**** Credit to MatthewHallberg, we followed his tutorial and created this workshop based on his work. Check out his youtube channel for more tutorials: https://www.youtube.com/channel/UCIm2DY6pj3ygKoKhEVr7KFw**

Getting Started

1. Create a new 3D project
2. Go to https://github.com/uwvr/AR-Project-Workshop to download assets (complete scripts are also available here)

Setting up the environment

1. Adjust the camera view to be the same dimensions as phone (16:9 is standard)
2. Attach "MainCamera" tag to the camera already present in the scene
3. Make sure camera position is (0, 0, -10)
4. Create a plane
5. Reset transform component
6. Set x rotation to (90, 180, 0) for android, (-90, 0, 0) for ios.
7. Adjust the x and y scale of the plane so it fills the camera view (x scale needs to be negative for ios)
8. Attach script to main camera ("webCamScript") *Edit 1*
    a. Insert the following right below the public class WebCamScript : MonoBehaviour
    ```
    public GameObject webCameraPlane;
    ```
    b. Insert the following in the Start function
    ```
    WebCamTexture webCameraTexture = new WebCamTexture();
    webCameraPlane.GetComponent<MeshRenderer>().material.mainTexture = webCameraTexture;
    webCameraTexture.Play();
    ```
9. Drag the plane from the scene into the inspector where the webCameraPlane variable is visible
10. Go back and edit ("WebCamScript") *Edit 2*
    a. Insert the following into the Start function
    ```
    if (Application.isMobilePlatform) {
      GameObject cameraParent = new GameObject("camParent");
      cameraParent.transform.position = this.transform.position;
      this.transform.parent = cameraParent.transform;
      cameraParent.transform.Rotate(Vector3.right, 90);
    }

    Input.gyro.enabled = true;
    ```
    b. Insert the following into the Update function
    ```
    Quaternion cameraRotation = new Quaternion(Input.gyro.attitude.x, Input.gyro.attitude.y,
    -Input.gyro.attitude.z, -Input.gyro.attitude.w);
    this.transform.localRotation = cameraRotation;
    ```
11. Insert a cube to define the bounds of visibility. The cube should have one face that's the exact dimensions of the plane, and its z scale should be modified to contain the camera

User Interface Elements

1. Add a canvas
2. Import crosshair and cockpit images and change texture settings to sprite
3. Add two images and a button to the canvas
4. Select the crosshair and cockpit images and the canvas image sources. Adjust their positions and scales in the camera view
5. Delete the text attached to the button
6. For button source image, choose knob to get a circular button. Choose a button colour, and adjust the button position

Creating the laser beam

1. Create a capsule and edit dimensions to look like a laser beam or bullet. I used (0.25, 2, 0.25) for scale and 90 degrees for x rotation
2. Add rigid body component to bullet
3. Deselect use gravity in the rigid body component
4. Create and add a material to bullet
5. Select is trigger in the mesh collider component
6. Drag laser into the prefabs folder
7. Go back and edit ("WebCamScript") *Edit 3*
    a. Insert the following right below the public class WebCamScript : MonoBehaviour

```
public Button fireButton;
public GameObject bullet;
```

    b. Insert the following into the Start function

```
fireButton.onClick.AddListener(OnButtonDown);
```

    c. Create a new function after the Start function

```
void OnButtonDown() {
  Vector3 bulletDirection = Camera.main.transform.position;
  Quaternion bulletRotation = Camera.main.transform.rotation * Quaternion.Euler(90, 0, 0);
  Vector3 bulletFireDirection = Camera.main.transform.forward;
  GameObject newBullet = Instantiate(bullet, bulletDirection, bulletRotation);

  Rigidbody rb = newBullet.GetComponent<Rigidbody>();
  rb.AddForce(bulletFireDirection * 500f);
  Destroy(newBullet, 3);
  GetComponent<AudioSource>().Play();
}
```

8. Drag the UI button and the bullet in the prefabs folder into the inspector where the fireButton and bullet variables are visible.
9. Attach laser audio source to main camera (turn off play on awake)
10. Set mesh collider of plane off

Creating enemies (geese)

1. Import standard assets
2. Drag the goose model into the assets folder
3. Create a goose the scene, move it to the boundary cube

4. Drag the goose texture into the assets folder and drag it onto the goose body
5. Turn metallic of goose body all the way up
6. Turn smoothness of goose body all the way down
7. Make the goose eye black
8. Add collider to goose. I used a capsule collider for the body, box collider for the wings
9. Add rigidbody component to goose and deselect use gravity
10. Attach a script (“EnemyScript”)
    a. Insert the following into the Start function
    ```
    StartCoroutine("Move");
    ```
    b. Insert the following into the Update function
    ```
    transform.Translate(Vector3.forward * 3f * Time.deltaTime);
    ```
    c. Create a coroutine after the Update function called Move
    ```
    IEnumerator Move() {
      while(true) {
        yield return new WaitForSeconds(2f);
        transform.eulerAngles += new Vector3(0, 180f, 0);
      }
    }
    ```
11. Copy goose, so there are 4 geese
12. Position them around the scene (make sure they’re inside the cube)
13. Add “Player” tag to all four geese
14. Drag all four geese into the prefabs folder
15. Drag the explosion mobile from the standard assets folder to the prefabs folder
16. Create a new script attached to bullet (“CollisionScript”)
    a. Insert the following right below the class CollisionScript : MonoBehaviour
    ```
    public GameObject goose;
    public GameObject goose1;
    public GameObject goose2;
    public GameObject goose3;
    public GameObject explosion;
    ```
    b. Insert the new function below the Update function
    ```
    void OnTriggerEnter(Collider other) {
      if (other.gameObject.CompareTag("Player")) {
        GameObject newExplosion = Instantiate(explosion, other.transform.position,
    other.transform.rotation);
        Destroy(other.gameObject);
        Destroy(newExplosion, 2);

        if (GameObject.FindGameObjectsWithTag("Player").Length == 0) {
          GameObject newGoose = Instantiate(goose);
          GameObject newGoose = Instantiate(goose1);
          GameObject newGoose = Instantiate(goose2);
          GameObject newGoose = Instantiate(goose3);
        }

        Destroy(gameObject);
      }
    }
    ```
17. Drag the 4 geese and the explosionmobile from the prefabs folder to the inspector where the geese and explosion are visible.

Building Out

1. Click build settings and choose android/iOS
2. Go to player settings
3. Select default orientation as landscape left
4. Change package name (com.companyname.productname). Replace company name and product name with whatever you want
5. For android devices:
    a. Make sure android studio is downloaded and up to date
    b. Make sure debugging mode for phone is on (different models have different ways of doing this, this is something to research)
    c. Select build system as internal instead of Gradle
    d. Plug device into computer and make sure it is selected on the build window
    e. Build and run
6. For iOS devices:
    a. Provide a camera usage description (this can be anything)
    b. Build and run (XCode should open up)
    c. If you've never used XCode before, make sure your apple ID is a developer account (this is free). In preferences, go to accounts, make a team, and sign in with your apple ID.
    d. Select "automatically manage signing
    e. Select your team under signing
    f. The prompts will help you create an Apple Developer account
    g. Connect your device and make sure it's selected in XCode
    h. Build out