Урок 5: Простой выбор группы

Последующие занятия будут посвящены расширению созданного нами ранее кода путем добавления дополнительных функций. На данном занятии мы доработаем функцию выбора, сделав ее проще для пользователя при выборе группы. Мы также исключим возможные неожиданные ситуации, приводящие к ошибкам.

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

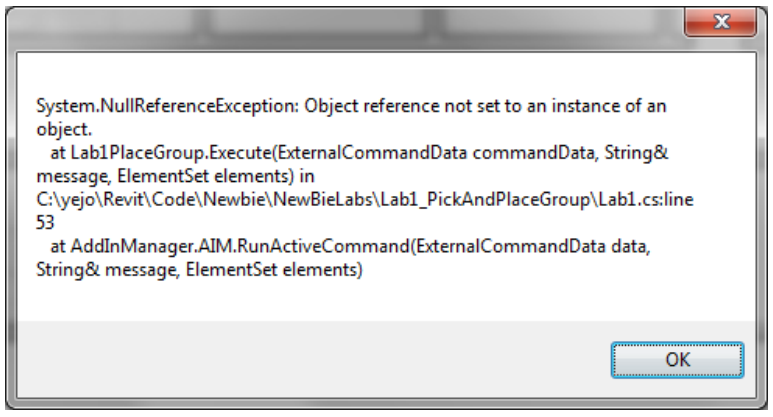
Скачать материалы для урока 5

- lesson5_revit_2013_projects.zip (zip - 32720Kb)
- lesson5_revit_2012_and_earlier_project_files.zip (zip - 7143Kb)

Планирование новых возможностей

При запуске внешней команды, созданной на уроке 1, в строке состояния отображается строка: Выберите группу. В данном представлении есть несколько случаев, которые могут привести к ошибке в плагине. Во-первых, пользователь может выбрать объект, отличный от группы. Во-вторых, пользователь может нажать не ту кнопку во время процесса выбора. Ни одна из этих проблем не была учтена в первоначальном коде. Нам нужно добавить в плагин код для проверки этих ошибок. Прежде всего, посмотрим, что происходит в описанных выше ситуациях:

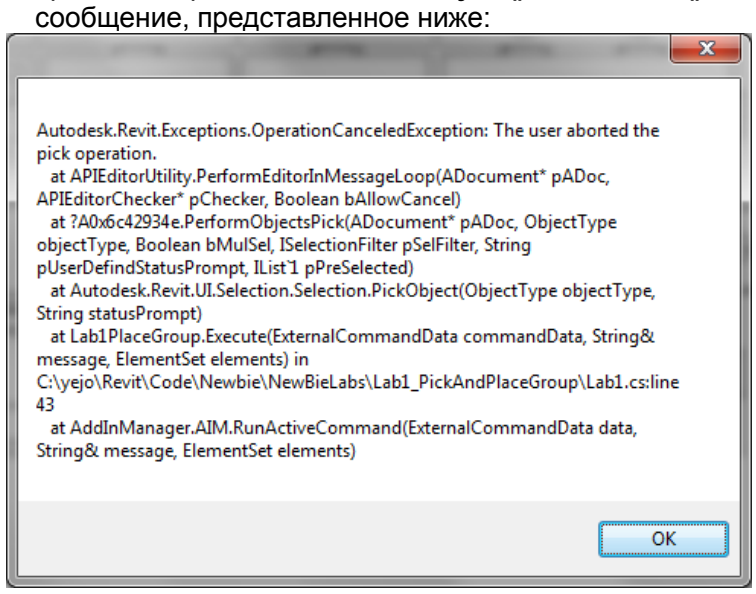
1. **Выбор элемента, не являющегося группой:**
   Когда пользователь выбирает группу, курсор находится над комнатой 1(Room 1). В данном случае комната ограничена стенами и самой группой, которая потенциально может быть подсвечена и выбрана. Но ваша команда ожидает, что пользователь выберет группу. Если будет выбрана стена или комната, то появится диалоговое окно, показанное ниже, прежде чем команда будет отменена.



   Чтобы избежать данной ошибки, пользователь вынужден очень аккуратно наводить курсор для выбора группы. Команда должна реагировать на ситуацию более корректно: ведь появившееся сообщение об ошибке кода никак не помогает пользователю, работающему с плагином.

   Решение для плагина
   Для того чтобы исключить неправильный выбор пользователя, требуется добавить фильтр выбора. Этот фильтр ограничивает типы объектов, которые может выбирать пользователь. Если пользователю предлагается выбрать группу, в пользовательском интерфейсе Revit будут подсвечиваться только группы. Это также позволит пользователю выбрать именно группу, даже если он не обратил внимания на запрос в строке состояния.

2. **Неожиданные щелчки мыши и нажатия клавиш:**
   Во время выбора группы или точки пользователь может случайно щелкнуть не той кнопкой мыши или нажать какую-либо клавишу на клавиатуре. Данная проблема может возникнуть во время выполнения всех методов, ожидающих щелчка левой кнопкой мыши во время выбора, таких как **PickObject()** и **PickPoint()**. Если пользователь нажмет клавишу Esc для отмены действия то команда выдает сообщение, представленное ниже:



   Решение для плагина
   Чтобы предотвратить появление данного диалогового окна, мы добавим код, который будет перехватывать исключение и обрабатывать его. Мы просто отменим нашу команду, если пользователь нажмет Esc или правую кнопку мыши. Для остальных исключений информация пользователю будет предоставляться через стандартное диалоговое окно Revit.

Описанного выше должно быть достаточно, чтобы исключить упомянутые ошибки. Мы также добавим механизм обработки исключений для предотвращения других ошибок, которые могут возникнуть.

Реализация 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 "Lab5PickFilter":

public class Lab5PickFilter : 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.

Мы начнем с добавления фильтра выбора во время вызова метода **PickObject()**. Затем добавим обработчик исключения, который избавит вас от окна ошибки в коде в случае, если пользователь выберет объект, отличный от группы.

1. Заново откройте проект из занятия 1 в Visual C# Express.

2. **Объявление фильтра выбора для групп:**
   Напечатайте следующий код после кода объявления класса **Lab1PickFilter** (после закрытия фигурной скобки Lab1PickFilter).

```
/// <summary>

/// Фильтр, ограничивающий выбор группами модели. Только они

/// выделяются и могут быть выбраны при наведении курсора.

/// </summary>

public class GroupPickFilter : ISelectionFilter

{

  public bool AllowElement(Element e)

  {

    return (e.Category.Id.IntegerValue.Equals(

      (int)BuiltInCategory.OST_IOSModelGroups));

  }

  public bool AllowReference(Reference r, XYZ p)

  {

    return false;

  }

}
```

   В окне кода замените фрагмент:

```
    pickedRef = sel.PickObject(ObjectType.Element,
      "Выберите группу");
```

   На следующий:

```
GroupPickFilter selFilter = new GroupPickFilter();
    pickedRef = sel.PickObject(ObjectType.Element, selFilter,
      "Выберите группу");
```

   Мы вернемся к разбору данного кода чуть позже.

3. *Создание обработчика ошибки:*
   Введите выделенный **жирным шрифтом** код в метод **Execute()** класса **Lab1PlaceGroup** после строки получения объекта документа и до строки выбора пользователем группы.

```
Document doc = uiApp.ActiveUIDocument.Document;

    try

    {

   //…Сюда переносится большая часть кода метода Execute

  }

    //Обработка исключения при щелчке правой кнопкой или нажатии ESC

    catch (Autodesk.Revit.Exceptions.OperationCanceledException)

    {

      return Result.Cancelled;

    }

    //Обработка других ошибок

    catch (Exception ex)

    {
```

```
      message = ex.Message;

      return Result.Failed;

    }
```

Затем переместите следующий код за два блока фигурных скобок слова **try**.

```
      //Определение объекта-ссылки для занесения результата указания
      Reference pickedRef = null;

      //Указание группы с помощью фильтра
      //Может быть выбрана только группа
      Selection sel = uiApp.ActiveUIDocument.Selection;
      GroupPickFilter selFilter = new GroupPickFilter();
      pickedRef = sel.PickObject(ObjectType.Element, selFilter,
        "Выберите группу");
      Element elem = pickedRef.Element;
      Group group = elem as Group;

      //Указание точки
      XYZ point = sel.PickPoint("Укажите точку для размещения группы");

      //Размещение группы
      Transaction trans = new Transaction(doc);
      trans.Start("Lab");
      doc.Create.PlaceGroup(point, group.GroupType);
      trans.Commit();
```

Все исключения будут перехвачены и обработаны данным новым механизмом.

Мы завершили ввод кода, относящегося к этому занятию. Полностью код можно скачать по ссылке ближе к концу текста. Проверьте набранное вами, чтобы удостовериться, что все было проделано правильно.

4. *Сохранение файла:*
   В меню **Файл (File)** выберите **Сохранить все** (**Save All)**.

5. *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 вы увидите сообщение **Построение успешно завершено (Build succeeded)** .

Запуск плагина

1. Запустите Autodesk Revit Architecture.
2. Откройте файл проекта Hotel.rvt.
3. Вызовите команду так же, как это делалось на занятии 1: на вкладке ленты Revit **Надстройки (Add-Ins)** выберите **Внешние инструменты (External Tools)** и из выпадающего списка выберите **Lab1PlaceGroup**.
4. Переместите курсор на **комнату 1**(**Room1)**. Независимо от того, над какой геометрией будет находиться курсор, будет подсвечена только группа с мебелью. Щелкните, чтобы выбрать группу.
5. Далее вам предлагается выбрать точку щелчком левой кнопки мыши. Щелкните правой кнопкой для тестирования обработчика исключений. Команда должна завершиться без сообщения об ошибке. Таким же образом ваш плагин должен среагировать на нажатие клавиши Esc.

Изучение кода

Выбор объекта

Вы использовали метод **PickObject()** для запуска процесса выбора пользователем группы. Данный метод имеет четыре варианта конструктора, которые показаны ниже.

Name	Description
PickObject(ObjectType)	Prompts the user to select one object.
PickObject(ObjectType, ISelectionFilter)	Prompts the user to select one object which passes a custom filter.
PickObject(ObjectType, String)	Prompts the user to select one object while showing a custom status prompt string.
PickObject(ObjectType, ISelectionFilter, String)	Prompts the user to select one object which passes a custom filter while showing a custom status prompt string.

Каждый вариант имеет специфичную для себя функцию и набор параметров. Аргумент **ObjectType** предназначен для указания типа элемента, который должен быть выбран. Всего их четыре вида.

В объектно-ориентированном программировании метод с одинаковым именем может иметь несколько вариантов решения с различными параметрами. Данный подход называется *перегрузка*. В вашем исходном коде вы вызываете метод PickObject() следующим образом:

```
      pickedRef = sel.PickObject(ObjectType.Element, selFilter,
        "Выберите группу");
```

Но его можно вызвать и другим образом:

```
public Reference PickObject(ObjectType objectType, string statusPrompt);
```

Данная перегрузка принимает два параметра: тип объекта и строку. Как указывает имя данного параметра, его значение будет отображено в строке состояния. Пользователь сможет выбрать только элементы, указанные в **ObjectType**.

Код, который вы добавили на данном занятии, указывает еще и фильтр элементов, который пользователь может выбрать. Это делается с помощью другой перегрузки:

```
public Reference PickObject(ObjectType objectType, ISelectionFilter pSelFilter, string statusPrompt);
```

Данная перегрузка просит вас указать три параметра: тип объекта, фильтр выбора и строковое значение. Второй параметр представляет

собой класс, реализующий интерфейс **ISelectionFilter**. Мы уже затрагивали интерфейсы на занятии 3, когда разговаривали об интерфейсе IExternalCommand. На данном занятии вы создали класс GroupPickFilter, который реализует интерфейс ISelectionFilter; этот интерфейс сообщает Revit, какие элементы и ссылки могут быть выбраны.

Интерфейс ISelectionFilter имеет два метода: **AllowElement()** и **AllowReference()**. Вам нужно реализовать их оба в классе GroupPickFilter. Функции параметров прописаны в AllowElement() и AllowReference() и указаны в Revit API. В коде ниже показаны описания этих двух методов.

Это скелет нашего нового класса GroupPickFilter:

```
public class GroupPickFilter : ISelectionFilter
{
  public bool AllowElement(Element e)
  {
    // сюда вставляется код
  }
  public bool AllowReference(Reference r, XYZ p)
  {
    // сюда вставляется код
  }
}
```

Во время процесса выбора элемента, когда курсор проходит над ним, данный элемент передается в метод AllowElement(). Метод AllowElement() позволяет вам проверить данный элемент и определить, подсвечивать его или нет. Если в качестве значения возвращается true, то элемент подсвечивается. Если возвращается false, то элемент не будет подсвечен и выбран.

В реализации метода AllowElement() мы указываем, какой элемент будет подсвечен. В данном коде мы проверяем элемент на соответствие категории “группа модели”:

```
    return (e.Category.Id.IntegerValue.Equals(
      (int)BuiltInCategory.OST_IOSModelGroups));
```

e.Category.Id.IntegerValue получает числовое значение ID элемента **e**, переданного в качестве параметра в AllowElement().

BuiltInCategory.OST_IOSModelGroups представляет собой номер, определяющий категорию “группа модели”, которая получена из перечислителя BuiltInCategory с фиксированными числовыми значениями.

Коллекция BuiltInCategory collection называется перечислителем и объявляется ключевым словом **enum** в языке C#. Перечислитель позволяет ассоциировать читабельное (строковое) название с целым числом. Перечислители представляют собой набор чисел, ассоциированных со строкой. OST_IOSModelGroups – это одно из значений данного перечислителя BuiltInCategory. Оно соответствует числу -2000095, но фактически вам не нужно знать, с каким числом оно связано, так как число в коде использовать не очень практично.

Так как членами данного перечислителя являются целые числа, то предпочтительно перевести BuildingCategory.OST_IOSModelGroups в число для сравнения значения ID категории. Если идентификационный номер элемента, переданный через AllowElement(), соответствует ID категории группы, то данный элемент может быть выбран пользователем.

Из-за того, что AllowElement() возвращает двоичное значение (которое соответствует “true” или “false”), решение возможно записать всего в одну строку:

```
  public bool AllowElement(Element e)
  {
    return (e.Category.Id.IntegerValue.Equals(
      (int)BuiltInCategory.OST_IOSModelGroups));
  }
```

Следующий метод, который вам нужно реализовать – это AllowReference(). В процессе выбора курсор может проходить над ссылками, которые будут переданы в метод AllowReference(). Для определения, будет ли подсвечиваться данная ссылка или нет, этот метод возвращает двоичное значение true или false. Метод AllowReference() также должен быть реализован для завершения ISelectionFilter, но из-за того, что вам вообще не нужны ссылки, вы просто возвращаете значение false:

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

Обработка ошибок

Если происходит что-то не учтенное программой во время её взаимодействия с пользователем, то это приводит к возникновению неожиданной ошибки. .NET предоставляет механизм обработки ошибок, который «перехватывает» данные ошибки, а также исключения. Мы начали написание нашего кода с размещения ограниченного фигурными скобками блока *try*. Это указывает C#, чтобы CLR (Command Language Runtime) следил за *исключениями* (которые могут представлять из себя ошибки или другие нетипичные события) и сообщал о них в блок **catch**. Представьте, что участок с try – это что-то вроде блока проверки исключений; как только обнаруживается исключение, он останавливает исполнение кода своего блока и «выбрасывает» его за пределы. Блок catch подхватывает такие исключение и обрабатывает их. Вы можете написать несколько блоков catch для создания нескольких вариантов поведения на различные исключения.

На данном занятии мы добавили несколько блоков catch. Первый перехватывает моменты, когда пользователь нажимает клавишу Esc или правую кнопку мыши; в этих случаях методами PickObject() или PickPoint() будет послано исключение **OperationCanceledException**. Используя данный блок, вы просто возвращаете из своей команды в качестве результата **Result.Cancelled**, так как вы предполагаете, что пользователь захочет отменить процесс выбора данными действиями:

```
    //Обработка исключения при щелчке правой кнопкой или нажатии ESC
    catch (Autodesk.Revit.Exceptions.OperationCanceledException)
    {
      return Result.Cancelled;
    }
```

Возможны другие исключения, происходящие во время исполнения кода; к примеру, в момент добавления нового объекта группы через метод **PlaceGroup()** (однако фактическая вероятность, что такое случится, очень мала). Бывают неконтролируемые виды ошибок, как, например, недостаточный объем памяти. Чтобы быть уверенным, что такие ошибки будут перехвачены, мы создаем второй блок с более общим исключением .NET. Параметр данного блока catch **System.Exception** (сокращенный вариант которого – просто Exception, в случае, если вы будете использовать ключевое слово **using** с пространством имен System), является базовым классом для всех исключений .NET. Вы можете создавать любое исключение .NET на осное этого типа. Во время исполнения данного блока должна произойти отмена команды; для этого нужно разместить текст сообщения (который доступен через свойство Message) в параметре сообщения метода Execute() и возвратить в качестве результата **Result.Failed**. Это укажет Revit, что команда выполнилась с ошибкой, и будет выведено стандартное окно ошибки.

```
    //Перехват других ошибок
    catch (Exception ex)
    {
      message = ex.Message;
```

```
      return Result.Failed;
    }
```

Мы только что внесли изменения в плагин, повысив функциональность выбора, а также доработав реакцию на ситуации с нетипичными действиями пользователя. Мы также изучили новые возможности для разработки: например перегрузки, перечислители, ключевые слова using, операторы try, catch для перехвата ошибок.