**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное

учреждение высшего образования

**Национальный исследовательский**

**Томский политехнический университет**

Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

Отчет по лабораторной работе №7 по дисциплине

**«Технологии виртуальной и дополненной реальности в мобильных приложениях»**

Мини-проект

Выполнил:

Студент группы 8ИМ11 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Прощенко

Проверил:

Ст. преп. ОИТ ИШИТР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Лоскутов

Томск 2023

# Задание

Придумайте и разработайте индивидуальный небольшой проект с использованием технологий дополненной реальности. Согласуйте тему и объём работы с преподавателем. Используйте технологии, изученные вами ранее и возможности среды Unity. Примеры тем (рекомендуется придумать свою собственную).

**Реализация**

Концептуально, программа представляет собой конструктор интерьера с элементами альтернативной реальности. При запуске программы пользователю предлагается с помощью кнопок выбрать одну из доступных функций:

1. Добавить объект
2. Передвинуть объект
3. Повернуть объект
4. Удалить объект.

Пользователю доступен лист с объектами интерьера. Некоторые объекты интерьера позволяют пользователю добавлять на объект другие объекты, например – цветочный горшок и.т.п. Данный функционал представлен в виде объектов пустышек. Пользователь должен выбрать «пустышку» и в контекстном меню заменить объект на выбранный.

# Текст программы

**ChangeMenu.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Events;

using UnityEngine.UI;

public class ChangeMenu : MonoBehaviour

{

public GameObject catalogueControls;

public GameObject moveControls;

public GameObject rotateControls;

public GameObject currentlyDisplayed;

public GameObject deleteControls;

public GameObject addItemControls;

public GameObject ghost;

public GameObject placementIndicator;

public bool isSectionDisplayedToggle = false;

private static bool isAddItemMode = false;

bool stopFlag = true;

public GameObject addItemButton;

private static GameObject currentEmptyZone;

private static List<GameObject> currentEmptyList;

void Start()

{

}

public void SetAddItem()

{

//ToggleMenu(addItemControls);

addItemControls.SetActive(true);

stopFlag = true;

isAddItemMode = false;

addItemButton.SetActive(false);

}

public void SetCatalogue()

{

ToggleMenu(catalogueControls);

}

public void SetMove()

{

ToggleMenu(moveControls);

}

public void SetRotate()

{

ToggleMenu(rotateControls);

}

public void SetDelete()

{

ToggleMenu(deleteControls);

}

public static void AddItemMode(bool b, List<GameObject> list, GameObject currentEmpty = null)

{

currentEmptyZone = null;

//currentEmptyList = null;

isAddItemMode = b;

if (currentEmpty != null )

{

currentEmptyZone = currentEmpty;

}

if(list != null)

{

currentEmptyList = list;

}

}

private void Update()

{

if (stopFlag && isAddItemMode)

{

addItemButton.SetActive(true);

addItemControls.SetActive(false);

stopFlag = false;

}

else if (!stopFlag && !isAddItemMode)

{

addItemButton.SetActive(false);

stopFlag = true;

currentEmptyZone = null;

}

}

private void ToggleMenu(GameObject menu)

{

addItemControls.SetActive(false);

if (!stopFlag && isAddItemMode)

{

stopFlag = true;

isAddItemMode = false;

addItemButton.SetActive(false);

}

if (currentlyDisplayed == catalogueControls || menu == addItemControls)

{

Destroy(ghost);

placementIndicator.SetActive(false);

}

if (currentlyDisplayed != menu)

{

currentlyDisplayed.SetActive(false);

menu.SetActive(true);

currentlyDisplayed = menu;

isSectionDisplayedToggle = true;

}

else

{

currentlyDisplayed.SetActive(!isSectionDisplayedToggle);

isSectionDisplayedToggle = !isSectionDisplayedToggle;

}

}

public GameObject GetCurrentEmptyZone()

{

addItemControls.SetActive(false);

return currentEmptyZone;

}

public List<GameObject> GetCurrentEmptyList()

{

return currentEmptyList;

}

}

**MoveObject.cs**

using System;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.XR.ARFoundation;

using UnityEngine.XR.ARSubsystems;

public class MoveObject : MonoBehaviour

{

public Camera aRCamera;

private GameObject objectToMove;

Vector3 oldPose;

Vector3 newPose;

void Update()

{

Touch touch = Input.GetTouch(0);

if (touch.phase == TouchPhase.Began)

{

Ray ray = Camera.current.ScreenPointToRay(touch.position);

RaycastHit hitObject;

if (Physics.Raycast(ray, out hitObject))

{

objectToMove = hitObject.transform.parent.transform.parent.gameObject;

oldPose = objectToMove.transform.position;

objectToMove.GetComponent<Recolour>().SetSelected();

objectToMove.transform.parent = aRCamera.transform;

}

}

}

public void Deselect()

{

objectToMove.GetComponent<Recolour>().SetOriginalMaterial();

if (objectToMove.tag == "Table" || objectToMove.tag == "rack")

{

Vector3 delta = objectToMove.transform.position - oldPose;

objectToMove.GetComponent<AdditionalObject>().NewPosition(delta);

}

objectToMove.transform.parent = null;

objectToMove = null;

oldPose = Vector3.zero;

newPose = Vector3.zero;

}

}

**Recolour.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEditor;

using UnityEngine;

public class Recolour : MonoBehaviour

{

public Material valid;

public Material invalid;

public Material selected;

private List<Material> originalMaterials = new List<Material>();

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

void Start()

{

if (gameObject.GetComponent<MeshRenderer>() != null) // has no children

{

originalMaterials.Add(gameObject.GetComponent<MeshRenderer>().material);

}

else

{

foreach(Transform child in transform.GetChild(0).transform)

{

Debug.Log(child.gameObject.name);

children.Add(child.gameObject);

originalMaterials.Add(child.GetComponent<MeshRenderer>().material);

}

}

}

public void SetValid()

{

SetMaterial(valid);

}

public void SetInvalid()

{

SetMaterial(invalid);

}

public void SetSelected()

{

SetMaterial(selected);

}

private bool SetMaterial(Material m)

{

if (gameObject.GetComponent<MeshRenderer>() != null) // has no children

{

gameObject.GetComponent<MeshRenderer>().material = m;

}

else

{

foreach(GameObject child in children)

{

child.GetComponent<MeshRenderer>().material = m;

}

}

return true;

}

public bool SetOriginalMaterial()

{

if (gameObject.GetComponent<MeshRenderer>() != null) // has no children

{

gameObject.GetComponent<MeshRenderer>().material = originalMaterials[0];

}

else

{

int i = 0;

foreach(GameObject child in children)

{

child.GetComponent<MeshRenderer>().material = originalMaterials[i];

i++;

}

}

return true;

}

}

**RotateObject.cs**

using System;

using UnityEngine;

using UnityEngine.UI;

public class RotateObject : MonoBehaviour

{

// Assign in the inspector

private GameObject objectToRotate;

public Slider slider;

// Preserve the original and current orientation

private float previousValue;

private float startValue = 0.5f;

void Awake ()

{

// Assign a callback for when this slider changes

slider.onValueChanged.AddListener (OnSliderChanged);

// And current value

previousValue = slider.value;

}

void OnSliderChanged (float value)

{

// How much we've changed

float delta = value - previousValue;

objectToRotate.transform.Rotate (Vector3.up \* delta \* 360);

// Set our previous value for the next change

previousValue = value;

}

void Update()

{

Touch touch = Input.GetTouch(0);

if (touch.phase == TouchPhase.Began)

{

Ray ray = Camera.current.ScreenPointToRay(touch.position);

RaycastHit hitObject;

if (Physics.Raycast(ray, out hitObject))

{

objectToRotate = hitObject.transform.parent.transform.parent.gameObject;

objectToRotate.GetComponent<Recolour>().SetSelected();

}

}

}

public void Deselect()

{

objectToRotate.GetComponent<Recolour>().SetOriginalMaterial();

objectToRotate = null;

slider.value = startValue;

}

}

**RemoveObject.cs**

using UnityEngine;

public class RemoveObject : MonoBehaviour

{

void Update()

{

Touch touch = Input.GetTouch(0);

if (touch.phase == TouchPhase.Began)

{

Ray ray = Camera.current.ScreenPointToRay(touch.position);

RaycastHit hitObject;

if (Physics.Raycast(ray, out hitObject))

{

if(hitObject.transform.parent.transform.parent.gameObject.tag == "Table" || hitObject.transform.parent.transform.parent.gameObject.tag == "rack")

{

AdditionalObject AO = hitObject.transform.parent.transform.parent.gameObject.GetComponent<AdditionalObject>();

if (!AO)

throw new MissingComponentException("!!!\_ "+ AO.GetType().Name + " component not found!");

AO.RemoveAdditional();

}

Destroy(hitObject.collider.transform.parent.transform.parent.gameObject);

}

}

}

}

**AdditionalObject.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.XR.ARFoundation;

public class AdditionalObject : MonoBehaviour

{

public GameObject emptyObject;

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

private GameObject objjj;

private ARRaycastHit lasthit;

public void CreateAdditionalTable(Vector3 pos , Quaternion rot)

{

Vector3 pos2 = new Vector3(pos.x, pos.y + 0.81f, pos.z);

objjj = Instantiate(emptyObject, pos2, rot);

objjj.AddComponent<ARAnchor>();

objects.Add(objjj);

objects[0].GetComponent<CreateAddObject>().InitializedList(objects);

}

public void CreateAdditionalRack(Vector3 pos, Quaternion rot)

{

Vector3 posFirst = new Vector3(pos.x, pos.y + 0.12f, pos.z);

objjj = Instantiate(emptyObject, posFirst, rot);

objjj.AddComponent<ARAnchor>();

objects.Add(objjj);

Vector3 posSecond = new Vector3(pos.x, pos.y + 0.5f, pos.z);

objjj = Instantiate(emptyObject, posSecond, rot);

objects.Add(objjj);

objjj.AddComponent<ARAnchor>();

Vector3 posThird = new Vector3(pos.x, pos.y + 0.9f, pos.z);

objjj = Instantiate(emptyObject, posThird, rot);

objects.Add(objjj);

objjj.AddComponent<ARAnchor>();

Vector3 posFour = new Vector3(pos.x, pos.y + 1.2f, pos.z);

objjj = Instantiate(emptyObject, posFour, rot);

objects.Add(objjj);

objjj.AddComponent<ARAnchor>();

for (int i = 0; i < objects.Count; i++)

{

objects[i].GetComponent<CreateAddObject>().InitializedList(objects);

}

}

public void RemoveAdditional()

{

//Destroy(objjj);

for (int i = 0; i < objects.Count; i++)

{

Destroy(objects[i]);

}

}

public void NewPosition(Vector3 delta)

{

for (int i = 0; i < objects.Count; i++)

{

if (objects[i] == null)

continue;

objects[i].transform.position = objects[i].transform.position + delta;

}

}

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

}

}

**ARTapAddItem.cs**

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEditor;

using UnityEngine;

using UnityEngine.XR.ARFoundation;

public class ARTapAddItem : MonoBehaviour

{

public GameObject plate;

public GameObject pot;

public GameObject floor;

public GameObject FineContol;

private ARRaycastManager aRRaycastManager;

// Start is called before the first frame update

void Start()

{

aRRaycastManager = FindObjectOfType<ARRaycastManager>();

}

// Update is called once per frame

void Update()

{

}

private void AddObjectOnItem(GameObject obj)

{

GameObject emptyZone = GetCurrentEmptyZone();

List<GameObject> list = GetCurrentEmptyList();

GameObject teee = Instantiate(obj, emptyZone.transform.position, Quaternion.identity);

CreateAddObject cao = teee.GetComponent<CreateAddObject>();

if(list != null)

{

cao.InitializedList(list);

int a = list.FindInstanceID(emptyZone);

CreateAddObject temp = emptyZone.GetComponent<CreateAddObject>();

temp.ReplaceObject(a, teee);

}

else

{

CreateAddObject temp = emptyZone.GetComponent<CreateAddObject>();

temp.ReplaceObject(0, teee);

}

}

public void AddFloorOnItem()

{

AddObjectOnItem(floor);

}

public void AddPotOnItem()

{

AddObjectOnItem(pot);

}

public void AddPlateOnItem()

{

AddObjectOnItem(plate);

}

private GameObject GetCurrentEmptyZone()

{

ChangeMenu temp = FineContol.GetComponent<ChangeMenu>();

GameObject currentEmpty = temp.GetCurrentEmptyZone();

if (currentEmpty == null)

{

Debug.Log("!!!\_ NULL EMPTY OBJECT");

}

else

{

Debug.Log(currentEmpty);

}

return currentEmpty;

}

private List<GameObject> GetCurrentEmptyList()

{

ChangeMenu temp = FineContol.GetComponent<ChangeMenu>();

List<GameObject> list = temp.GetCurrentEmptyList();

if (list == null)

{

Debug.Log("!!!\_ NULL EMPTY list");

}

return list;

}

}

**ARTapToPlaceObject.cs**

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UIElements;

using UnityEngine.XR.ARFoundation;

using UnityEngine.XR.ARSubsystems;

public class ARTapToPlaceObject : MonoBehaviour

{

public GameObject placementIndicator;

public GameObject ghost;

public GameObject objectToPlace;

// furniture

public GameObject chair;

public GameObject table;

public GameObject pouf;

public GameObject shelf;

public GameObject sofa;

public GameObject emptyZone;

private Pose PlacementPose; // contains a Vector3 for a position and a quaternion for rotation

private ARRaycastManager aRRaycastManager;

private ARAnchorManager aRanchorManager;

private ARPlaneManager aPlaneManager;

private bool placementPoseIsValid = false;

private ARRaycastHit lastHit;

ARAnchor localAnchor;

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

void Start()

{

aRRaycastManager = FindObjectOfType<ARRaycastManager>();

aRanchorManager = FindObjectOfType<ARAnchorManager>();

aPlaneManager = FindObjectOfType<ARPlaneManager>();

}

void Update()

{

UpdatePlacementPose();

UpdatePlacementIndicator();

}

private void UpdatePlacementIndicator()

{

if (placementPoseIsValid) // works

{

placementIndicator.SetActive(true);

placementIndicator.transform.SetPositionAndRotation(PlacementPose.position, PlacementPose.rotation);

}

else

{

placementIndicator.SetActive(false);

}

}

// does raycast to center of screen, looks for planes, and stores the results in hits.

// then if there are hits, set placementPose to that pose.

private 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 (hits[0].trackable is ARPlane plane)

{

lastHit = hits[0];

}

if (placementPoseIsValid)

{

PlacementPose = hits[0].pose;

}

}

// Next section

public void PlaceObject()

{

GameObject temp = ghost;

if (placementPoseIsValid)

{

ghost.GetComponent<Recolour>().SetOriginalMaterial();

ghost.transform.parent = null;

temp = Instantiate(objectToPlace, PlacementPose.position, PlacementPose.rotation);

//AnchorContent(temp, temp.GetComponent<Pose>());

//temp.AddComponent<ARAnchor>();

temp.GetComponent<Recolour>().SetValid();

temp.AddComponent<ARAnchor>();

//ghost = Instantiate(objectToPlace, PlacementPose.position, PlacementPose.rotation);

//ghost.GetComponent<Recolour>().SetValid();

placementList.Add(temp);

ghost.transform.parent = placementIndicator.transform;

}

else

{

Debug.Log("!!!\_POSE INVALID");

}

if(objectToPlace.tag == "Table")

{

AdditionalObject temp2 = temp.GetComponent<AdditionalObject>();

temp2.CreateAdditionalTable(PlacementPose.position, PlacementPose.rotation);

}

else if(objectToPlace.tag == "rack")

{

AdditionalObject temp2 = temp.GetComponent<AdditionalObject>();

temp2.CreateAdditionalRack(PlacementPose.position, PlacementPose.rotation);

}

}

ARAnchor AnchorContent(GameObject content, Pose pose)

{

if(lastHit != null)

{

if (aRanchorManager && aPlaneManager)

{

if(lastHit.trackable is ARPlane plane)

{

ARAnchor anchor;

var oldPrefab = aRanchorManager.anchorPrefab;

aRanchorManager.anchorPrefab = content;

anchor = aRanchorManager.AttachAnchor(plane, lastHit.pose);

aRanchorManager.anchorPrefab = oldPrefab;

if (anchor != null)

{

return anchor;

}

}

}

}

return null;

}

private void UseObject(GameObject o)

{

objectToPlace = o;

Destroy(ghost);

ghost = Instantiate(o, PlacementPose.position, PlacementPose.rotation);

ghost.GetComponent<Recolour>().SetValid();

ghost.transform.parent = placementIndicator.transform;

}

public void UseChair()

{

UseObject(chair);

}

public void UseTable()

{

UseObject(table);

}

public void UsePouf()

{

UseObject(pouf);

}

public void UseShelf()

{

UseObject(shelf);

}

public void UseSofa()

{

UseObject(sofa);

}

}

# Результаты работы

На рис. 1 представлена программа в момент исполнения.

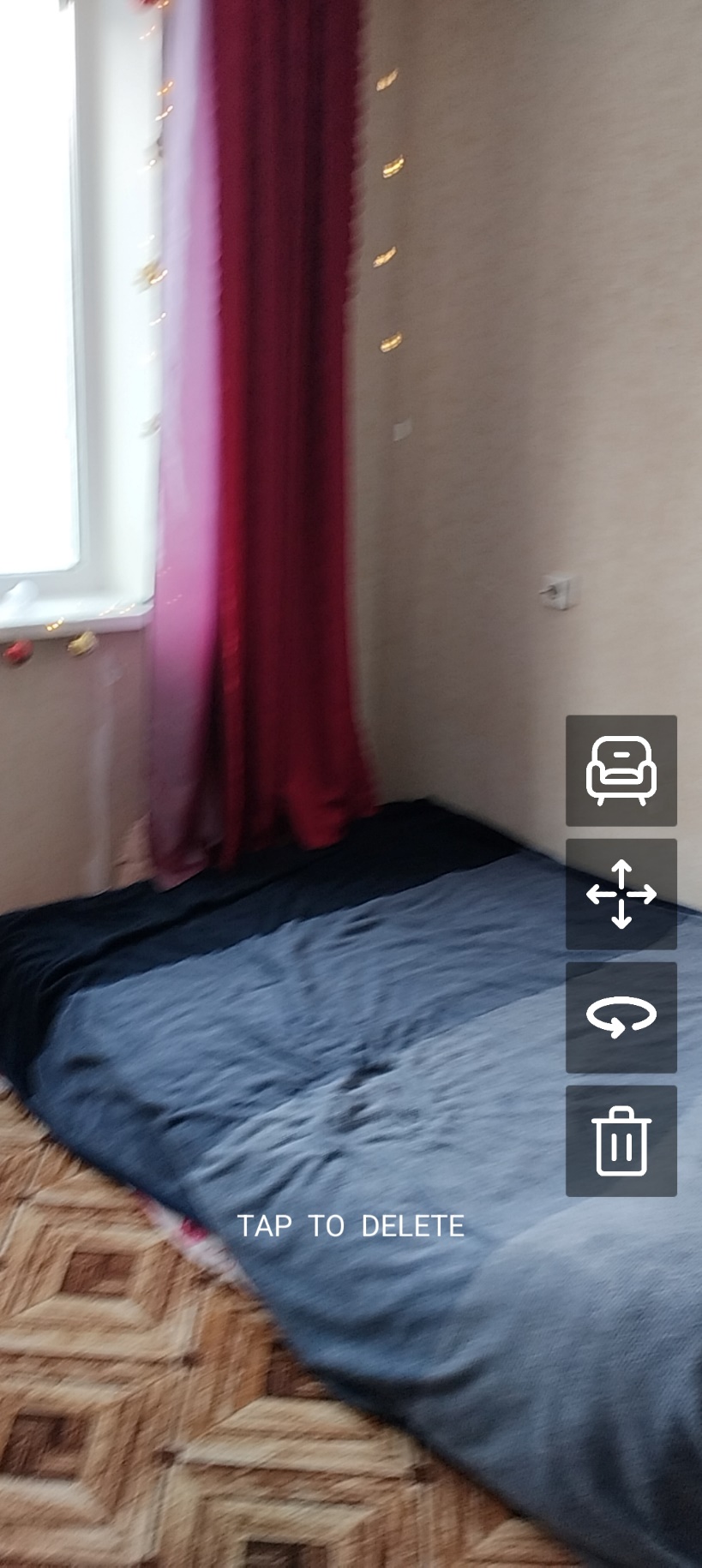


Рис. 1 – Программа в момент исполнения

На рис. 2 Выбор объекта для создания. Размещение на поверхности



Рис. 2 – Программа в момент исполнения. Выбор объекта для создания. Размещение на поверхности

На рис.3 На некоторых объектах можно размещать другие объекты интерьера. Так же представлен интерфейс для работы с интерьером



Рис. 3 – Программа в момент исполнения. Появляется возможность размещать объекты интерьера

На рис.4 Добавили на объект «шкаф» объекты интерьера. Так же, представлен интерфейс для вращения объекта на сцене.



Рис. 4 – Программа в момент исполнения

# Выводы

В результате выполнения лабораторной работы изучены основные аспекты работы AR приложений в Unity. Используя знания полученные в ходе прохождения курса «Дополненная реальность» разработан проект для оформления интерьера в помещениях, например, для новых квартир.

Реализована возможность добавлять и удалять объекты, поворачивать и перемещать их. На объектах, на которых возможно разместить объекты (например, стол или шкаф) возможно размещать прочие объекты (например, объекты интерьера).