GERMAN UNIVERSITY IN CAIRO

Augmented Reality App for Egyptian Museum

Prepared by: Hadwa Pasha, 34-0476
Noha Hamid, 34-9697
Safya Elzayat, 34-4643
Ziad Omar, 34-4824
Ayman Hamdy, 34-12777
Antoin Maged, 34-9123

December 16, 2018

DESCRIPTION

Objective

It’s an AR mobile App for the Egyptian Museum in Tahrir. Since it is difficult to have access to the artifacts in the actual museum, we are going to use images from known artifacts in the museum and we will simulate the museum to be able to have users move fast in the development activity to explore the museum artifacts in an interactive, engaging way and learn about the different monuments inside.

Project Features

- We will use Unity as our building development platform.
- We will use Vuforia as our AR platform to simulate the museum environment through Vuforia’s Feature: Image Targets Detection. We will use this feature to put the images of the artifacts in Vuforia’s database then use them in our project to be detected by the mobile.
- After detection, the image 3D model will come out linearly to be visible to the user with all the real info about this artifact as shown below.
- We will simulate an empty room containing these images in a specific environment required for good detection.
- Also we will provide for the user the opportunity to interact with the monument to rotate it and see all its features from all sides.
- Some of the artifacts will only have info on detection and some will have their own 3D models appearing based on the availability of these models.
- The app will have its own UI on start and appearing hints to help the user understand how to use it and how to explore the museum and interact with the different artifacts.
- Also the app will notify the user once each artifact is detected successfully and a particle system will be used to make the appearing of the model more appealing and interesting for the user.

IMPLEMENTATION

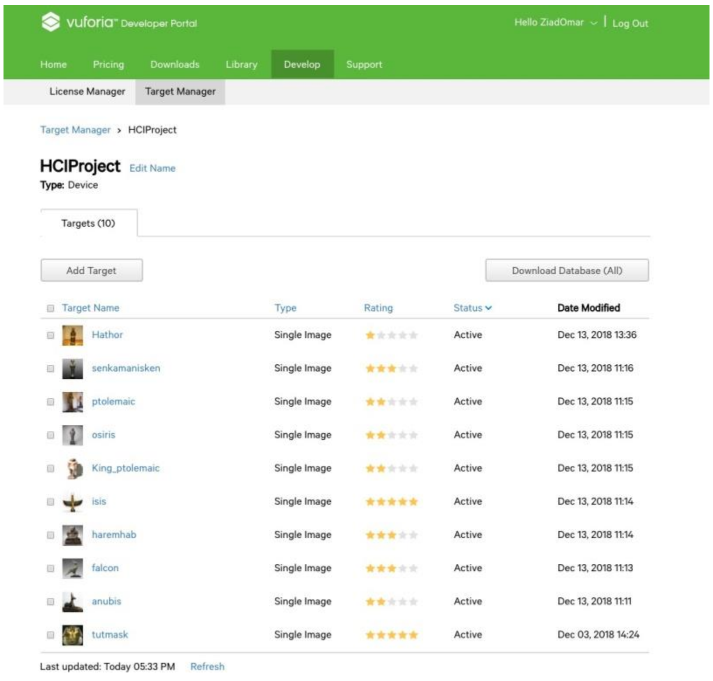
Vuforia

Vuforia is a cross-platform Augmented Reality (AR) and Mixed Reality (MR) application development platform. Vuforia is widely used by many platforms and famous applications to create a new experience of engaging their users in AR. Vuforia is special in its wide range of features that help a lot in making the AR experience perfect and reaching to the users' expectations. Vuforia gives the developers the opportunity to attach digital content to specific objects through 6 main AR features.
One of them is,

Image Targets: This is the easiest feature that allows the recognition of at images on a cardboard or pages or photographs. The SDK detects and tracks the features that are naturally found in the image itself by comparing these natural features against a known target resource database. Once the Image Target is detected, the SDK will track the image as long as it is at least partially in the camera's field of view. This feature is the feature we used in our project.

Vuforia Database

We upload the images used in models detection to the target manager of the Vuforia development portal, then we download the database as a unity package and import it in our unity project.



HCIProject
Type: Device

Target Name	Type	Rating	Status	Date Modified
Hathor	Single Image	★☆☆☆☆	Active	Dec 13, 2018 13:36
senkamanisken	Single Image	★★★☆☆	Active	Dec 13, 2018 11:16
ptolemaic	Single Image	★★☆☆☆	Active	Dec 13, 2018 11:15
osiris	Single Image	★★☆☆☆	Active	Dec 13, 2018 11:15
King_ptolemaic	Single Image	★★☆☆☆	Active	Dec 13, 2018 11:15
isis	Single Image	★★★★★	Active	Dec 13, 2018 11:14
haremhab	Single Image	★★★☆☆	Active	Dec 13, 2018 11:14
falcon	Single Image	★★★☆☆	Active	Dec 13, 2018 11:13
anubis	Single Image	★★☆☆☆	Active	Dec 13, 2018 11:11
tutmask	Single Image	★★★★★	Active	Dec 03, 2018 14:24

Last updated: Today 05:33 PM

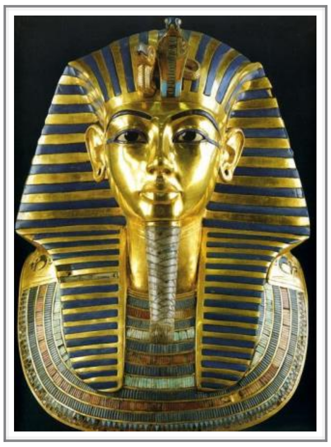
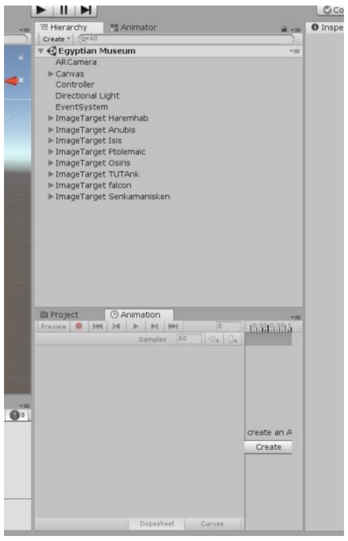
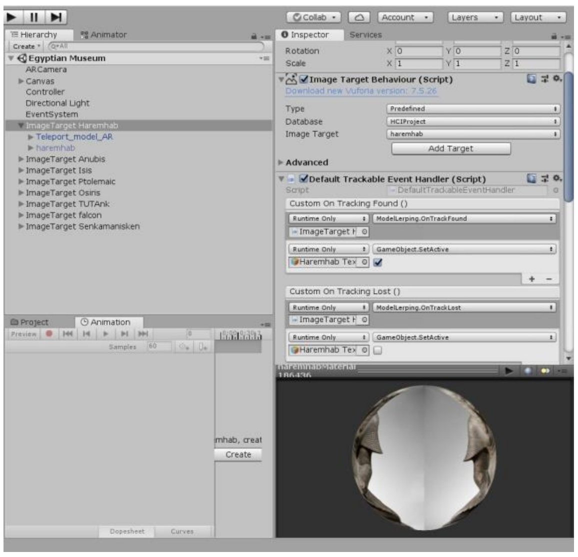


Image Target

We create new image target for each image inserted in the database and insert the corresponding model as a child for the target image as well as the teleport model that is responsible for the lerping effect.



Egyptian Museum
ARCamera
Canvas
Controller
Directional Light
EventSystem
ImageTarget Haremhab
ImageTarget Anubis
ImageTarget Isis
ImageTarget Ptolemaic
ImageTarget Osiris
ImageTarget TUTAnk
ImageTarget falcon
ImageTarget Senkamanisken



Egyptian Museum
ARCamera
Canvas
Controller
Directional Light
EventSystem
ImageTarget Haremhab
Teleport_model_AR
haremhab
ImageTarget Anubis
ImageTarget Isis
ImageTarget Ptolemaic
ImageTarget Osiris
ImageTarget TUTAnk
ImageTarget falcon
ImageTarget Senkamanisken

Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Image Target Behaviour (Script)
Download new Vuforia version: 7.5.26
Type Predefined
Database HCIProject
Image Target haremhab
Advanced
Default Trackable Event Handler (Script)
Script DefaultTrackableEventHandler
Custom On Tracking Found ()
Runtime Only ModelLerping.OnTrackFound
ImageTarget H
Runtime Only GameObject.SetActive
Haremhab Tex ✓
Custom On Tracking Lost ()
Runtime Only ModelLerping.OnTrackLost
ImageTarget H
Runtime Only GameObject.SetActive
Haremhab Tex ☐
HaremhabMaterial
186436

Animation

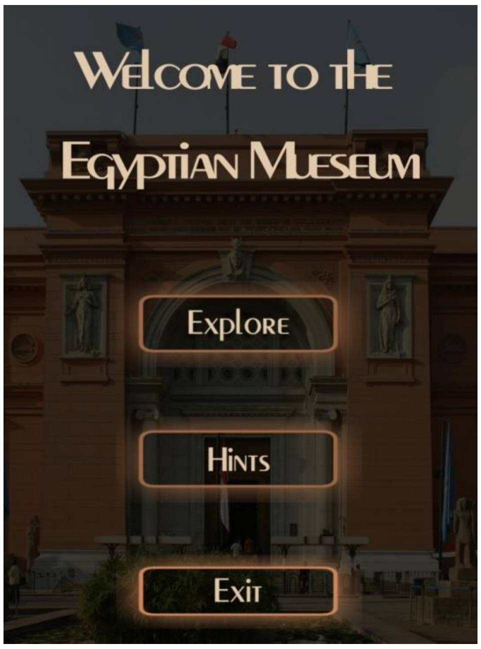
We created an animation clip for each model to rotate around itself when the user touches the model.

Sounds

We created an audio source and added to it a sound clip to be played on awake and loop as background music to the app to give the user the feel of being inside the ancient Egyptian environment.

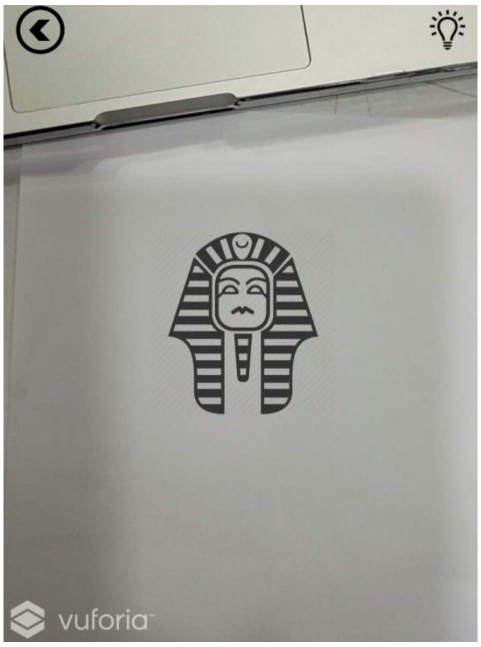
UI

There is a main screen with three buttons Explore, Hints and Exit.

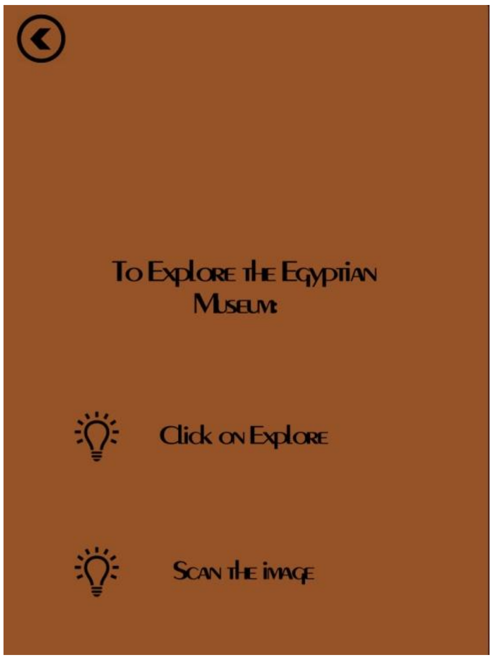


<u>Explore</u>: Take us to scanning screen.

Scan Screen: where the user will see an icon guiding him to scan the images he want , a bulb icon if clicked will remind him of what should he do and a back arrow to get back to the main screen.



Hints: Tell the user how to use the app.


To Explore the Egyptian Museum:

Click on Explore

Scan the image

Exit: to quit the app.

Scripts

There are 2 main scripts, one is responsible for user interaction with the model and the other is responsible for each model's detection and animation. **(ModelLerping.cs & TouchObject.cs)**

ModelLerping.cs

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

public class ModelLerping : MonoBehaviour {


    public GameObject Monument; public
    GameObject TeleportModel; Vector3
    ModelStartPos;
    public Vector3 ModelEndPos;

    public GameObject ScanIcon;

    // Use this for initialization void
    Start() {

        TeleportModel.SetActive(false);
        ModelStartPos = Monument.transform.localPosition;
        Monument.GetComponent<Animator>().enabled = false;
        ScanIcon.SetActive(true);
    }

    // Update is called once per frame void
    Update() {

    }

    public void OnTrackFound()
    {
        TeleportModel.SetActive(true);
        ScanIcon.SetActive(false);
        StartCoroutine("WaitForGreen");

    }
    public void OnTrackLost()
    {
        Monument.GetComponent<Animator>().enabled = false;
        ScanIcon.SetActive(true);
    }


    IEnumerator WaitForGreen()
    {
        yield return new WaitForSeconds(2f);
        Monument.transform.localPosition = ModelStartPos;
        Monument.SetActive(true); StartCoroutine(MoveObject(ModelStartPos,
        ModelEndPos));
    }

    IEnumerator MoveObject(Vector3 startPos, Vector3 endPos)
    {
        float progress = 0.0f;
        float speed = 1f;

        while (progress < 1.0f)
        {
            Monument.transform.localPosition = Vector3.Lerp(startPos, endPos, progress);

            yield return new WaitForSeconds(0.1f);
            progress += Time.deltaTime * speed;
        }
        Monument.transform.localPosition = endPos;
        StartCoroutine("WaitForObject");
    }

    IEnumerator WaitForObject()
    {
        yield return new WaitForSeconds(2f);
        TeleportModel.SetActive(false);
        Monument.GetComponent<Animator>().enabled = true;
    }
}
```

TouchObject.cs

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

public class TouchObject : MonoBehaviour { public

     GameObject main;
     public GameObject scan;
     public GameObject rules;
     public GameObject hints;

     public GameObject ModelsText;


     // Use this for initialization void
     Start()
     {
          main.SetActive(true);
          rules.SetActive(false);
          scan.SetActive(false);
          hints.SetActive(false);

          ModelsText.SetActive(false);
     }

          // Update is called once per frame void
          Update()
     {

          if ((Input.touchCount > 0 && Input.touches[0].phase == TouchPhase.Began) || Input.GetMouseButtonDown(0)
|| Input.GetKeyDown(KeyCode.A))
          {

               Ray ray = Camera.main.ScreenPointToRay(Input.GetTouch(0).position);
               //Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition); RaycastHit
               hit;

               if (EventSystem.current.IsPointerOverGameObject(0)) return; if

               (Physics.Raycast(ray, out hit, 1000))
               {
                    print("1");
                    if (hit.transform.tag == "Monument")
                    {
                         print("2"); hit.transform.gameObject.GetComponent<Animator>().SetTrigger("Rotate");
                    }


               }
          }
     }

     public void goToExit()
     {
          // save any game data here #if
        UNITY_EDITOR
          // Application.Quit() does not work in the editor so
          // UnityEditor.EditorApplication.isPlaying need to be set to false to end the game
          UnityEditor.EditorApplication.isPlaying = false;
        #else
                    Application.Quit();
       #endif
     }

     public void goToExplore()
     {
               scan.SetActive(true);
               main.SetActive(false);
               rules.SetActive(false)
               hints.SetActive(false);
               ModelsText.SetActive(true);
     }
```

```
public void goToRules()
{
    print("d");
    scan.SetActive(false);
    main.SetActive(false);
    rules.SetActive(true);
    hints.SetActive(false);

    ModelsText.SetActive(false);
}

public void goToHint()
{
    scan.SetActive(false);
    main.SetActive(false);
    rules.SetActive(false);
    hints.SetActive(true);

    ModelsText.SetActive(false);
}
public void backToMain()
{
    scan.SetActive(false);
    main.SetActive(true);
    rules.SetActive(false);
    hints.SetActive(false);

    ModelsText.SetActive(false);
}

public void backToScan()
{
    scan.SetActive(true);
    main.SetActive(false);
    rules.SetActive(false);
    hints.SetActive(false);


    ModelsText.SetActive(true);
}
```