Урок 6: Работа с геометрией комнаты

На следующих двух занятиях мы улучшим процедуру копирования группы в другое место в комнате, выбранное пользователем. Это занятие посвящено копированию группы в центр комнаты. Затем мы расширим возможности плагина, позволив выбрать сразу несколько комнат, в которые можно скопировать группы.

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

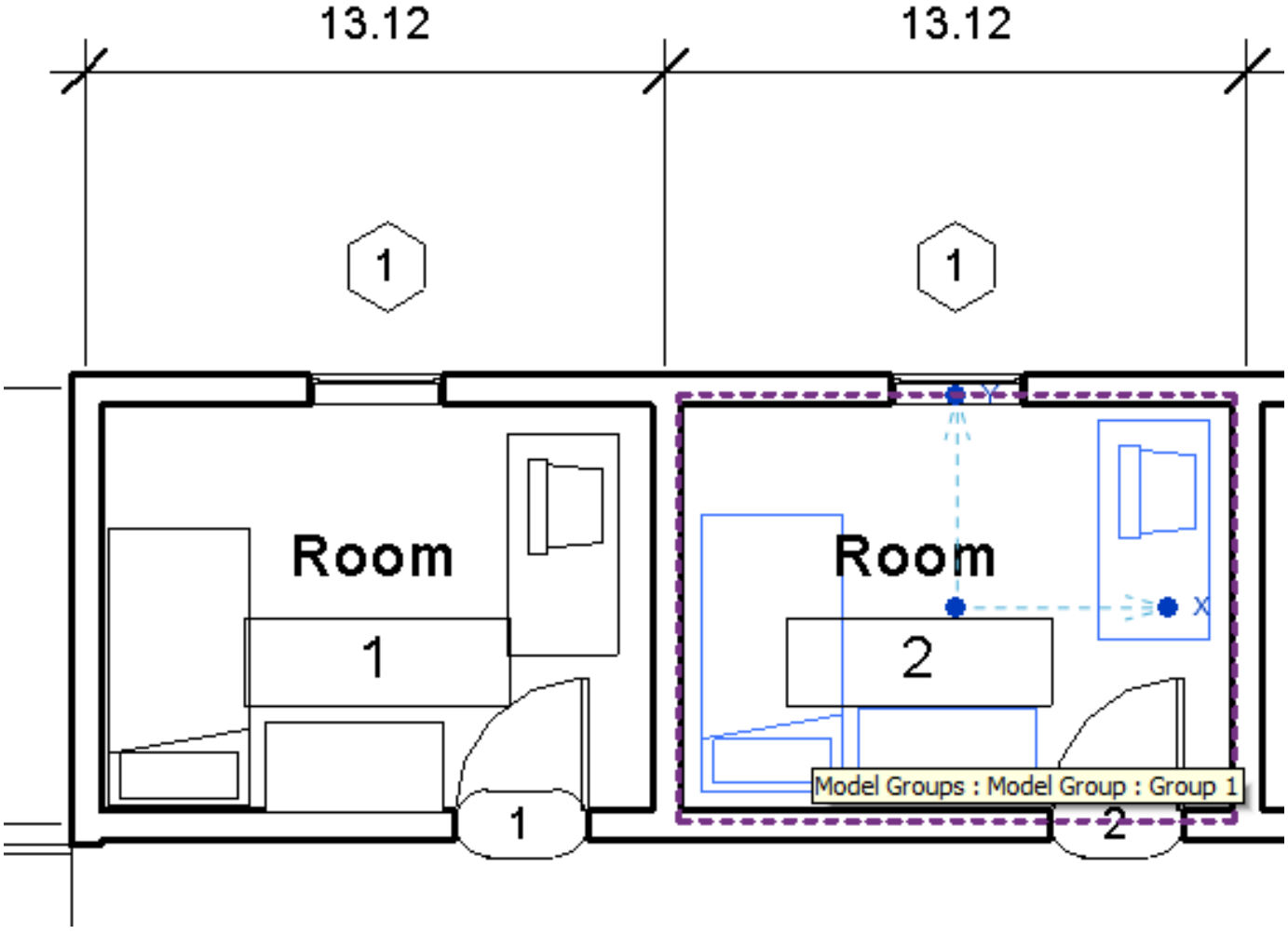
Скачать материалы для урока 6

lesson6_revit_2013_projects.zip (zip - 21635Kb)

lesson6_revit_2012_and_earlier_project_files.zip (zip - 7148Kb)

Планирование новых возможностей

При добавлении новой группы метод **PlaceGroup()** размещает ее центр в указанной точке. На этом занятии мы установим смещение исходной группы относительно комнаты, в которой она размещена. Затем мы сделаем так, чтобы новая группа размещалась в том же месте в указанной комнате.



Один из самых важных моментов – это найти ту комнату, в которой размещена исходная группа. Для упрощения задачи предположим, что выбранная группа находится полностью в одной комнате и не выходит за ее пределы: она не должна пересекать другие комнаты. Затем следует найти, в какой комнате находится центр данной группы. После этого мы определим положение центра данной комнаты и центра копируемой группы и рассчитаем величину смещения. И наконец, скопируем новую группу в другую комнату с учетом смещения.

Задачи, которые мы решили на предыдущих занятиях:

a. предложить пользователю выбрать группу для копирования;

b. предложить пользователю выбрать целевую точку;

c. разместить копию группы в целевой точке.

Чтобы внести дальнейшие улучшения, нам нужно выполнить действия, показанные в списке ниже (выделены **жирным шрифтом**). Поскольку пункт b из старого списка больше не нужен, он будет заменен новой логикой:

a. предложить пользователю выбрать группу для копирования;

b. рассчитать положение центра выбранной группы;

c. найти комнату, которая содержит центр группы;

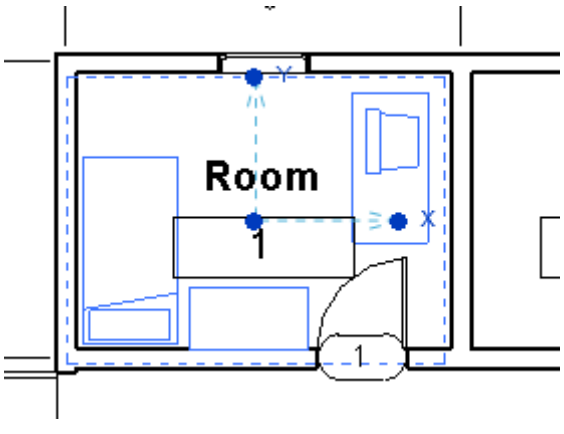
d. рассчитать положение центра комнаты;

e. отобразить координаты x, y и z центра комнаты в диалоговом окне;

f. рассчитать намеченное положение новой группы, основываясь на центре комнаты;

g. разместить копию группы в целевой точке (**требуются некоторые изменения**).

Как видно из списка выше, нам нужно рассчитать положение центра комнаты и группы. Чтобы это сделать, мы определим, где находится центр ограничивающей их области. Для примера на иллюстрации ниже синим цветом выделены ограничивающая рамка и точка в ее центре.



У классов **Group** и **Room** есть свойство **BoundingBox**. Присутствие данного свойства обусловлено тем, что они *наследуют* класс **Element**, у которого есть данное свойство. Классы Group и Room наследуют это свойство от базового класса. Вы можете работать с этим свойством как для объектов Group, так и Room (и любых других объектов, основанных на классе Element).

Свойство объекта Element BoundingBox возвращает объект **BoundingBoxXYZ**, содержащий максимальную и минимальную точки геометрии объекта. Чтобы рассчитать центральную точку, нужно сложить координаты этих двух точек и поделить на два.

Чтобы найти комнату, которая содержит копируемую группу, нужно перебрать все комнаты и проверить каждую на содержание центра нашей группы. Центр комнаты можно рассчитать по тому же принципу, что и центр группы.

Кодирование нового функционала

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 “Lab6FindRoom”:

public class Lab6FindRoom : 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.

1. Откройте проект C#, который был создан на занятии 4, если вы его уже закрыли.
2. **b. Расчет положения центра группы:**

   Напечатайте следующий код внутри класса **Lab1PlaceGroup**, Убедитесь, что он находится за пределами метода **Execute()**. Код создает новый метод **GetElementCenter()**, который получает в качестве параметра **Element** и возвращает его центр.

```
/// <summary>

/// Возвращает центр ограничивающей рамки элемента

/// </summary>

public XYZ GetElementCenter(Element elem)

{

  BoundingBoxXYZ bounding = elem.get_BoundingBox(null);

  XYZ center = (bounding.Max + bounding.Min) * 0.5;

  return center;

}
```

   В методе Execute(), после того как мы получили выбранную группу, напишите строку **выделенную жирным**. Данная строка вызывает метод GetElementCenter() для получения местоположения центра группы.

```
    Group group = elem as Group;

    // Получение центра группы
    XYZ origin = GetElementCenter(group);
```

3. **c. Поиск комнаты, которая содержит центр данной группы**

   Напечатайте код ниже в классе команды **Lab1PlaceGroup**, убедившись, что он находится за пределами других методов. Этот код объявляет новый метод **GetRoomOfGroup()**, который получает в качестве параметра Document и точку, а возвращает комнату, которая содержит данную точку.

```
/// Возвращает комнату, в которой находится точка
Room GetRoomOfGroup(Document doc, XYZ point)
{
  FilteredElementCollector collector =
    new FilteredElementCollector(doc);
  collector.OfCategory(BuiltInCategory.OST_Rooms);
  Room room = null;
  foreach (Element elem in collector)
  {
    room = elem as Room;
    if (room != null)
    {
```

```
        // Точка в указанной комнате?
        if (room.IsPointInRoom(point))
        {
          break;
        }
      }
    }
    return room;
  }
```

Вернувшись в метод Execute() после строки **GetElementCenter()**, которую вы добавили в предыдущем шаге, добавьте строки, **выделенные жирным** ниже. Новый код вызывает метод GetRoomOfGroup(), чтобы найти комнату, которая содержит центр группы.

```
      // Получение центра группы
      XYZ origin = GetElementCenter(group);

      // Получение комнаты, в которой находится указанная группа
      Room room = GetRoomOfGroup(doc, origin);
```

4. **d. Расчет положения центра комнаты. Отображение координат этого центра в диалоговом окне:**

   Добавьте следующий код в класс команды, убедившись, что он не находится внутри существующих методов. Код объявляет новый метод **GetRoomCenter()**, который получает в качестве параметра комнату, а возвращает точку ее центра. Вы будете использовать ранее объявленный метод GetElementCenter() для расчета, но при этом измените координату Z возвращенной точки, чтобы быть уверенным, что она расположена на уровне пола.

```
/// </summary>
  /// Возвращает координаты центра комнаты
  /// Координата Z – уровень пола комнаты
  /// </summary>
  public XYZ GetRoomCenter(Room room)
  {
    // Получение центра комнаты
    XYZ boundCenter = GetElementCenter(room);
    LocationPoint locPt = (LocationPoint)room.Location;
    XYZ roomCenter =
      new XYZ(boundCenter.X, boundCenter.Y, locPt.Point.Z);
    return roomCenter;
  }
```

   В методе Execute() после строк, где определяется комната, которая содержит центр группы, добавьте код, **выделенный жирным**. После того как центр комнаты найден, открывается диалоговое окно с координатами этого центра (мы будем использовать диалоговое окно в стиле пользовательского интерфейса Autodesk Revit).

```
// Получение комнаты, в которой находится указанная группа
      Room room = GetRoomOfGroup(doc, origin);

      // Получение центра комнаты
      XYZ sourceCenter = GetRoomCenter(room);

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

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

   Первый аргумент метода **TaskDialog.Show()** – заголовок диалогового окна.

5. Удалите или закомментируйте следующие строки ниже (закомментировать можно с помощью двух наклонных черт: “//”). Ваша новая группа теперь будет копироваться относительно центра выбранной группы, и от пользователя больше не требуется выбор точки.

```
      //Указание точки

      XYZ point = sel.PickPoint("Please pick a point");
```

6. **f. Расчет целевой точки расположения новой группы, основанной на центре комнаты g. Копирование группы в целевое положение:**
   Удалите или закомментируйте вызов метода **PlaceGroup()** в методе Execute() и замените выделенным **жирным** кодом ниже. Ваша новая группа будет размещена со смещением (13.12, 0, 0) футов от центра комнаты (13.12 – это расстояние между двумя комнатами). Две переменные **sourceCenter** и **new XYZ(13.12,2,0)**, относящиеся к типу XYZ, могут складываться для получения новой точки расположения.

```
doc.Create.PlaceGroup(point, group.GroupType);



      // Расчет положения новой группы

      XYZ groupLocation = sourceCenter + new XYZ(13.12, 0, 0);

      doc.Create.PlaceGroup(groupLocation, group.GroupType);
```

   Мы завершили ввод кода, относящегося к этому занятию. Проверьте набранное вами, чтобы удостовериться, что все было проделано правильно.

7. **Сохраните файл:**
   В меню **Файл (File)** выберите **Сохранить все (Save All)**.

8. **Построение проекта:**
   Примечание: закройте Revit Architecture, если он запущен.

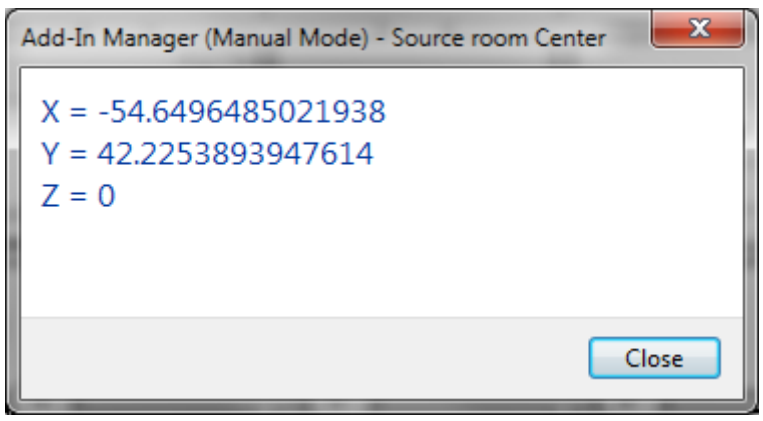
   Для компиляции плагина в среде Visual C# Express в меню **Отладка (Debug)** выберите **Построить решение (Build Solution)**. Если построение прошло успешно, то в строке состояния Visual C# Express выводится сообщение **«Построение успешно завершено» (Build succeeded)**.

Запуск плагина

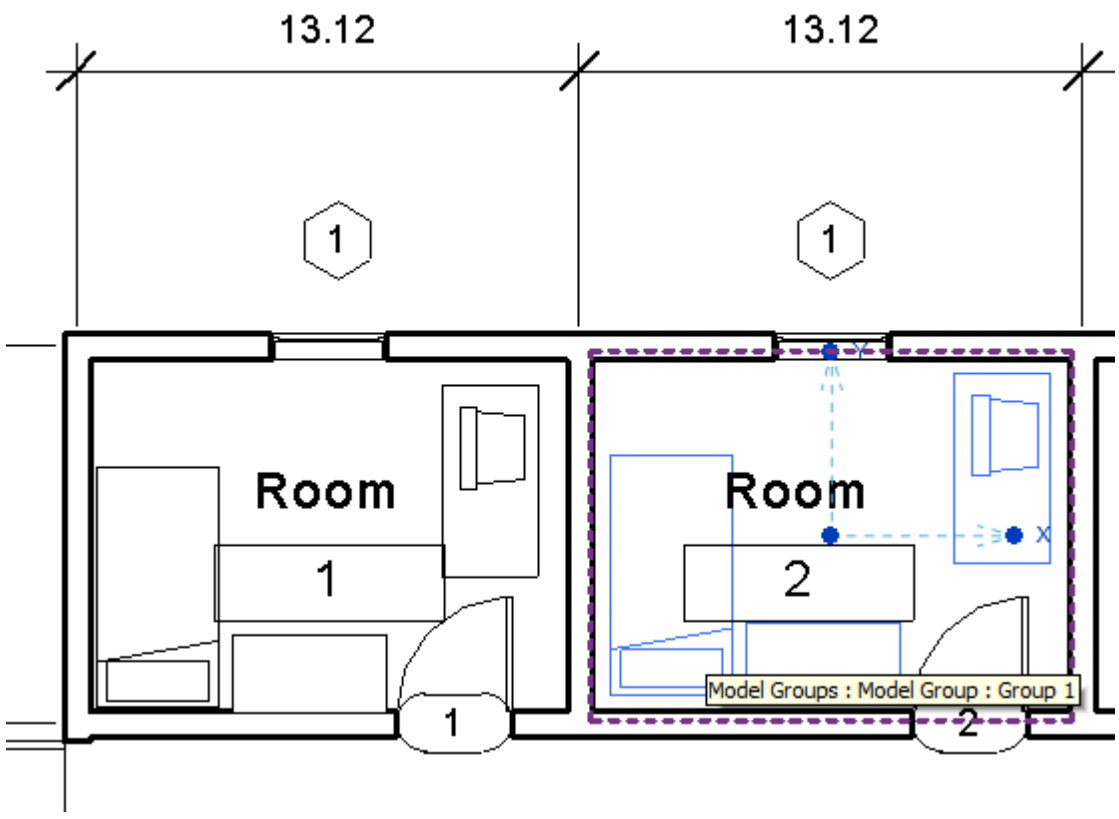
Запуск команды осуществляется так же, как и на предыдущих занятиях.

1. Запустите Autodesk Revit Architecture.
2. Откройте файл проекта Hotel.rvt.
3. Вызовите команду **Lab1PlaceGroup**.
4. Выберите группу в комнате Room 1, как это делалось раньше.

Вы увидите следующее диалоговое окно, в котором будут отображены координаты центра комнаты.



Нажмите кнопку **Закрыть (Close)**. Новая группа должна быть размещена в комнате Room 2. Вектор от центра Room 1 в центр Room 2 – это (13.12,0,0); группа копируется из Room 1 в то же положение в комнате Room 2.



Анализ кода

Давайте взглянем на код более внимательно. Прежде чем приступить к изучению метода Execute(), изучим код новых методов.

Вы объявили метод **GetElementCenter()**:

```
public XYZ GetElementCenter(Element elem)

{

  BoundingBoxXYZ bounding = elem.get_BoundingBox(null);

  XYZ center = (bounding.Max + bounding.Min) * 0.5;

  return center;

}
```

В начале метода GetElementCenter() вы получаете доступ к свойству **BoundingBox** объекта Element и присваиваете его переменной **bounding**.

```
BoundingBoxXYZ bounding = elem.get_BoundingBox(null);
```

Метод BoundingBox немного необычно выводит результат. Если параметр данного метода равен null, он возвращает ограничивающую рамку геометрии модели. Если данный метод принимает больше параметров, перед именем требуется префикс **get**_, чтобы получить значение. Данный префикс не нужен, если метод принимает пустое значение: достаточно просто указать имя параметра.

Переданный объект **BoundingBoxXYZ** содержит координаты минимальной и максимальной точки геометрии. Центр рассчитывается путем получения среднего значения этих точек. Далее это значение присваивается переменной **center**.

```
    XYZ center = (bounding.Max + bounding.Min) * 0.5;
```

Наконец, вы возвращаете данную точку в качестве результата метода.

Вы объявили метод **GetRoomOfGroup ()**:

```
Room GetRoomOfGroup(Document doc, XYZ point)
  {
    FilteredElementCollector collector =
      new FilteredElementCollector(doc);
    collector.OfCategory(BuiltInCategory.OST_Rooms);
    Room room = null;
    foreach (Element elem in collector)
    {
      room = elem as Room;
      if (room != null)
      {
        // Точка в указанной комнате?
        if (room.IsPointInRoom(point))
        {
          break;
        }
      }
    }
    return room;
  }
```

Давайте поближе взглянем на реализацию метода **GetRoomOfGroup()**. В этом методе вы начинаете с получения всех комнат в документе и, перебирая их, находите комнату, которая содержит группу. Класс **FilteredElementCollector** поможет вам в этом: он собирает элементы указанного типа из документа. Именно поэтому нужно было передать параметр документа в метод GetRoomOfGroup().

```
    FilteredElementCollector collector = new FilteredElementCollector(doc);
```

Объект **collector** используется для применения к элементам фильтра, который оставляет в коллекции только комнаты.

```
    collector.OfCategory(BuiltInCategory.OST_Rooms);
```

Вы добавляете фильтр категории с помощью метода **OfCategory()**. Как только данный метод будет применен, в коллекция останутся только комнаты. Класс FilteredElementCollector представляет еще несколько фильтров. Все эти методы могут быть применены несколько раз подряд. За дополнительной информацией по этому поводу, обращайтесь в раздел «Дополнительные темы».

Затем с помощью ключевого слова **foreach** происходит перебор всех комнат. Код в фигурных скобках перебирает по порядку каждый элемент коллекции. Так как вы знаете, что из-за примененного фильтра класса FilteredElementCollector в коллекции содержатся только комнаты, мы получаем доступ к ним.

```
    foreach (Element elem in collector)
    {
      //код в фигурных скобках выполняется циклически
    }
```

Переменная **elem** получает каждый элемент коллекции по порядку. Когда код в теле foreach исполняется в первый раз, данная переменная получает первый элемент коллекции. Когда данный код исполняется второй раз, данная переменная получает вторую комнату коллекции и так далее, пока не будут пройдены все комнаты из документа.

Как упоминалось ранее, данная коллекция возвращает набор элементов базового класса **Element** . Как вы знаете, данные элементы по сути – это экземпляры класса Room, и для корректной работы функции (то есть метода **IsPointInRoom()**) нужно привести данный элемент к этому классу.

```
      room = elem as Room;
```

Ключевое слово **as** перед назначением проверяет тип объекта. Если тип объекта – не Room, то переменной передается значение null. Даже если вы уверены, что в коллекции находятся только объекты класса Room, нужно на всякий случай проверить соответствие типа.

```
      if (room != null)
```

Оператор **if** представляет **условные** операции. Если условие в скобках верно, то блок в фигурных скобках будет выполнен. Также можно использовать ключевое слово **else**, после которого следует блок, который выполнится, если условие неверно. Ключевое слово **if** – очень важный элемент программирования; о нем можно прочитать в разделе «Дополнительные темы».

Как видно из названия метода IsPointInRoom(), он определяет, находится ли точка внутри комнаты. Если точка внутри, то возвращается значение **true**, в противном случае – **false**. Вы вызываете метод IsPointInRoom() для каждой комнаты; как только данный метод вернет значение true, вы понимаете, что данная комната содержит эту группу и остальные комнаты перебирать не нужно. Оператор **break** осуществляет выход из цикла, даже если в коллекции остались непросмотренные комнаты.

```
        if (room.IsPointInRoom(point))
        {
          break;
        }
```

По завершении цикла переменная room содержит комнату, в которой находится точка, определенная методом **IsPointInRoom()**, или последняя комната в коллекции. Переменная возвращается в качестве результата метода **GetRoomOfGroup()**.

```
    return room;
```

Вы определили метод **GetRoomCenter()** так:

```
public XYZ GetRoomCenter(Room room)
{
  // Получение центра комнаты
  XYZ boundCenter = GetElementCenter(room);
  LocationPoint locPt = (LocationPoint)room.Location;
  XYZ roomCenter =
    new XYZ(boundCenter.X, boundCenter.Y, locPt.Point.Z);
  return roomCenter;
}
```

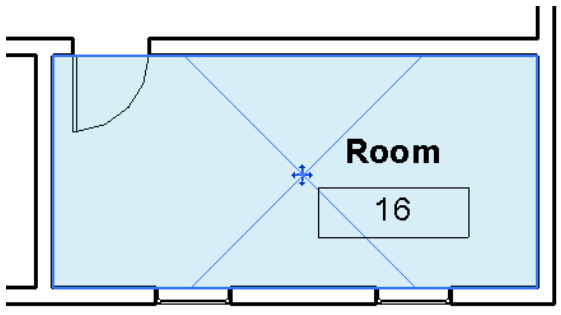
Давайте разберемся с методом GetRoomCenter(). Этот метод использует уже ранее созданный метод GetElementCenter() с изменением возвращаемой точки так, чтобы она находилась на полу.

Первой строчкой вы вызываете метод GetRoomCenter(), возвращающий центральную точку комнаты, и присваиваете результат переменной с именем boundCenter:

```
  // Получение центра комнаты
  XYZ boundCenter = GetElementCenter(room);
```

Чтобы быть уверенным, что данная точка находится на уровне комнаты, мы получим доступ к точке размещения комнаты. Точка размещения комнаты всегда находится на полу, так что вы забираете из этой точки координату Z.

Точка размещения комнаты находится на пересечении линий, отображаемых в Revit при выборе комнаты.



Доступ к этой точке вы получаете через свойство **Location**. Если комната размещена, то свойство **Location** возвращает объект **LocationPoint**. Но свойство Location как параметр класса Element возвращает более обобщенный элемент Location. Для получения доступа к информации о точке размещения нам нужно привести объект к типу LocationPoint. Мы присваиваем данное значение переменной типа LocationPoint.

```
  LocationPoint locPt = (LocationPoint)room.Location;
```

Данное присвоение не использует ключевое слово as, как это было раньше. Ключевое слово as проверяет тип до передачи значения переменной. Код выше не предполагает такой проверки и используется в тех случаях, когда вы уверены, что объект имеет данный тип. Так как вы знаете, что класс Room всегда возвращает LocationPoint из свойства Location, мы можем использовать данный код. Если данное свойство каким либо образом не вернет LocationPoint (гипотетически), операция вызовет исключение **InvalidCastException**.

Для изменения координат точки мы возьмем координаты X и Y из значения центра комнаты, а координату Z – из объекта LocationPoint и используем их для новой точки XYZ.

```
  XYZ roomCenter =
    new XYZ(boundCenter.X, boundCenter.Y, locPt.Point.Z);
```

Наконец, вы возвращаете в качестве значения метода **GetRoomCenter()** данную точку.

С методами мы разобрались. Теперь давайте посмотрим на изменения в методе Execute().

Теперь, когда у вас есть методы по расчету точек, вам нужно просто выполнить их по порядку. Вы видите, как изменяются данные, проходя через новые методы.

```
    // Получение центра группы
    XYZ origin = GetElementCenter(group);

    // Получение комнаты, в которой находится указанная группа
    Room room = GetRoomOfGroup(doc, origin);

    // Получение центра комнаты
    XYZ sourceCenter = GetRoomCenter(room);
```

Центр группы определяется методом **GetElementCenter()**; возвращаемое значение присваивается переменной **origin** типа XYZ. Затем данные передаются в метод GetRoomOfGroup() вместе с активным документом для помощи с перебором комнат в проекте. Результат присваивается переменной room. Затем данные передаются в метод GetRoomCenter() для получения центра комнаты, который присваивается переменной **sourceCenter**.

Переменная sourceCenter имеет тип XYZ и содержит точку центра комнаты, в которой находится выбранная пользователем группа. Теперь нам надо отобразить диалоговое окно с координатами данной точки.

Каждая координата – X, Y и Z – класса XYZ class имеет тип *double*. Этот тип представляет собой *число двойной точности с плавающей запятой*. Тут не о чем волноваться. Переменные двойной точности используют 64 бита для хранения довольно больших чисел (до 15-16 знаков).

Для того чтобы отобразить их в диалоговом окне, нужно преобразовать числа в строки. **sourceCenter.X** возвращает координату X как double, которое может быть переведено в строку методом **ToString()**. Оператор + объединяет две строки в одну. Строка "\r\n" обозначает переход на следующую строку (это комбинация двух символов: возврат каретки (\r) и затем перевод строки (\n)). Это позволяет вам разделить выводимый текст на несколько строк.

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

```
        "Z = " + sourceCenter.Z.ToString();
```

Класс **TaskDialog** позволяет отобразить стандартное диалоговое окно интерфейса Revit. Можно разнообразить данное диалоговое окно информацией, но данный пример использует метод **Show()** для простого отображения строки. Перый параметр – это заголовок окна, а второй – основное сообщение.

```
      TaskDialog.Show("Source room Center", coords);
```

На этом работа с кодом занятия 6 завершена, но до того как вы перейдете к следующему занятию, мы глубже изучим пару тем: фильтрация спомощью FilteredElementCollector и условные операторы.

Дополнительные темы

Фильтрация с помощью FilteredElementCollector

На данном занятии мы использовали модель фильтрации элементов Revit с помощью метода **OfCategory()**:

```
    collector.OfCategory(BuiltInCategory.OST_Rooms);
```

Доступ к элементам и их фильтрация – очень важная часть в разработке приложений для Revit. Класс **FilteredElementCollector** предоставляет несколько методов для добавления фильтров. Метод **OfClass()** добавляет фильтр класса, так что все элементы коллекции будут представлять один класс. Метод **OfCategoryId()** фильтрует элементы коллекции по категории, указанной в параметрах. Для объединения фильтров можно использовать метод **WherePassed()**. Использование данных методов помогает взаимодействовать с большим количеством элементов, что является довольно требовательной к времени и ресурсам процедурой. Вы можете найти больше информации о фильтрации в главе «Chapter 6 Filtering» документа Revit API Developer Guide в Revit SDK.

Оператор if

Возможность выполнять различные части кода в зависимости от различных ситуаций является ключевой особенностью любого языка программирования. В C# это достигается с помощью оператора **if**.

Оператор if позволяет использовать другие операторы (или набор операторов) которые возвращают в качестве результата логическое значение (то есть true или false).

В данном примере двоичной переменной **flagCheck** присваивается true, а затем она проверяется оператором if . На выходе будет строка: «Установлено значение true».

```
    bool flagCheck = true;
    if (flagCheck)
    {
      Console.WriteLine("Установлено значение true.");
    }
    else
    {
      Console.WriteLine("Установлено значение false.");
    }
```

Данное условие, как и ожидалось, возвращает значение true, и выполняется строка **Console.WriteLine("Установлено значение true.");**. Блок else не выполняется в данном примере; после блока if контроль передается следующему за else блоку.

Если вы хотите, чтобы в условном операторе выполнялось более одной строки, то код нужно заключить в фигурные скобки {}.