, choose Assets > Import Package > Custom package, and choose the package that you created.
- If needed, scale down the imported prefabs to be suitable for AR.
  - Ex) around 10-20cm size will be quite good for an AR object.
- Save the exported package in your project root.

c) Set up a Reference Image Library (1p)

- Find two images from the internet that can be used as tracked images.

    - Images with high contrast and unique features are good.
- Use the arcodeimg tool to check quality of the images (run at command prompt, cmd.exe):

  arcoreimg.exe eval-img --input_image_path=c:\Path\To\marker.png

    - If the score is < 75, then find a better image.
    - Take screenshots of passed tests and store them to your project root.
- Create a reference image library, add the images to the library, give them names and set their physical sizes.

d) Set up AR Tracked Image Manager, and assign the ReferenceImageLibrary and one of your prefabs to it. Then build and run the scene to test it. (0.5p)

- If the prefab seems to be too large or small, scale it and test again.

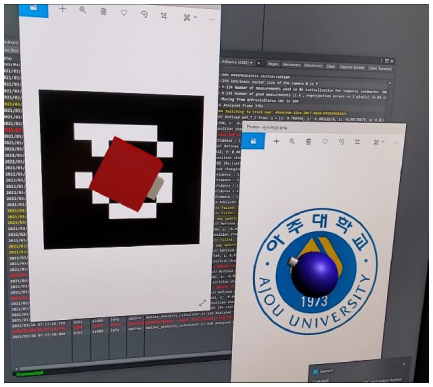
Task 2: Spawning objects on markers (3p)

Demo: Lab4_Task2.mp4

Make a copy of the Task1 scene, rename to Task2. Then set Tracked Image Prefab to None.

- Write a script instantiates your Lab 1 prefabs on the markers (different prefab on each marker).
- If your prefabs don't have a clearly identifiable front side, then modify them so that you can see where they are "looking" at (see pictures below).
- In the script, handle (see lecture and sample code):
  - added, updated and removed lists of AR Tracked Images.
  - Tracking, Limited and None tracking states.
- The gameobjects should only be active when the markers are visible.
- Make the gameobjects rotate.

Ex) both gameobjects have a front side that can be easily seen:



Task 3: Looking and raycasting (5p)

Demo: Lab4_Task3.mp4

Make a copy of the Task2 scene, rename to Task3. Then:

- Make the gameobjects look at each other, but only when the target gameobject is active (*activeInHierarchy* is true). (1p)
- When the user taps the screen, do a raycast (Physics.Raycast) from the tap position. (1p)
  - If the raycast hits one of the gameobjects,
    - Play a sound effect (1p)
      - Different sound effect for each gameobject.
      - Make sure to add both AudioListener and AudioSource components!
    - Do different things to the gameobjects if the ray hits them. (2p)
      - Different effect for each gameobject!
      - Try to make your effects different than what you created in Lab 3 :)

Hint: you can detect a tap as:

a) a mouse click (Input.GetMouseButtonDown()), OR

b) a touch (Input.GetTouch())

- In this case you also need to check the number of touches and touching phase (see the link).

Task 4: 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.