

Урок 7: Окончательный вариант плагина

На этом занятии мы добавим в плагин последнюю функцию, сделав его более полезным на практике.

На предыдущем занятии мы использовали Autodesk Revit API для расчета и отображения центра комнаты, которая содержит выбранную группу. Затем мы скопировали группу в центр другой комнаты. На этом занятии мы дополним функциональность плагина, позволив пользователю выбрать несколько комнат для копирования группы.

Обратная связь: напишите нам об этом уроке или обо всем курсе «Моя первая программа»: myfirstplugin@autodesk.com
(Пожалуйста, пишите на английском языке)

Скачать материалы для урока 7

 [lesson7_revit_2013_projects.zip](#) (zip - 21633Kb)

 [lesson7_revit_2012_and_earlier_project_files.zip](#) (zip - 7126Kb)

Планирование новой возможности

Давайте быстро просмотрим те задачи, которые мы решили на предыдущем занятии, для того чтобы понять, куда добавить новую функцию:

- предложить пользователю выбрать группу для копирования;
- рассчитать положение центра выбранной группы;
- найти комнату, которая содержит центр группы;
- рассчитать положение центра комнаты;
- отобразить координаты x , y и z центра комнаты в диалоговом окне;
- рассчитать намеченное положение группы, основываясь на центре комнаты;
- разместить копию группы в целевой точке.

Основная задача данного занятия – позволить пользователю выбрать комнаты, чтобы получить нужную информацию для копирования. Тут есть несколько вариантов; один из способов – четко проконтролировать то, что вводит пользователь в начале, убеждаясь, что вся требуемая информация подготовлена до того, как начнется выполнение основной работы. Другой вариант – это сначала собрать информацию о выбранной группе, а затем выполнить новую задачу по выбору нескольких комнат. В данный момент мы выбираем второй вариант. Так как вам уже не нужна информация о координатах в окне сообщения, мы удалим этот код и добавим новый (выделен **жирным** ниже).

- предложить пользователю выбрать группу для копирования;
- рассчитать положение центра выбранной группы;
- найти комнату, которая содержит центр группы;
- рассчитать положение центра комнаты;
- предложить пользователю выбрать комнаты, в которые будет скопирована группа;**
- разместить копии группы в центре каждой комнаты.**

Итак, когда у нас есть план, давайте посмотрим на тот код, который нам предстоит добавить для повышения функциональности.

Кодирование новых возможностей

Для облегчения работы с исходными кодами, которыми сопровождается каждый урок, мы изменили имена классов так, чтобы они отражали как номер урока, так и его тему.

В частности, в приложении к данному уроку имя класса было изменено с "Lab1PlaceGroup" на "Lab7SelectedRoomsAndPlaceGroups":

```
public class Lab7SelectedRoomsAndPlaceGroups : IExternalCommand
{
```

Обратите внимание, это переименование не является обязательным, и описание к уроку предполагает, что вы всегда работаете с исходным кодом Lab1PlaceGroup. Именно это имя по-прежнему используется в последующем тексте, хотя оно и отличается от имени в доступном к скачиванию приложении к уроку. Вы можете использовать любое имя по своему усмотрению, годится любое.

- Откройте проект C#, который вы создали на занятии 5, в Visual C# Express.
- е. Предложить пользователю выбрать комнаты, в которые будет скопирована группа:**
Начнем с добавления нового фильтра вне тела класса **Lab1PlaceGroup**.

```
/// Фильтр, ограничивающий выбор только комнатами
public class RoomPickFilter : ISelectionFilter
{
    public bool AllowElement(Element e)
    {
        return (e.Category.Id.IntegerValue.Equals(
            (int)BuiltInCategory.OST_Rooms));
    }

    public bool AllowReference(Reference r, XYZ p)
    {
        return false;
    }
}
```

Данный фильтр будет использован в методе **Execute()**.

Код с выводом координат в окно сообщения больше не нужен; можно его удалить или закомментировать, заключив в `/**/`, как в примере ниже:

```
/**
string coords =
    "X = " + sourceCenter.X.ToString() + "\r\n" +
    "Y = " + sourceCenter.Y.ToString() + "\r\n" +
    "Z = " + sourceCenter.Z.ToString();
```

```
TaskDialog.Show("Центр исходной комнаты", coords);
*/
```

Ниже добавим код, который позволит пользователю выбрать комнаты с использованием нового класса **RoomPickFilter**.

```
// Предложение выбрать комнаты для копирования группы
RoomPickFilter roomPickFilter = new RoomPickFilter();
IList<Reference> rooms =
    sel.PickObjects(
        ObjectType.Element,
        roomPickFilter,
        "Выберите комнаты, в которые требуется скопировать группу мебели");
```

3. f. Разместить копии группы в центре каждой выбранной комнаты:

Мы начнем данную задачу с добавления нового метода в класс **Lab1PlaceGroup**. Добавьте код в тело класса **Lab1PlaceGroup**:

```
/// <summary>
/// Группа копируется во все выбранные комнаты. Расположение
/// копий группы отсчитывается от центральных точек комнат;
/// смещение копий от центров комнат – такое же,
/// как у оригинала
/// </summary>
public void PlaceFurnitureInRooms(
    Document doc,
    IList<Reference> rooms,
    XYZ sourceCenter,
    GroupType gt,
    XYZ groupOrigin)
{
    XYZ offset = groupOrigin - sourceCenter;
    XYZ offsetXY = new XYZ(offset.X, offset.Y, 0);

    foreach (Reference r in rooms)
    {
        Room roomTarget = r.Element as Room;
        if (roomTarget != null)
        {
            XYZ roomCenter = GetRoomCenter(roomTarget);
            Group group =
                doc.Create.PlaceGroup(roomCenter + offsetXY, gt);
        }
    }
}
```

Примечание. При работе с API Revit 2013 и API более старших версий, вам будет необходимо заменить следующую строку в коде, который был представлен выше:

```
Room roomTarget = r.Element as Room;
```

На строку представленную ниже

```
Room roomTarget = doc.GetElement(r) as Room;
```

Теперь отредактируем метод **Execute()**, чтобы научить его копировать копии групп в центры выбранных комнат.

Приведенные ниже строки больше не нужны; их можно удалить или закомментировать:

```
XYZ groupLocation = sourceCenter + new XYZ(13.12, 0, 0);
doc.Create.PlaceGroup(groupLocation, group.GroupType);
```

Добавьте следующие строки, вызывающие новый метод:

```
PlaceFurnitureInRooms(
    doc, rooms, sourceCenter,
    group.GroupType, origin);
```

В результате код должен получиться таким:

```
// Размещение мебели в каждой из комнат
Transaction trans = new Transaction(doc);
trans.Start("Lab");
PlaceFurnitureInRooms(
    doc, rooms, sourceCenter,
    group.GroupType, origin);
trans.Commit();
```

Мы ввели весь программный код плагина. Он также доступен для скачивания в верхней части текста занятия. Вы можете сравнить скачанный файл с тем, что у вас получилось.

1. Сохраните файл

2. В меню **Файл (File)** выберите **Сохранить все (Save All)**.

3. Построение проекта:

Примечание: закройте Revit Architecture, если он запущен.

Для компиляции плагина в среде Visual C# Express в меню **Отладка (Debug)** выберите **Построить решение (Build Solution)**. Если построение прошло успешно, то в строке состояния Visual C# Express выводится сообщение «**Построение успешно завершено (Build succeeded)**».

Запуск плагина

1. Запустите Autodesk Revit Architecture.
2. Откройте файл проекта [Hotel.rvt](#).
3. Запустите команду так же, как это делалось на занятии 1. На вкладке ленты Revit **Надстройки (Add-Ins)** нажмите **Внешние инструменты (External Tools)** и из выпадающего списка выберите **Lab1PlaceGroup**.
4. Переместите курсор на **Room1**. Вне зависимости от того, над какой геометрией будет находиться курсор, подсвечена будет только группа с мебелью. Щелкните, чтобы выбрать группу.
5. Выберите несколько комнат, в которые вы хотите скопировать выбранную группу.

Выбранная группа будет скопирована в то же самое место в выбранных комнатах, что и исходная.

Анализ кода

Давайте более внимательно взглянем на только что добавленный код.

Первым делом рассмотрим функцию выбора комнат.

```
IList<Reference> rooms =
    sel.PickObjects(
        ObjectType.Element,
        roomPickFilter,
        "Выберите комнаты, в которые требуется скопировать группу мебели");
```

Вы уже использовали методы **PickObject()** и **PickPoint()** на предыдущих занятиях для выбора одного объекта (например Group) и точки. Основываясь на имени данного метода, давайте посмотрим, какие еще возможности у нас есть для выбора. Если вы откроете файл RevitAPI.chm (из документации Revit API Help) и запустите поиск по слову *Pick*, вы найдете ещё другие методы. Среди них так же есть **PickObjects()**. Как видно из его описания, это как раз то, что мы искали:






Revit 2011 API

Selection.PickObjects Method

[Selection Class](#) [See Also](#) [Send Feedback](#)

Prompts the user to select multiple objects.

Overload List

	Name	Description
	PickObjects(ObjectType)	Prompts the user to select multiple objects.
	PickObjects(ObjectType, ISelectionFilter)	Prompts the user to select multiple objects which pass a customer filter.
	PickObjects(ObjectType, String)	Prompts the user to select multiple objects while showing a custom status prompt string.
	PickObjects(ObjectType, ISelectionFilter, String)	Prompts the user to select multiple objects which pass a custom filter while showing a custom status prompt string.
	PickObjects(ObjectType, ISelectionFilter, String, IList<Reference>)	Prompts the user to select multiple objects which pass a custom filter while showing a custom status prompt string. A preselected set of objects may be supplied and will be selected at the start of the selection.

Как видно, у этого метода есть пять вариантов PickObjects() (вы уже сталкивались с этим при анализе метода PickObject(), известного как *перегрузка*). Так же как и с PickObject(), нас интересует четвертый вариант, который позволяет указать фильтр выбора и строку для пользователя.

Чтобы детально изучить данный вариант, щелкните по ссылке четвертого варианта метода.

Revit 2011 API

Selection.PickObjects Method (ObjectType, ISelectionFilter, String)

[Selection Class](#) [Example](#) [See Also](#) [Send Feedback](#)

Prompts the user to select multiple objects which pass a custom filter while showing a custom status prompt string.

Namespace: [Autodesk.Revit.UI.Selection](#)

Assembly: RevitAPIUI (in RevitAPIUI.dll) Version: 2011.0.0.0

Syntax

C#

```
public IList<Reference> PickObjects(
    ObjectType objectType,
    ISelectionFilter pSelFilter,
    string statusPrompt
)
```

Visual Basic (Declaration)

```
Public Function PickObjects ( _
    objectType As ObjectType, _
    pSelFilter As ISelectionFilter, _
    statusPrompt As String _
) As IList(Of Reference)
```

Visual C++

```
public:
    IList<Reference>^ PickObjects(
        ObjectType objectType,
        ISelectionFilter^ pSelFilter,
        String^ statusPrompt
    )
```

Как видно из описания, этот метод возвращает *IList* объектов *Reference*. *IList* – это общий класс списка, находящийся в пространстве имен **System.Collections.Generic**, который может быть приспособлен под определенный тип объектов – в нашем случае экземпляров Revit API класса *Reference*. Мы объявили переменную типа *IList<Reference>*, которой будет передан результат метода *PickObjects()*.

Прежде чем вызывать данный метод, нам нужно объявить новый класс фильтра выбора по тому же принципу, как это делалось раньше для выбора групп.

Ниже представлен код объявления класса фильтра выбора, ограничивающего выбор только объектами типа *Room*:

```
public class RoomPickFilter : ISelectionFilter
{
    public bool AllowElement(Element e)
    {
        return (e.Category.Id.IntegerValue.Equals(
            (int)BuiltInCategory.OST_Rooms));
    }

    public bool AllowReference(Reference r, XYZ p)
    {
        return false;
    }
}
```

Этот класс очень похож на созданный ранее; изменено имя класса (сейчас он называется **RoomPickFilter**) и ID категории (**OST_Rooms**).

Вернемся к методу *Execute()*. Вы создали экземпляр класса *RoomPickFilter* до вызова метода *PickObjects()*.

```
RoomPickFilter roomPickFilter = new RoomPickFilter();
```

В следующем разбитом на несколько строк фрагменте объявляется переменная списка ссылок, которой будет передано значение метода *PickObjects()*:

```
IList<Reference> rooms =
    sel.PickObjects(
        ObjectType.Element,
        roomPickFilter,
        "Выберите комнаты, в которые требуется скопировать группу мебели");
```

Первая строка содержит переменную (*rooms*) типа (*IList<Reference>*). Остальные вызывают метод, передавая соответствующие параметры.

Следующей задачей этого занятия было размещение группы в центре выбранных комнат. Данный метод перебирает все комнаты в коллекции и рассчитывает центральные точки для размещения групп.

Ниже представлен решающий эту задачу метод:

```
public void PlaceFurnitureInRooms(
    Document doc,
    IList<Reference> rooms,
    XYZ sourceCenter,
    GroupType gt,
    XYZ groupOrigin)
{
    XYZ offset = groupOrigin - sourceCenter;
    XYZ offsetXY = new XYZ(offset.X, offset.Y, 0);

    foreach (Reference r in rooms)
    {
        Room roomTarget = r.Element as Room;
        if (roomTarget != null)
        {
            XYZ roomCenter = GetRoomCenter(roomTarget);
            Group group =
                doc.Create.PlaceGroup(roomCenter + offsetXY, gt);
        }
    }
}
```

В строках, следующих сразу же после объявления метода, где перечисляются передаваемые параметры, объявляются две переменные **offset** и **offsetXY**. Переменная *offset* содержит данные о смещении центра выбранной группы от центра комнаты, которая его содержит. Это смещение можно будет использовать для размещения копий групп в выбранных комнатах. Так как требуется, чтобы группы размещались на том же уровне, что и комнаты, мы заботимся только о смещении X и Y. Это смещение применяется к новым группам.

Следующим шагом был цикл **foreach**. В нем последовательно берется каждый элемент из набора комнат, и над ним выполняются действия.

В теле цикла сначала выполняется приведение каждой ссылки из списка, чтобы можно было работать с ней как с комнатой. Вы делали это, используя ключевое слово «*as*», которое, как вы знаете, возвращает *null* (пустое значение), если объект не является комнатой. Именно поэтому мы использовали оператор «*if*» для того, чтобы убедиться, что работаем с комнатой (то есть условие не равно *null*; «*!=*» означает «не равно»).

Убедившись, что работа ведется с комнатой, мы использовали метод **GetRoomCenter()** из предыдущего занятия для того, чтобы получить центр комнаты, сохранили его в переменной **roomCenter** и разместили группу на подходящем расстоянии от центра целевой комнаты (*roomCenter* + *offsetXY*).

Используя новый метод **PlaceFurnitureInRoom()**, осталось просто обновить метод **Execute()**, чтобы он выполнял нужные действия.

Несмотря на предыдущее обращение к *PlaceGroup()*, мы изменили код, чтобы вызвать *PlaceFurnitureInRoom()* с необходимыми параметрами.

```
// Вставка мебели в каждую из комнат
Transaction trans = new Transaction(doc);
trans.Start("Lab");
PlaceFurnitureInRooms(
    doc, rooms, sourceCenter,
    group.GroupType, origin);
trans.Commit();
```

На этом занятии и вся практическая часть учебного пособия завершаются.

Вначале вы были просто пользователем продукта Revit, затем выучили основы программирования, познакомились с API Revit и создали первый работоспособный плагин для Revit.

Все, что осталось, – это взглянуть на некоторые ресурсы, которые будут вам полезны при дальнейшем применении API Autodesk Revit. Желаем вам всего наилучшего!