**Experiment 3 Build an Augmented Reality application having Placement Indicator in it to summon 3D object in it using Unity.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.XR.ARFoundation;

using UnityEngine.XR.ARSubsystems;

public class ARPlacement : MonoBehaviour

{

public GameObject arObjectToSpawn;

public GameObject placementIndicator;

private GameObject spawnedObject;

private Pose PlacementPose;

private ARRaycastManager aRRaycastManager;

private bool placementPoseIsValid = false;

void Start()

{

aRRaycastManager = FindObjectOfType<ARRaycastManager>();

}

void Update()

{

if(spawnedObject == null && placementPoseIsValid &&

Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Began)

{

ARPlaceObject();

}

UpdatePlacementPose();

UpdatePlacementIndicator();

}

void UpdatePlacementIndicator()

{

if(spawnedObject == null && placementPoseIsValid)

{

placementIndicator.SetActive(true);

placementIndicator.transform.SetPositionAndRotation(PlacementPose.position

, PlacementPose.rotation);

}

else

{

placementIndicator.SetActive(false);

}

}

void UpdatePlacementPose()

{

var screenCenter = Camera.current.ViewportToScreenPoint(new

Vector3(0.5f, 0.5f));

var hits = new List<ARRaycastHit>();

aRRaycastManager.Raycast(screenCenter, hits,

TrackableType.Planes);

placementPoseIsValid = hits.Count > 0;

if(placementPoseIsValid)

{

PlacementPose = hits[0].pose;

}

}

void ARPlaceObject()

{

spawnedObject = Instantiate(arObjectToSpawn,

PlacementPose.position, PlacementPose.rotation);

}

**Experiment 4 Build an Augmented Reality application using Unity for inserting multiple AR objects.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.XR.ARFoundation;

using UnityEngine.XR.ARSubsystems;

public class ARRaycastPlace : MonoBehaviour

{

public ARRaycastManager raycastManager;

public GameObject objectToPlace;

public Camera arCamera;

private List<ARRaycastHit> hits = new List<ARRaycastHit>();

void Update()

{

Ray ray = arCamera.ScreenPointToRay(Input.mousePosition);

if(Input.GetMouseButton(0)) {

if(raycastManager.Raycast(ray, hits, TrackableType.Planes)) {

Pose hitPose = hits[0].pose;

Instantiate(objectToPlace, hitPose.position,

hitPose.rotation);

}

}

}}

1. Build an Augmented Reality application using Unity for summoning multiple AR objects.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using Vuforia;

public class ARObjectSummoner : MonoBehaviour, ITrackableEventHandler

{

private TrackableBehaviour mTrackableBehaviour;

public GameObject[] arObjects;

void Start()

{

mTrackableBehaviour = GetComponent<TrackableBehaviour>();

if (mTrackableBehaviour)

{

mTrackableBehaviour.RegisterTrackableEventHandler(this);

}

// Hide all AR objects when the scene starts

foreach (GameObject arObject in arObjects)

{

arObject.SetActive(false);

}

}

public void OnTrackableStateChanged(TrackableBehaviour.Status previousStatus, TrackableBehaviour.Status newStatus)

{

if (newStatus == TrackableBehaviour.Status.DETECTED || newStatus == TrackableBehaviour.Status.TRACKED)

{

// Show the AR objects when the target is detected

foreach (GameObject arObject in arObjects)

{

arObject.SetActive(true);

}

}

else

{

// Hide the AR objects when the target is lost

foreach (GameObject arObject in arObjects)

{

arObject.SetActive(false);

}

}

}

}

1. Build an Augmented Reality application using Unity to use arrows as placement indicator to summon multiple AR objects.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using Vuforia;

public class ARObjectSummoner : MonoBehaviour, ITrackableEventHandler

{

private TrackableBehaviour mTrackableBehaviour;

public GameObject[] arObjects;

public GameObject placementIndicator;

private List<GameObject> instantiatedObjects = new List<GameObject>();

private bool canSummonObject = false;

void Start()

{

mTrackableBehaviour = GetComponent<TrackableBehaviour>();

if (mTrackableBehaviour)

{

mTrackableBehaviour.RegisterTrackableEventHandler(this);

}

placementIndicator.SetActive(false);

}

void Update()

{

if (canSummonObject && Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Began)

{

Vector2 touchPosition = Input.GetTouch(0).position;

if (ARCamera.ScreenPointToRay(touchPosition).origin.y >= 0f)

{

// Summon the selected AR object at the placement indicator position

int index = Random.Range(0, arObjects.Length);

GameObject newObject = Instantiate(arObjects[index], placementIndicator.transform.position, placementIndicator.transform.rotation);

instantiatedObjects.Add(newObject);

}

}

}

public void OnTrackableStateChanged(TrackableBehaviour.Status previousStatus, TrackableBehaviour.Status newStatus)

{

if (newStatus == TrackableBehaviour.Status.DETECTED || newStatus == TrackableBehaviour.Status.TRACKED)

{

// Show the placement indicator when the target is detected

placementIndicator.SetActive(true);

canSummonObject = true;

}

else

{

// Hide the placement indicator and all AR objects when the target is lost

placementIndicator.SetActive(false);

foreach (GameObject arObject in instantiatedObjects)

{

Destroy(arObject);

}

instantiatedObjects.Clear();

canSummonObject = false;

}

}

}