Санкт-Петербургский политехнический университет Петра Великого
Институт Прикладной Математики и Механики
Кафедра «Теоретическая механика»

К У Р С О В А Я Р А Б О Т А

Создание приложения для поиска необходимого помещения на базе карты Политехнического университета.

по дисциплине «Языки программирования»

Выполнил
студент гр.23632/2 А.Д. Ершов

Руководитель
Ассистент А.Ю. Панченко

«___» _____ 2018 г.

Санкт-Петербург
2018

СОДЕРЖАНИЕ

Введение.

На данный момент существует бесчисленное множество сервисов, позволяющих работать с онлайн-картами, и интерес к ним, существовавший с 2006 года благодаря компании Google, все больше и больше падает. Онлайн-карты превратились в ежедневную рутину и позволяют построить собственный маршрут со всеми необходимыми деталями любому пользователю, от поездки в магазин - с контролем пробок на дороге – до путешествия на другой конец света.

Однако существуют ситуации, в которых GPS-навигация оказывается бессильной – например, когда расстояние до объектов слишком мало, или состояние объекта тяжело отслеживать. В таком случае пользователь может рассчитывать только на схемы прохода и сведения других людей, но каждодневные поиски при таком раскладе занимают слишком большое время. Для решения именно этой проблемы и возникло приложение PolyGIS.

1. Понятия и термины.

- **Пользователь –** лицо или организация, которое использует действующую систему для выполнения конкретной функции.
- **Прикладная программа**, или **приложение**, — программа, предназначенная для выполнения определённых задач и рассчитанная на непосредственное взаимодействие с пользователем.
- **Маршрут –** путь следования объекта, учитывающий направление движения относительно координат, с указанием начальной, конечной и промежуточных точек, в случае их наличия.
- **Операционная система**, сокр. **ОС** (англ. *operating system, OS*) — комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.
- **Unity** — межплатформенная среда разработки компьютерных игр. Unity позволяет создавать приложения, работающие под более чем 20 различными операционными системами, включающими персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие.
- **C# (C Sharp)** — объектно-ориентированный язык программирования. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java.
- **Авторизация -** предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.

2. Основные принципы создания приложения.

1. Приложение должно решать проблемы Пользователя с определением и поиском необходимой аудитории, сохранять заметки Пользователя о любых помещениях на карте.
2. Интерфейс приложения должен быть интуитивно понятным, доступ к главном функциям приложения должен быть быстрым.
3. Дизайн приложения должен быть лаконичен и не должен содержать яркие/несочетаемые цветовые решения.
4. Приложение должно поддерживаться как можно большим количеством устройств.

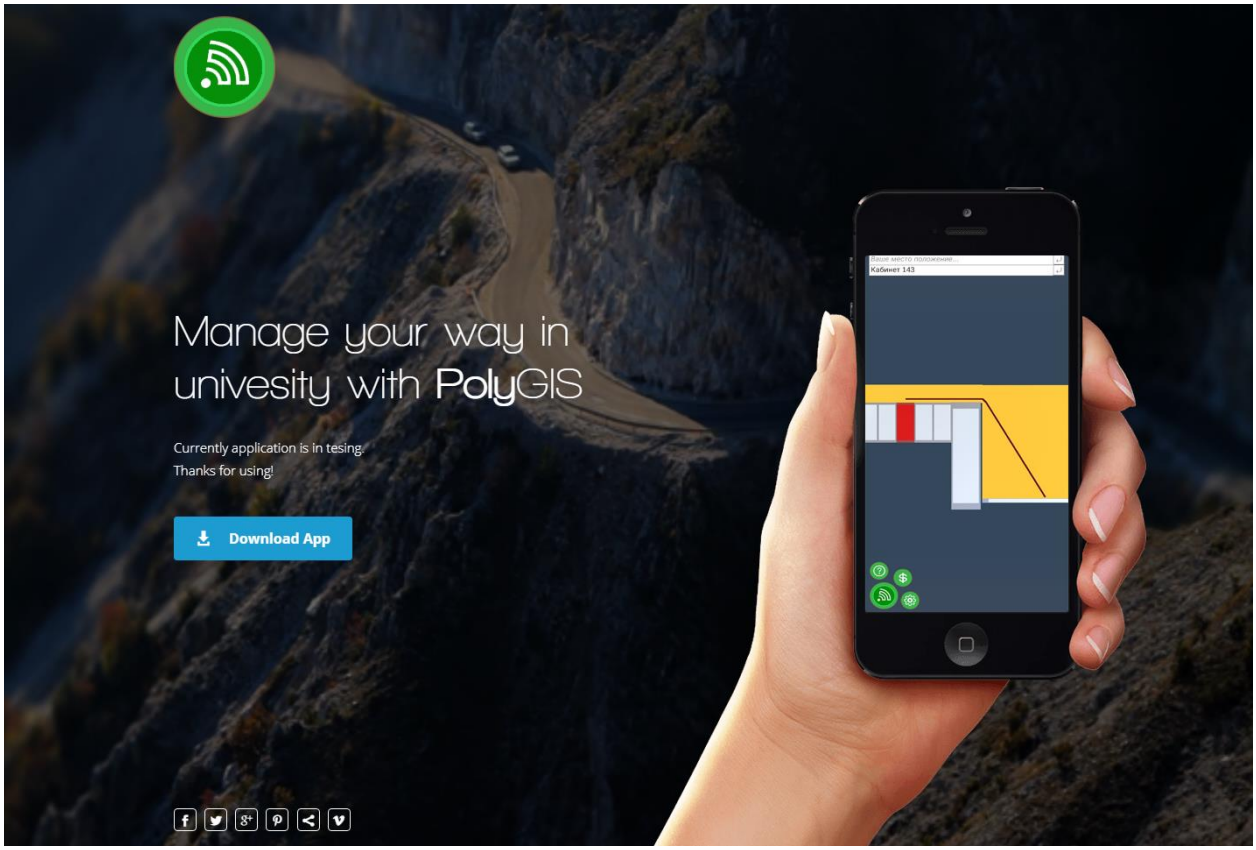
Так как в рамках курсовой работы существовало желание получить новые знания, связанные с языками программирования, было решено выбрать Unity и C# соответственно, хотя для решения поставленной задачи было достаточно уже изученных языков (например, JavaScript)

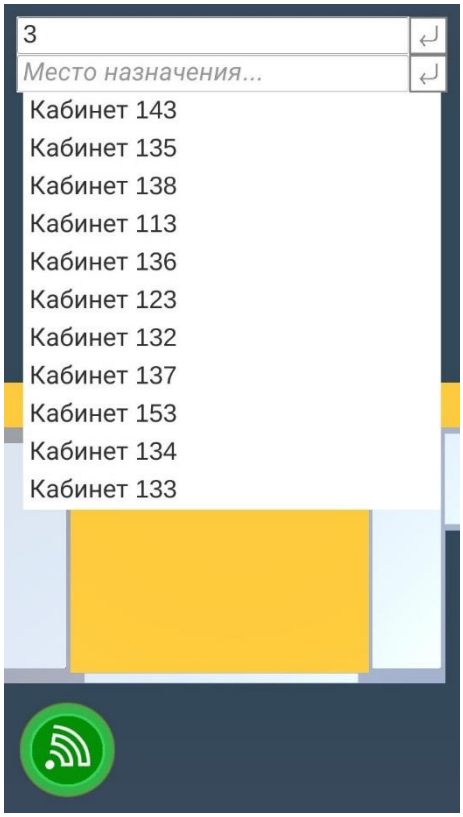
Кроме того, для более широких возможностей бета-тестирования было решено, что приложение будет доступно с сайта приложения, а не в магазине приложений, игр, книг, музыки и фильмов компании Google Google Play.

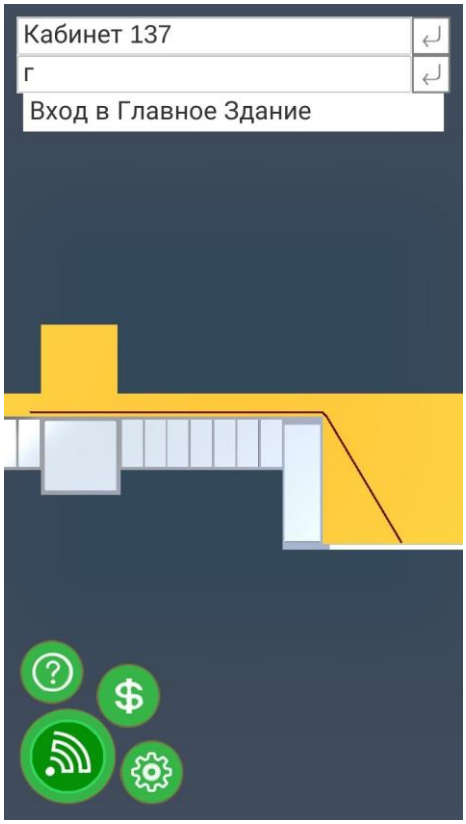
3. Функциональные характеристики.

1. Язык программирования: C#
2. Наличие адаптации к формату экрана: присутствует
3. Портретная ориентация, наличие альбомной ориентации не предусмотрено
4. На данном этапе разработки приложение не требует доступа в интернет
5. Приложение доступно для операционной системы Android на сайте приложения http://polygis.uidev.ru/

4. Демонстрация итогового результата.







5. **Список модулей.**

- AppManager.cs
  Управляет данными о местах, отвечает за взаимодействие других модулей, содержит общие функции.
- Swiper.cs
  Отвечает за перемещение камеры в приложении.
- AutofillSearch.cs
  Модуль автозаполняемого поиска.
- NavManager.cs
  Модуль, отвечающий за навигацию.
- ObjInfo.cs
  Модуль информации о месте.
- Place.cs
  Класс места.
- UIHoverListener.cs
  Проверка на использование интерфейса.
- HelpMenu.cs
  Модуль меню помощи.

Заключение.

Все поставленные требования к функционалу приложения выполнены, приложение работает исправно. Для широкого использования приложения необходимо дополнить карту Главного здания, загрузить режим работы аудиторий и добавить обратную связь в приложении. Проблем, связанных с выбранным языком программирования, не обнаружено, редактор Unity оказался прост и удобен для разработки приложений.

Используемая литература.

1. Okita A., Learning C# Programming with Unity 3D / CRC Press. - 2014
2. Хейлсберг А., Язык программирования C# - 4-е издание / Питер - 2012
3. Система вопросов и ответов о программировании – URL: https://stackoverflow.com/ – (дата обращения: 13.05.2018).

Приложение 1. Пример реализации модуля автозаполняемого поиска.

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

public class AutofillSearch : MonoBehaviour {

        public GameObject itemPrefab;
    public InputField inputField;
    public List<Place> places = new List<Place>();
    public List<Place> possiblePlaces = new List<Place>();
    public List<GameObject> items;
    private GameObject currentItem;
    public string typeSearch;

    private void Awake()
    {
        itemPrefab = transform.FindChild("Item").gameObject;
        inputField = transform.Find("InputField").GetComponent(typeof(InputField)) as
InputField;
    }

    private void Start()
    {
        places = appManager.instance.places;
    }

    public void OnInputChange()
    {
        foreach (GameObject item in items)
        {
            Destroy(item);
        }
        items.Clear();
        string currentText = inputField.text;
        possiblePlaces = GetPossibleMathces(currentText);
        Vector3 pos = itemPrefab.transform.position;
        int i = 1;
        foreach(Place place in possiblePlaces)
        {
            pos.y -= 60;
            GameObject item = Instantiate(itemPrefab, pos, itemPrefab.transform.rotation);
            item.transform.SetParent(transform);
            item.tag = "item";
            item.GetComponent<RectTransform>().offsetMin = new Vector2(10f,
item.GetComponent<RectTransform>().offsetMin.y);
            item.GetComponent<RectTransform>().offsetMax = new Vector2(-10f,
item.GetComponent<RectTransform>().offsetMax.y);
            Transform text = item.transform.FindChild("Text");
```

```
            Text t = text.GetComponent(typeof(Text)) as Text;
            t.text = place.title;
            item.name = i.ToString();
            item.GetComponent<PlaceHolder>().place = place;
            items.Add(item);
            i++;
        }
    }

    public void OnSeachButtonPressed()
    {
        if ((currentItem == null)&&(items.Count != 0))
        {
            SetItem(items[0]);
            foreach (GameObject item in items)
            {
                Destroy(item);
            }
            items.Clear();
        }
    }

    public List<Place> GetPossibleMathces(string typedCharacters)
    {
        List<Place> possibleMathces = new List<Place>();
        int typedLength = typedCharacters.Length;
        foreach (Place currentPlace in places)
        {
            bool flag = true;
            for (int i = 0; i <= (currentPlace.title.Length - typedLength); i++)
            {
                if ((currentPlace.title.Substring(i,
typedLength).ToLower().Equals(typedCharacters.ToLower())) && (flag))
                {
                    possibleMathces.Add(currentPlace);
                    flag = false;
                }

            }
        }
        return possibleMathces;
    }

    public List<Place> SortByDistance(List<Place> placelist)
    {
        List<Place> newplacelist = new List<Place>();
        bool isSorted = false;
        int i_min = 0;
        float d_min = Vector3.Distance(NavManager.instance.curPlace.position,
placelist[0].position);
        while (!isSorted)
        {
```

```
            int i = 0;
            foreach (Place place in placelist)
            {
                isSorted = true;
                i++;
                float d = Vector3.Distance(NavManager.instance.curPlace.position, place.position);
                if (d < d_min)
                {
                    d_min = d;
                    i_min = i;
                    isSorted = false;
                }
            }
            newplacelist.Add(placelist[i_min]);
            placelist.Remove(placelist[i_min]);
            d_min = Vector3.Distance(NavManager.instance.curPlace.position, placelist[0].position);
        }
        return newplacelist;
    }

    public void SetItem()
    {
        if (typeSearch == "destination")
        {
            currentItem = EventSystem.current.currentSelectedGameObject;
            NavManager.instance.SetGoal(currentItem.GetComponent<PlaceHolder>().place);
            transform.FindChild("InputField").GetComponent<InputField>().text =
currentItem.GetComponent<PlaceHolder>().place.title;
            foreach (GameObject item in items)
            {
                Destroy(item);
            }
            items.Clear();
            currentItem = null;
        } else
        {
            currentItem = EventSystem.current.currentSelectedGameObject;
            NavManager.instance.SetPosition(currentItem.GetComponent<PlaceHolder>().place);
            transform.FindChild("InputField").GetComponent<InputField>().text =
currentItem.GetComponent<PlaceHolder>().place.title;
            foreach (GameObject item in items)
            {
                Destroy(item);
            }
            items.Clear();
            currentItem = null;
        }
    }

    public void SetItem(GameObject cur_item)
    {
        if (typeSearch == "destination")
```

```
        {
            currentItem = cur_item;
            NavManager.instance.SetGoal(currentItem.GetComponent<PlaceHolder>().place);
            transform.FindChild("InputField").GetComponent<InputField>().text =
currentItem.GetComponent<PlaceHolder>().place.title;
            currentItem = null;
        }
        else
        {
            currentItem = cur_item;
            NavManager.instance.SetPosition(currentItem.GetComponent<PlaceHolder>().place);
            transform.FindChild("InputField").GetComponent<InputField>().text =
currentItem.GetComponent<PlaceHolder>().place.title;
            currentItem = null;
        }
    }

}
```