UDC 004.418

Kudrenko S.A.,
orcid.org/0000-0002-0759-3908
Fomina N.B.,
orcid.org/0000-0003-2357-6371
Kramarenko I.P.

METHOD FOR COMPLEX OBJECTS AUTOMATED DESIGN ON AUTODESK REVIT BASED

National Aviation University

stanislava@i.ua
nfomina@ukr.net

Introduction

Computer-aided design (CAD) systems can be customized using a variety of programming techniques that vary in complexity. Object links programming is the most complex, followed by macro languages, menu customization, and custom graphics. Custom graphic – the easiest type of CAD systems programming. Adding the block to a library of blocks takes that programming up a level. The next step up in custom graphics is to define a custom linetype. Another step up the ladder would be creating a custom hatch pattern or shape set. Most programmers are quite satisfied with learning to create block libraries [1].

Recent advances in technologies such as Internet of Things (IoT), wireless sensors, data processing and analysis, and Building Information Modelling (BIM) have the potential to transform how we interact with the built environment and improve the experience for end users and service providers [1]. The IoT devices and sensors are increasingly being deployed in the built environment and industrial applications.

The number of connected devices have already overtaken connected human beings and are estimated to be around 9 billion. The sensor nodes are being deployed in various application areas such as the industrial, transportation, health and wellbeing, building automation, automotive and retail. The number of sensor installation is increasing at an exponential rate and some estimates suggest that there will be around 50 billion connected devices by 2020.

According to the Building Information Model concept each and every engineering object, particularly Autodesk Revit object, has its own properties and methods. By operating with these properties and methods, it is possible to automate and speed up designing process with the help of programming. In modern modelling systems the ability to work with programming frameworks is established at the core level. Computer engineers only need to choose which software development system will be used to create projects in order to manipulate BIM objects.

Problem statement

Paper purposes are increasing designing efficiency, simplifies and speeds up a work of engineers in modeling of complex objects such as system of buildings engineering equipment structures.

For the implementation of method an integrated development environment Visual Studio and C# programming language was chosen for the creation of the plug-in because of the exhaustive documentation and flexibility of this framework.

Usage of Revit API

API is the acronym for Application Programming Interface: the way a software programmer can communicate with a software product. For instance, the Revit API is the way programmers can work with Revit, and it establishes what functionality a software programmer can use within Revit (fig.1.). Such

as the Revit API allows to write instructions for Revit to execute one after the other.

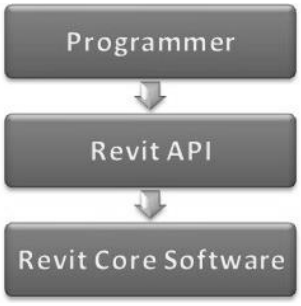

Fig. 1. Revit API interconnection

A software plug-in is a type of program module (or file) that adds functionality to a software product, usually in the form of a command automating a task or some customization of the product's behavior. When we talk about a plug-in for Revit – and the term Add-In used for this product – we mean a module containing code that makes use of the Revit API. Revit loads such plug-ins and uses them to adjust its behavior under certain conditions, such as when a particular command is executed by the user of the plug-in.

Both Revit and BIM are very important for CAD systems future development. Revit helps designers design, simulate, visualize and collaborate in order to capitalize on the advantages of the interconnected data within a BIM model.

Another of the advantages of BIM is the increasing number of simulation tools that allow designers to visualize such things as the sunlight during different seasons or to quantify the calculation of building energy performance. The intelligence of the software, particularly Autodesk Revit, to apply rules that are based on physics and best practices provides a complement for engineers and other project team members. The software can do much more of the analysis and modeling to achieve peak performance, condensing knowledge and rules into a service that can run with the click of a button.

In modern modelling systems the ability to work with programming systems is established at the core level. Developers only need to choose software development system connected with API to create projects.

Creation of plug-ins for engineering equipment modeling

An AddIn manifest is a file located in a specific location checked by Revit when the application starts. The manifest includes information used by Revit to load and run the plug-in.

For plug-ins to load into Revit, they need to be Class Library assemblies (DLLs). It's for this reason, in the second step, that it should be selected the Class Library template. The entered name is used to identify the project within the solution.

The blank project, as created by Visual Studio, did not automatically make use of the Revit API. For it to do so, it should be added project references to the interface DLLs in Revit describing its API, dlland RevitAPIUI.dll [3].

When using the Revit API, it is usual to add project references to the two separate interface DLLs making up the API: one deals with core product functionality, the other with the product's user interface. We must link the project to these files to be able to work with Revit API.

- dll contains the APIs to access the Revit application, documents, elements, parameters, etc.
- dll contains the APIs related to manipulation and customization of the Revit user interface, including command, selections and dialogs.

Having added project references, it's important to set one of their properties appropriately (fig.2).

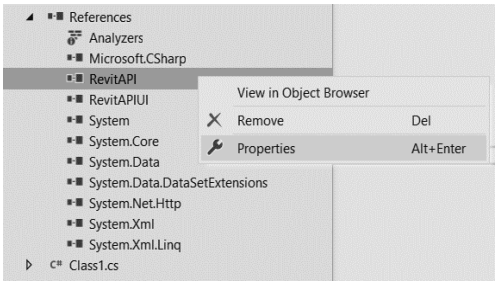

Fig. 2. The properties of RevitAPI

By default, Visual Studio adds project references with its Copy Local property set to True. This means that the referenced DLLs will get copied to the project's output folder when it is built.

Next, we added C# code using the Revit API into the project. In other words, providing Revit with instructions on how to perform the functionality of copying a user-selected group from one place to another.

While developing code, it's a good idea to build the solution from time to time, to check whether errors have been introduced in the code. The code does not necessarily have to be complete or functional when building the solution. This approach can help avoid potentially lengthy troubleshooting once the code is complete, and has the side benefit of automatically saving any edited source files before the build starts.

Revit plug-ins are compiled into library assembly files (DLLs) which are then loaded and executed from within Revit's memory space (fig.3).

During execution of the .NET assembly, CIL (residing in the assembly) is passed through the CLR's just-in-time (JIT) compiler to generate native (or machine) code. JIT compilation of the CIL to native code occurs when the application is executed. As not all of the code is required during execution, the JIT compiler only converts the CIL when it is needed, thus saving time and memory. It also stores any generated code in memory, making it available for subsequent use without the need to recompile.

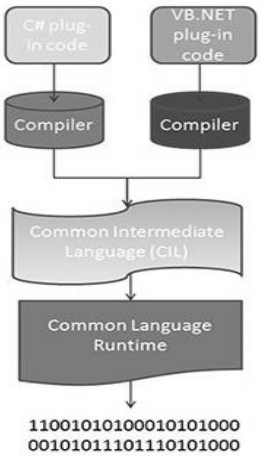

Fig. 3. Running Executables

During execution of the .NET assembly, CIL (residing in the assembly) is passed through the CLR's just-in-time (JIT) compiler to generate native (or machine) code. JIT compilation of the CIL to native code occurs when the application is executed. As not all of the code is required during execution, the JIT compiler only converts the CIL when it is needed, thus saving time and memory. It also stores any generated code in memory, making it available for subsequent use without the need to recompile.

In the last step of this process, the native code gets executed by the computer's processor.

Analysis of the classes in a revit plug-in.

The class in a Revit plug-in that implements this interface is known as the entry point for that plug-in: it's the class that Revit will attempt to find and call the Execute() method upon. Putting it another way, when a Revit user clicks on a command in the Revit user interface listed under the External Tools drop-down button on the Add-Ins tab, the code in the Execute() method is run (executed) from the corresponding class which implements this IExternalCommand interface.

[TransactionAttribute(TransactionMode.Manual)]public class Class1: IExternalCommand{ public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements) { } }

Any block of code in a class which performs a particular task (or action) is called a method. The method declaration starts with the word public in this case.

This method returns a Result (in fact an Autodesk.Revit.UI.Result) rather than being declared void (i.e. not returning anything). The Result returned from the Execute() method will tell Revit whether the command execution has succeeded, failed or been cancelled. If the command does not succeed, any changes it made will be reversed (Revit will cause the transaction that was used to make them to be rolled back).

The Execute() method has three parameters: commandData, message and elements. Let's take a closer look at what each of these parameters refer to:

1. commandData is of type ExternalCommandData and provides with API access to the Revit application. The application object in turn provides with access to the document that is active in the user interface and its corresponding database. All Revit data (including that of the model) is accessed via this commandData parameter.

2. message is a string parameter with the additional ref keyword, which means it can be modified within the method implementation. This parameter can be set in the external command when the command fails or is cancelled. When this message gets set – and the Execute() method returns a failure or cancellation result – an error dialog is displayed by Revit with this message text included.

3. elements is a parameter of type ElementSet which allows to choose elements to be highlighted on screen should the external command fail or be cancelled.

Let's now look at the code inside in the Execute() method. This is the actual set of instructions which uses the Revit API to perform certain tasks when command is executed.

Let's look at the code, line-by-line:

// Get application and document objects UIApplication uiApp = commandData.Application;

In the first line, uses the commandData parameter that was passed into the Execute() method to access the Application property of this object, which provides with access to the Revit application. For more details on understanding properties and reviewing the main Revit API classes and the correlation between them, see the Additional Topics.

To be able to use the Application property just retrieved from the commandData parameter, it was created a container variable for the object named uiApp of type UIApplication. Then we assigned the value of commandData.Application to it for later use in program. Variables can be named as long as the name is unique in that code-block and is not a reserved word (such as the "using" keyword mentioned earlier).

Document doc = uiApp.ActiveUIDocument.Document;

The uiApp variable (which contains the Revit Application object) provides access to the active document in the Revit user interface via the ActiveUIDocument property. In the above line of code – in just one line –directly accessed the database of the active document (this database is represented by the Document class).

Object Selection

Let's look at how to prompted users to select Groups using the API.

//Define a Reference object to accept the pick result. Reference pickedRef = null;

Start by creating an empty variable named pickedRef of type Reference and set its initial value to be null (which literally means nothing). Reference is a class which can contain elements from a Revit model associated with valid geometry.

//Pick a group Selection sel = uiApp.ActiveUIDocument.Selection; pickedRef = sel.PickObject(ObjectType.Element, "Please select a group"); Element elem = doc.GetElement(pickedRef); Group group = elem as Group;

Next, we accessed the current user selection using the API. The user selection from the user interface is represented by the Selection property on the ActiveUIDocument object: placed this Selection object into a variable named sel of type Selection. This Selection object provides with a method named PickObject(). As the method's name suggests, it shifts focus to the user interface and prompts the user to select an object. The parameters of this method allow to specify the type of element the user to select (it can be specified if expecting users to select a face, an element, an edge, etc.) along with the message the user will see in the lower left corner of the Revit user interface while the plug-in waits for the selection to occur.

As the selected Group object has geometry data associated with it, it was safe to place it in the pickedRef variable declared previously. Then it mast used the reference's Element property to gain access to the reference's associated element: in this case it assigned its value to a variable named elem, of type Element. As we are expecting the elem

object to be of type Group, in the last line of the above code snippet we performed a “cast”, allowing us to treat the elem variable as a Group via the variable named group.

Element elem = doc.GetElement(pickedRef);

In the manufacturing world, the term casting refers to the act of setting a given material into a mold to shape it into an object of a particular form. Similarly, in the programming world, casting means the act of trying to set a value of one type into another. Casting asks the language compiler to consider a value in a different way. The as operator in C# will cause the compiler to check the actual type of the object being cast: if it is incompatible with the target type, the value returned by the operator will be null.

The aim of this initial plug-in is to place a selected group at a location selected by the user. To perform this task, it has been used the PlaceGroup() method from the active document’s database object under the creation-related methods made accessible via its Create property. This Create property makes it possible to add new instances of elements – such as Groups – to the Revit model. The PlaceGroup() method, as expected, required to pass in the location at which we wanted to place group, as well as the type (used in the context of Revit, rather than C#) of the group selected by the user.

Finally, it is committed the transaction object using the Commit() method. This ensured the changes encapsulated by the transaction were successfully written to the Revit model.

Coding the New Functionality

For clarity and better organization of the completed source code that we provide as an attachment for each project, we have changed the class names to match the project and the functionality we are working with group.

Type the following code fragment inside the class Class1, making sure it is outside the Execute()method. The code defines a new method, GetElementCenter(), which takes an Element as a parameter and returns its center.

```
Return the center of an element based on its BoundingBox.
public XYZ GetElementCenter(Element elem) {
BoundingBoxXYZ bounding = elem.get_BoundingBox(null);
XYZ center = (bounding.Max + bounding.Min) * 0.5;
return center;
}
```

In the Execute() method, after the line where we get the selected group, type the lines of code highlighted below in bold. The new statement calls new GetElementCenter() method to get the center point of the selected group.

```
Group group = elem as Group;
// Get the group's center point
XYZ origin = GetElementCenter(group);
```

Find the room that contains the center of the group.

Type the following code fragment inside the command class Class1, making sure it is outside any existing method implementations. This code defines a new GetRoomOfGroup() method, which takes a Document and a point as parameters and returns the Room in which the specified point lies.

```
/// Return the room in which the given point is located
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) {
  // Decide if this point is in the picked room
  if (room.IsPointInRoom(point)){
   break;
  }
 }
}
return room;
```

```
}
```

Back in the Execute() method, after the line, GetElementCenter(), which was be added in the last step, type the lines of code highlighted below in bold. The new statement calls new GetRoomOfGroup() method to find the room containing the center of the selected group.

```
// Get the group's center point
XYZ origin = GetElementCenter(group);
// Get the room that the picked group is located in
Room room = GetRoomOfGroup(doc, origin);
```

Calculate the center of the room and e. Display the x, y and z coordinate of the center of the room in a dialog box:

Type the following code fragment inside the command class, once again making sure the code is outside any existing methods. The code defines a new GetRoomCenter() method, which takes a Room and – as the name suggests – returns its center point. We use the previously defined GetElementCenter() to calculate this, but we modify the Z coordinate of the point we return to make sure it's on the floor of the room.

```
/// Return a room's center point coordinates.
/// Z value is equal to the bottom of the room
public XYZ GetRoomCenter(Room room){
// Get the room center point.
XYZ boundCenter = GetElementCenter(room);
LocationPoint locPt = (LocationPoint)room.Location;
XYZ roomCenter =
new XYZ(boundCenter.X, boundCenter.Y, locPt.Point.Z);
return roomCenter;
}
```

In the Execute() method, after the statement which finds the room containing the center point of group, type in the lines of code highlighted in bold, below. The code gets the room's center point and displays it to the user via a task dialog (a type of dialog that uses the Autodesk Revit user interface style).

```
// Get the room that the picked group is located in
Room room = GetRoomOfGroup(doc, origin);
// Get the room's center point
XYZ sourceCenter = GetRoomCenter(room);
string coords =
"X = " + sourceCenter.X.ToString() + "\r\n" +
"Y = " + sourceCenter.Y.ToString() + "\r\n" +
"Z = " + sourceCenter.Z.ToString();
TaskDialog.Show("Source room Center", coords);
```

The first argument of TaskDialog.Show() is the name of which should appear in the title bar at the top of the dialog.

Remove or comment out (using two forward slashes: "//") the following line, which was former step b. New group will be placed relative to the center of the original group's room, so we do not need the user to select anything else, at this stage.

Calculate the target group location based on the room's center and g. Place the copy of the group at the target location:

Remove or comment out the current PlaceGroup() call in the Execute() method and replace it with the following lines in bold. New group will be placed at a displacement of (20, 0, 0) in feet from the center point of the original group's room (20 feet is the width of the two rooms and therefore the horizontal distance between their center points). As both sourceCenter and new XYZ (20,0,0) are of type XYZ, they can be added together to get the new location coordinates.

```
// Calculate the new group's position
XYZ groupLocation = sourceCenter + new XYZ(20, 0, 0);
doc.Create.PlaceGroup(groupLocation, group.GroupType);
```

This completes code for this project. The complete code for this project is also provided for download at the top of this project.

It can be useful to see the complete code to compare results and ensure they are correct.

Save the file:

Build the project:

Inside Visual Studio, in the Debug menu, click Build Solution to compile and build plug-in.

Running the Plug-in

The steps to run the command:

1. Start Autodesk Revit 2019.
2. Open the Project file.
3. Start the command DiplomaPlaceGroup.
4. Select the group in Room 1.

We will see the following task dialog (fig.4) showing the coordinates of the room's center.

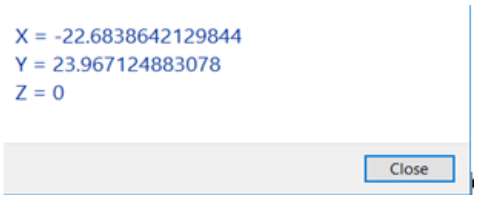

Fig. 4. The task dialog window

Following this, a new group should be inserted into Room 2. Because the displacement of (20,0,0) is the vector from the center of Room 1 to the center of Room 2, the group appears to be copied from Room 1 to the same relative location in Room 2 (fig.5).

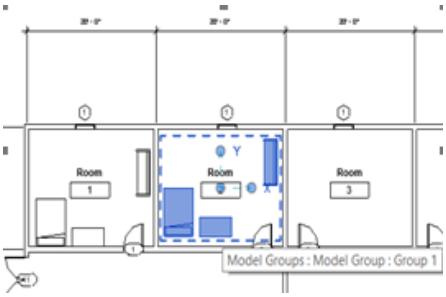

Fig. 5. The window with new functionality

the GetElementCenter() method as follows:

public XYZ GetElementCenter(Element elem) {

BoundingBoxXYZ bounding = elem.get_BoundingBox(null);

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

return center;

In the implementation of the GetElementCenter() method, we started by accessing the BoundingBox property of the Element passed in, storing its value in a variable named bounding.

BoundingBoxXYZ bounding = elem.get_BoundingBox(null);

The BoundingBox property is slightly unusual in that it takes a parameter: the view for which the bounding box is to be calculated. If this parameter is null, the property returns the bounding box of the model geometry. If a property of a class takes one or more parameters, the get_ prefix is needed before the property name to read the property value. This prefix isn't needed if the property doesn't take any parameters: it can be just using the property name.

The returned BoundingBoxXYZ contains the coordinates of the minimum and maximum extents of the Element's geometry. The center point is calculated by taking the average (or mid-point) of these two points. For the sake of clarity, it should be stored this in another variable named center.

Let's now take a closer look at the implementation of the GetRoomOfGroup() method. In this method, start by retrieving all the rooms in the document, going through them to find the room that contains the group. The FilteredElementCollector class helped with this task: it collects elements of a certain type from the document provided. That's why is it necessary to pass a document parameter to the GetRoomOfGroup() method, so it can be used there.

FilteredElementCollector collector = new FilteredElementCollector(doc);

The collector object is now used to filter the elements in the document. In the next step we added a filter requesting that only rooms be collected.

It has been added category filter to the collector using the OfCategory() method. Once the filter was applied, the collector only provided access to rooms. The FilteredElementCollector class provides several methods to add filters (and multiple methods can be applied at the same time for more complex requirements).

Then we iterated through each room in the collector using a foreach expression. The

code between the braces is repeatedly executed on each of the elements found by the collector. These elements will be rooms, at this stage we accessed them as generic elements, as that's how the FilteredElementCollector provides access to them.

foreach (Element elem in collector) {

//code between braces pair executed repetitively.}

The elem variable represents the current element in the collector. So, when the code in the body of the foreach statement gets executed for the first time, the elem variable contains the first room. When the code in the body of the foreach statement is executed again, this time the elem variable contains the second room.

The **as** keyword first checks the actual type of the object before performing the type conversion: if the object is not of type Room, the variable will be set to null. Even though it has been fully expected the collector only to return rooms, it is still good practice to double-check that the room variable contains a valid room, just in case.

if (room != null)

The above **if** statement performs a **conditional** operation. If the condition provided between the brackets evaluates to true, the subsequent code block gets executed. An optional **else** clause can be used to execute different code when the condition evaluates to false (although this particular statement does not have one).

Then uses a break statement to escape the iteration, even though there may well have been rooms that had not yet been checked. The break statement stops execution of code in the enclosing loop (in this case the foreach) and starts executing the code following it.

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

On completion of the loop, the room variable either contains the room in which the point was found – if IsPointInRoom() succeeded for it – or the last room in the list of rooms, otherwise. In either case, the contents of this variable gets returned as the result of the GetRoomOfGroup() method.

return room;

GetRoomCenter() method was defined as follows:

```
public XYZ GetRoomCenter(Room room)
{
  // Get the room center point.
  XYZ boundCenter = GetElementCenter(room);
  LocationPoint locPt = (LocationPoint)room.Location;
  XYZ roomCenter = new XYZ(boundCenter.X, boundCenter.Y, locPt.Point.Z);
  return roomCenter;
 }
```

Conclusions

Modern projects for complex objects the construction of, structures and entire infrastructures take years and thousands of man hours. This work is filled with routine actions that engages almost half of this time.

For example, if we are talking about building design, engineers arrange electrical equipment, ventilation, piping, heating elements and a lot of the rest manually. Family Browser allows to do it in a few seconds and just in two mouse clicks with clarity and filtering of the desired type. This is a dynamic interface to control, store and locate Revit families and types. Family Browser stays up to date with any changes made in windows explorer. Ideal for any small or large practice no matter what flavor of Autodesk's Revit is uses. All families can be controlled from a central location allowing a BIM Manager to instantly make changes, adding groups, tabs or families.

The features of the developed plugin are:

1) during the installation of the .msi package (which is also ready to use), keys are created in the registry dynamically to control versions and directories, i.e. all libraries, .exe files, local databases and. adding files, which Revit needs directly;

2) when we start the audit, the relevance of all files required for the plugin is checked i.e. versions from the server are compared with the versions from the registry of a current

user. If there is any update, the user has an opportunity to download all necessary files (families, templates, ifc export, etc.). Downloading these files is not just the case, but using the self-made FamiliesDownloader.exe file, which connects with the server and then downloads only necessary archives, extracts everything from them and distributes its contents into the required directories;

3) to simplify and perceptibly speed up the design in Revit software using the panel. The panel contains types from families that are grouped into categories – family categories are buttons from the header (fig. 3.14), for example, cables, electrical appliances, safety sensors etc. These types can be double-clicked or dragged into the project without any extra effort of uploading particular family into the project.

4) The insertion occurs by a query in the database, which stores all the info about families and its types (path to the family from which to insert it, path to the picture, description, name, installation type, installation place.

According to the Building Information Model, each and every engineering object, particularly Autodesk Revit object, has its own properties and methods. By opering with these properties and methods, it is possible to automate and speed up designing process with the help of programming.

In modern modelling systems the ability to work with programming frameworks is established at the core level. Computer engineers only need to choose which software development system will be used to create projects in order to manipulate BIM objects.

For the implementation of the practical part of my graduation project an integrated development environment Visual Studio and C# programming language was chosen for the creation of the system because of the exhaustive documentation and flexibility of this framework.

Literature

1. *Eastman C.M.* BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors. / C.M. Eastman. – Hoboken, NJ: John Wiley & Sons, 2011. – 491 p.

2. Azure .NET Developer's Guide Tutorials, Create a C# Template for AutoCAD [Internet Resource] / Web-site: azurewebsites.net; Access mode: http://gilecad.azurewebsites.net/Resources/Template_Csharp_EN.pdf, free.

3. Guide of Revit Image Printer [Internet Resources] / Web-site: Buildin360; Access mode: https://www.building360.ch/ImagePrinter, free.

4. Family Browser Help Page [Internet Resources] / Web-site: Building360; Access mode: https://www.building360.ch/FamilyBrowser/en, free.

Кудренко С.О., Фоміна Н.Б., Крамаренко І.П.

МЕТОД АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ СКЛАДНИХ ОБ'ЄКТІВ НА ОСНОВІ AUTODESK REVIT

Сучасні проекти складних об'єктів, споруд та інфраструктури займають роки та тисячі людських годин. Ця робота наповнена рутинними діями, які займають майже половину цього часу. Дослідження даної статті були спрямовані на підвищення ефективності систем автоматизованого проектування, спрощення роботи інженерів у моделюванні складних об'єктів, таких як система інженерних конструкцій. Для реалізації методу та створення плагіна було обрано інтегроване середовище розробки Visual Studio та мову програмування C # через її вичерпну документацію та гнучкість.

Системи автоматизованого проектування (САПР) можна налаштувати за допомогою різноманітних методів програмування, що відрізняються за складністю. Ehe Revit API - це інструмент, яким програмісти можуть працювати з Revit, і він встановлює, яку функціональність програміст може використовувати в Revit. Revit API дозволяє писати інструкції для Revit для виконання одне за іншим.

Відповідно до концепції Building Information Model, кожен інженерний об'єкт, зокрема об'єкт Autodesk Revit, має свої властивості та методи. Працюючи з цими властивостями та методами, можна автоматизувати та пришвидшити процес проектування за допомогою програмування.

У сучасних системах автоматизованого проектування здатність працювати з рамками програмування встановлена на базовому рівні. Інженерам потрібно лише вибрати, яка система розробки програмного забезпечення буде використовуватися для створення проектів, щоб маніпулювати об'єктами BIM.

Kudrenko S.A., Fomina N.B., Kramarenko I.P.

METHOD FOR COMPLEX OBJECTS AUTOMATED DESIGN ON AUTODESK REVIT BASED

Modern projects for complex objects the construction of, structures and entire infrastructures take years and thousands of man hours. This work is filled with routine actions that engages almost half of this time. Article purposes are increasing designing efficiency, simplifies and speeds up a work of engineers in modeling of complex objects such as system of buildings engineering equipment structures. For the implementation of method an integrated development environment Visual Studio and C# programming language was chosen for the creation of the plug-in because of the exhaustive documentation and flexibility of this framework.

Computer-aided design (CAD) systems can be customized using a variety of programming techniques that vary in complexity. Ehe Revit API is the way programmers can work with Revit, and it establishes what functionality a software programmer can use within Revit. Such as the Revit API allows to write instructions for Revit to execute one after the other.

According to the Building Information Model, each and every engineering object, particulary Autodesk Revit object, has its own properties and methods. By opering with these properties and methods, it is possible to automate and speed up designing process with the help of programming.

In modern modelling systems the ability to work with programming frameworks is established at the core level. Computer engineers only need to choose which software development system will be used to create projects in order to manipulate BIM objects.

Keywords: *Computer-aided design, Building Information Modelling, Application Programming Interface, Executables.*

Кудренко С.А., Фомина Н.Б., Крамаренко И.П.

МЕТОДЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ СЛОЖНЫХ ОБЪЕКТОВ НА ОСНОВЕ AUTODESK REVIT

Современные проекты сложных объектов, сооружений и инфраструктуры занимают годы и тысячи человеческих часов. Эта работа наполнена рутинными действиями, которые занимают почти половину этого времени. Исследование данной статьи были направлены на повышение эффективности систем автоматизированного проектирования, упрощение работы инженеров в моделировании сложных объектов, таких как система инженерных конструкций. Для реализации метода и создание плагина была избрана интегрированная среда разработки Visual Studio и язык программирования C # за его исчерпывающую документацию и гибкость.

Системы автоматизированного проектирования (САПР) можно настроить с помощью различных методов программирования, которые отличаются по сложности. Ehe Revit API – это инструмент, с помощью которого программисты могут работать с Revit, и он устанавливает, какую функциональность программист может использовать в Revit. Revit API позволяет писать инструкции для Revit для выполнения одно за другим.

Согласно концепции Building Information Model, каждый инженерный объект, в том числе объект Autodesk Revit, имеет свои свойства и методы. Работая с этими свойствами и методами, можно автоматизировать и ускорить процесс проектирования с помощью программирования.

В современных системах автоматизированного проектирования способность работать с рамками программирования установлена на базовом уровне. Инженерам нужно только выбрать, какая система разработки программного обеспечения будет использоваться для создания проектов, чтобы манипулировать объектами BIM.