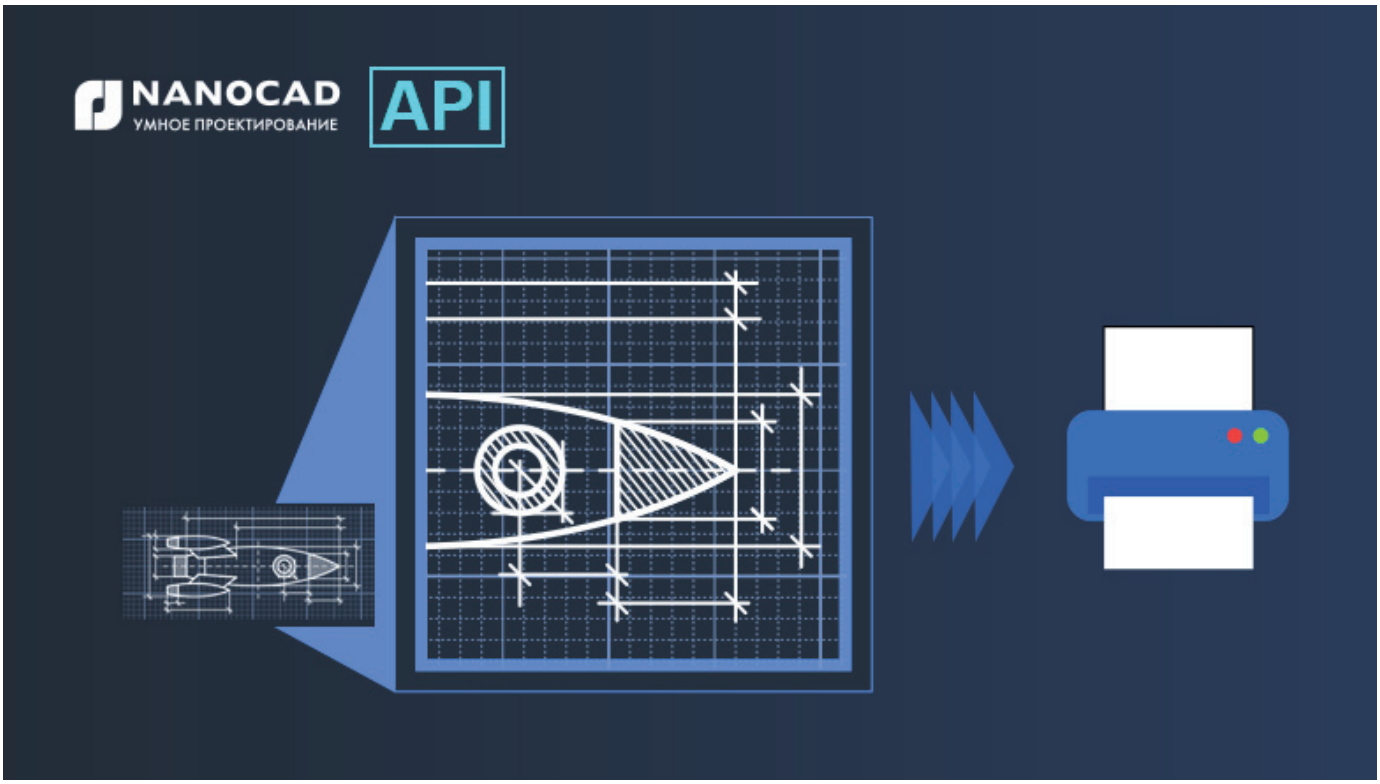

НАСТРОЙКА ОБЛАСТИ ПЕЧАТИ ЧЕРЕЗ API nanoCAD

В nanoCAD при настройке параметров печати необходимо выбрать область печати. Если оставить настройку по умолчанию, может быть напечатано совсем не то, что требовалось.

В этой статье мы разберемся, как работать с областью печати через API nanoCAD.

Чтобы выбрать область печати через интерфейс nanoCAD, следует вызвать окно *Параметры листа* либо *Печать*. В этом окне есть блок *Область печати*, где доступен выпадающий список, из которого можно выбрать нужный тип области печати (рис. 1).

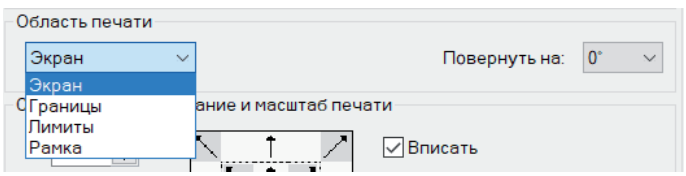

Рис. 1

В случае работы с API nanoCAD за настройку области печати отвечает свойство PlotType документа nanoCAD.

```
nanoCAD.Document comDoc = doc.AcadDocument as nanoCAD.Document;
OdaX.AcadLayout activeLayout = comDoc.ActiveLayout;

activeLayout.PlotType = OdaX.AcPlotType.acExtents;
```

Чтобы разобраться, как работать с областями печати через API nanoCAD, создадим .NET (C#)-приложение, которое устанавливает активному листу выбранную пользователем область печати.

Рассмотрим поэтапно, как создать такое приложение.

Шаг 1. Создаем в Visual Studio проект типа "Библиотека классов (.NET Standard)". Подключаем к нему библиотеки *hostmgd.dll*, *hostdbmgd.dll* и *ncauto.dll*.

Шаг 2. Создаем класс, в котором будет находиться метод, устанавливающий активному листу выбранную пользователем область печати.

```
public partial class Commands
  public partial class Commands
  {
```

```
    [Teigha.Runtime.CommandMethod("SetPlotArea")]
     public void SetPlotAreaToActiveLayout()
    {
    }
}
```

В последующих шагах содержится инструкция по реализации метода SetPlotAreaToActiveLayout().

Шаг 3. Получаем доступ к документу nanoCAD и его настройкам:

```
// Получение ссылки на активный документ
HostMgd.ApplicationServices.Document doc =
   HostMgd.ApplicationServices.Application.DocumentManager.
   MdiActiveDocument;
nanoCAD.Document comDoc = doc.AcadDocument as nanoCAD.
Document;

// Получение ссылки на редактор активного документа
HostMgd.EditorInput.Editor ed = doc.Editor;

// Получение ссылки на активный лист документа
OdaX.AcadLayout activeLayout = comDoc.ActiveLayout;
```

Помимо ссылок на .NET- и COM-документы nanoCAD, нам будут нужны:

- ссылка на редактор документа, которую мы возьмем из .NET-документа (переменная *doc*);
- ссылка на активный лист, которую мы возьмем из COM-документа nanoCAD (переменная *comDoc*).

Через редактор документа будем вести диалог с пользователем: запрашивать у него данные и получать его ответы. Через ссылку на активный лист будем получать его текущие настройки и изменять их.

Шаг 4. Формируем приглашающее сообщение пользователю для выбора области печати.

При формировании запроса лучше добавить к нему ключевые слова, чтобы у пользователя была подсказка, дающая понять, какой формат ответа будет принят и обработан программой. В нашем случае ключевыми словами запроса служат названия областей печати:

```
// Приглашающее сообщение для выбора области печати
HostMgd.EditorInput.PromptStringOptions promptOptions =
   new HostMgd.EditorInput.PromptStringOptions("Выберите об-
   ласть печати");

// Запись в ключевые слова приглашающего сообщения наимено-
ваний доступных типов
// областей печати, которые возможны и в пространстве модели,
и в листах
promptOptions.Keywords.Add("Экран");
promptOptions.Keywords.Add("Границы");
promptOptions.Keywords.Add("Рамка");

// Для области печати "Вид" создаем коллекцию,
// в которую будем записывать названия именованных видов ак-
тивного листа
System.Collections.Generic.List<string> namedViews =
   new System.Collections.Generic.List<string>();

// Тип области печати "Вид" доступен только
// если у активного листа есть сохраненные именованные виды
if (comDoc.Views.Count != 0)
{
   // Если коллекция Views не пустая, ищем в ней именованные виды
   // активного листа
   foreach (OdaX.AcadView namedView in comDoc.Views)
   {
      // Если найден именованный вид для активного листа,
      // добавляем название этого вида в коллекцию namedViews
      if (namedView.LayoutId == activeLayout.ObjectID)
      namedViews.Add(namedView.Name);
   }

   // Если в итоге коллекция namedViews не пустая,
   // то добавляем к запросу ключевое слово "Вид"
   if (namedViews.Count != 0) promptOptions.Keywords.Add("Вид");
}

// Добавление к запросу ключевых слов в зависимости от типа ак-
тивного листа
if (activeLayout.ModelType) promptOptions.Keywords.Add("Лимиты");
else promptOptions.Keywords.Add("Лист");

// Запрос пользователю с предложением выбрать область печати
HostMgd.EditorInput.PromptResult result = ed.
GetString(promptOptions);
```

Сначала определяем основной текст приглашающего сообщения, потом добавляем к сообщению ключевые слова:

- области печати "Экран", "Границы", "Рамка" доступны для любого типа листа и при любых настройках документа nanoCAD. Их сразу добавляем к ключевым словам запроса;
- область печати "Вид" доступна для выбора только если в активном листе документа nanoCAD есть сохраненные именованные виды. Все именованные виды документа хранятся в коллекции Views COM-документа;
- область печати "Лимиты" доступна только для пространства модели, то есть при значении свойства ModelType активного листа, равном true;
- область печати "Лист" доступна только для листов, то есть в случае, когда значение свойства ModelType активного листа равно false.

Сформированное нами сообщение пользователю в новом документе без сохраненных именованных видов в листе будет выглядеть так, как показано на рис. 2.

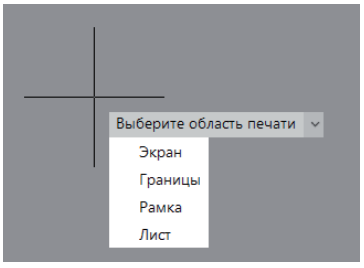

Рис. 2

В пространстве модели вместо ключевого слова "Лист" будет другое — "Лимиты".

Шаг 5. Обработка ответа пользователя.

Теперь пользователь должен выбрать одно из ключевых слов, а программе нужно обработать его ответ.

Для начала следует проверить, совпадает ли введенный пользователем ответ с одним из ключевых слов запроса. Если да, то продолжаем обработку, если нет – завершаем работу программы, оставив настройки печати активного листа без изменений:

```
// Обработка полученного результата запроса
if (result.Status == HostMgd.EditorInput.PromptStatus.Keyword)
{
   switch (result.StringResult)
   {
   }
   // Составление словаря для сопоставления названий
   // типов областей печати в перечислении OdaX.AcPlotType и в интерфейсе
   System.Collections.Generic.Dictionary<string, OdaX.AcPlotType>
dictionary =
   new System.Collections.Generic.Dictionary<string, OdaX.AcPlotType>();
   dictionary.Add("Экран", OdaX.AcPlotType.acDisplay);
   dictionary.Add("Границы", OdaX.AcPlotType.acExtents);
   dictionary.Add("Лист", OdaX.AcPlotType.acLayout);
   dictionary.Add("Лимиты", OdaX.AcPlotType.acLimits);
   dictionary.Add("Вид", OdaX.AcPlotType.acView);
   dictionary.Add("Рамка", OdaX.AcPlotType.acWindow);

   // Поиск в словаре выбранного типа области печати
   // и запись найденного значения в переменную ptResult
   OdaX.AcPlotType ptResult;
   dictionary.TryGetValue(result.StringResult, out ptResult);

   // Назначение выбранной области печати активному листу
   activeLayout.PlotType = ptResult;
   ed.WriteMessage("Листу '{0}' назначена область печати '{1}'",
         activeLayout.Name, result.StringResult);
}
else ed.WriteMessage("Область печати не изменена");
```

Чтобы изменить область печати активного листа, необходимо сопоставить название области печати из запроса пользователю со значением из перечисления OdaX.AcPlotType. Для этого создаем словарь, в каждой записи которого будет пара значений: ключевое слово из запроса и соответствующее ему значение перечисления OdaX.AcPlotType. Далее, используя этот словарь, мы сможем легко находить нужное нам значение OdaX.AcPlotType по ответу пользователя.
При обработке ответа пользователя также стоит учесть, что есть два типа области печати, для которых требуются дополнительные данные, – это acView ("Вид") и acWindow ("Рамка"). Для области печати "Вид" нужно указать название именованного вида, а для области печати "Рамка" – указать две точки области печати. То есть, если пользователь выбрал один из этих двух типов области печати, программа должна сделать больше чем просто изменить значение свойства PlotType.
В фрагменте кода, приведенном в этом шаге, есть пустой оператор switch, который был добавлен именно для этих двух областей печати:

```
switch (result.StringResult)
{
}
```

Такую обработку необходимо выполнить именно до изменения значения свойства PlotType, иначе наша программа не сработает как надо.

Шаг 5.1. Обработка ключевого слова "Вид".

```
case "Вид":
   {
      // Приглашающее сообщение для выбора именованного вида
      HostMgd.EditorInput.PromptStringOptions opts =
      new HostMgd.EditorInput.PromptStringOptions("Выберите именованный вид");

      // Запись названий именованных видов активного листа
      // в ключевые слова запроса
      namedViews.ForEach(nv => opts.Keywords.Add(nv));

      // Запрос пользователю с предложением выбрать именованный вид
      HostMgd.EditorInput.PromptResult res = ed.GetString(opts);

      // Обработка полученного результата запроса
      if (res.Status == HostMgd.EditorInput.PromptStatus.Keyword)
         activeLayout.ViewToPlot = res.StringResult;
      break;
      }
```

Если пользователь выбрал тип области печати "Вид", необходимо сформировать еще один запрос, в результате которого пользователь выберет, какой именно именованный вид будет использован.
Для того чтобы пользователь выбрал именованный вид, добавляем в ключевые слова второго запроса названия именованных видов активного листа, которые хранятся в коллекции namedViews. Далее нужно обработать ответ пользователя. Выбранный им именованный вид назначаем свойству ViewToPlot активного листа.
И только после этого меняем значение свойства PlotType на acView.

Шаг 5.2. Обработка ключевого слова "Рамка".

```
case "Рамка":
   {
      // Запрос точки левого нижнего угла области печати
      var point1 = ed.GetPoint("Укажите левую нижнюю точку области печати:");

      if (point1.Status == HostMgd.EditorInput.PromptStatus.OK)
      {
      // Вывод в консоль координат выбранной точки
      ed.WriteMessage("({0};{1})", point1.Value.X, point1.Value.Y);

      // Запрос точки правого верхнего угла области печати
      var point2 = ed.GetCorner("Укажите правую верхнюю точку области печати:", point1.Value);

      if(point2.Status== HostMgd.EditorInput.PromptStatus.OK)
      {
         // Вывод в консоль координат выбранной точки
         ed.WriteMessage("({0};{1})", point2.Value.X, point2.Value.Y);

         // Очистка коллекции PlotAreas
         nanoCAD.InanoCADPlot plot = (nanoCAD.InanoCADPlot)comDoc.Plot;
         nanoCAD.InanoCADPlotCustomParams param = plot.CustomPlotSe
```

```
ttings[activeLayout];

        if (param.PlotAreas.Count != 0)
        {
        // Добавление области печати "Рамка" методом
SetWindowToPlot()
        // происходит верно только если коллекция PlotAreas пустая
        param.PlotAreas.Clear();
        plot.CustomPlotSettings[activeLayout] = param;
        }

        // Получение текущего вида
        Teigha.DatabaseServices.ViewTableRecord curView =
ed.GetCurrentView();

        // Получение матрицы преобразования из WCS в DCS
        Teigha.Geometry.Matrix3d matrix;
        matrix = Teigha.Geometry.Matrix3d.PlaneToWorld(curView.
ViewDirection);
        matrix = Teigha.Geometry.Matrix3d.Displacement(
           curView.Target - Teigha.Geometry.Point3d.Origin)* matrix;
        matrix = Teigha.Geometry.Matrix3d.Rotation(-curView.ViewTwist,
curView.ViewDirection, curView.Target) * matrix;
        matrix = matrix.Inverse();

        // Преобразование координат полученных точек в DCS
        double[] ptMin = new double[] { point1.Value.
TransformBy(matrix).X, point1.Value.TransformBy(matrix).Y };
        double[] ptMax = new double[] { point2.Value.
TransformBy(matrix).X, point2.Value.TransformBy(matrix).Y };

        // Добавление новой области печати "Рамка" методом
SetWindowToPlot()
        activeLayout.SetWindowToPlot(ptMin, ptMax);
     }
  }
  else ed.WriteMessage("Cancel");
  break;
        }
```

Если пользователь выбрал область печати "Рамка", необходимо запросить у него две точки этой прямоугольной области печати: левый нижний и правый верхний углы.
В нашу программу координаты точек, указанных пользователем, будут переданы в WCS (World Coordinate System). Для того чтобы область печати правильно отобразилась на экране, координаты этих точек должны быть переведены в DCS (Display Coordinate System). Далее передаем преобразованные координаты точек в метод SetWindowToPlot(), который преобразует координаты двух точек в прямоугольную область печати.
После всех этих действий меняем значение свойства PlotType на acWindow.

Шаг 6. Компилируем наше приложение, загружаем в nanoCAD и запускаем.
Для наглядности приведем текст нашей программы целиком:

```
public partial class Commands
{
  [Teigha.Runtime.CommandMethod("SetPlotArea")]
  public void SetPlotAreaToActiveLayout()
  {
     // Получение ссылки на активный документ
     HostMgd.ApplicationServices.Document doc = HostMgd.
ApplicationServices.Application.DocumentManager.MdiActiveDocument;
     nanoCAD.Document comDoc = doc.AcadDocument as nanoCAD.
Document;

     // Получение ссылки на редактор активного документа
     HostMgd.EditorInput.Editor ed = doc.Editor;

     // Получение ссылки на активный лист документа
     OdaX.AcadLayout activeLayout = comDoc.ActiveLayout;

     // Приглашающее сообщение для выбора области печати
     HostMgd.EditorInput.PromptStringOptions promptOptions = new
HostMgd.EditorInput.PromptStringOptions("Выберите область печати");

     // Запись в ключевые слова приглашающего сообщения наименований доступных
     // типов областей печати, которые возможны и в пространстве модели, и в листах
     promptOptions.Keywords.Add("Экран");
     promptOptions.Keywords.Add("Границы");
     promptOptions.Keywords.Add("Рамка");

     // Для области печати "Вид" создаем коллекцию string,
     // в которую будем записывать названия именованных видов
     System.Collections.Generic.List<string> namedViews = new System.
Collections.Generic.List<string>();

     // Тип области печати "Вид" доступен только
     // если у активного листа есть сохраненные именованные виды
      if (comDoc.Views.Count != 0)
     {
        // Если коллекция Views не пустая, ищем в ней именованные виды активного листа
        foreach (OdaX.AcadView namedView in comDoc.Views)
        {
           // Если найден именованный вид для активного листа,
           // добавляем название этого вида в коллекцию namedViews
           if (namedView.LayoutId == activeLayout.ObjectID)
namedViews.Add(namedView.Name);
        }

        // Если в итоге коллекция namedViews не пустая,
        // добавляем к запросу ключевое слово "Вид"
        if (namedViews.Count != 0) promptOptions.Keywords.
Add("Вид");
     }

     // Добавление к запросу ключевых слов в зависимости от типа активного листа
     if (activeLayout.ModelType) promptOptions.Keywords.
Add("Лимиты");
     else promptOptions.Keywords.Add("Лист");

      // Запрос пользователю с предложением выбрать область печати
     HostMgd.EditorInput.PromptResult result =
ed.GetString(promptOptions);

     // Обработка полученного результата запроса
```

```
    if (result.Status == HostMgd.EditorInput.PromptStatus.Keyword)

        // Составление словаря для сопоставления названий типов областей печати в перечислении OdaX.AcPlotType и в интерфейсе
        System.Collections.Generic.Dictionary<string, OdaX.AcPlotType> dictionary = new System.Collections.Generic.Dictionary<string, OdaX.AcPlotType>();
        dictionary.Add("Экран", OdaX.AcPlotType.acDisplay);
        dictionary.Add("Границы", OdaX.AcPlotType.acExtents);
        dictionary.Add("Лист", OdaX.AcPlotType.acLayout);
        dictionary.Add("Лимиты", OdaX.AcPlotType.acLimits);
        dictionary.Add("Вид", OdaX.AcPlotType.acView);
        dictionary.Add("Рамка", OdaX.AcPlotType.acWindow);

        switch (result.StringResult)
        {
          case "Вид":
          {
            // Приглашающее сообщение для выбора именованного вида
            HostMgd.EditorInput.PromptStringOptions opts = new HostMgd.EditorInput.PromptStringOptions("Выберите именованный вид");

            // Запись названий именованных видов активного листа
            // в ключевые слова запроса
            namedViews.ForEach(nv => opts.Keywords.Add(nv));

            // Запрос пользователю с предложением выбрать именованный вид
            HostMgd.EditorInput.PromptResult res = ed.GetString(opts);

            // Обработка полученного результата запроса
            if (res.Status == HostMgd.EditorInput.PromptStatus.Keyword)
              activeLayout.ViewToPlot = res.StringResult;
            break;
          }
        case "Рамка":
          {
            // Запрос точки левого нижнего угла области печати
            var point1 = ed.GetPoint("Укажите левую нижнюю точку области печати:");
            if (point1.Status == HostMgd.EditorInput.PromptStatus.OK)
            {

            //Вывод в консоль координат выбранной точки
            ed.WriteMessage("({0};{1})", point1.Value.X, point1.Value.Y);

            // Запрос точки правого верхнего угла области печати
            var point2 = ed.GetCorner("Укажите правую верхнюю точку области печати:", point1.Value);

            // Обработка результата запроса
            if(point2.Status== HostMgd.EditorInput.PromptStatus.OK)
            {
              // Вывод в консоль координат выбранной точки
              ed.WriteMessage("({0};{1})", point2.Value.X, point2.Value.Y);
              // Очистка коллекции PlotAreas
              nanoCAD.InanoCADPlot plot = (nanoCAD.InanoCADPlot)comDoc.Plot;
              nanoCAD.InanoCADPlotCustomParams param = plot.CustomPlotSettings[activeLayout];
              if (param.PlotAreas.Count != 0)
              {
                // Добавление области печати "Рамка" методом SetWindowToPlot()
                // происходит верно только если коллекция PlotAreas пустая
                param.PlotAreas.Clear();
                plot.CustomPlotSettings[activeLayout] = param;
              }

              // Получение текущего вида
              Teigha.DatabaseServices.ViewTableRecord curView = ed.GetCurrentView();

              // Получение матрицы преобразования из WCS в DCS
              Teigha.Geometry.Matrix3d matrix;
              matrix = Teigha.Geometry.Matrix3d.PlaneToWorld(curView.ViewDirection);
              matrix = Teigha.Geometry.Matrix3d.Displacement(curView.Target - Teigha.Geometry.Point3d.Origin) * matrix;
              matrix = Teigha.Geometry.Matrix3d.Rotation(-curView.ViewTwist, curView.ViewDirection, curView.Target) * matrix;
              matrix = matrix.Inverse();

              // Преобразование координат полученных точек в DCS
              double[] ptMin = new double[] { point1.Value.TransformBy(matrix).X, point1.Value.TransformBy(matrix).Y };
              double[] ptMax = new double[] { point2.Value.TransformBy(matrix).X, point2.Value.TransformBy(matrix).Y };

              // Добавление новой области печати "Рамка" методом SetWindowToPlot()
              activeLayout.SetWindowToPlot(ptMin, ptMax);
            }
          }
          else ed.WriteMessage("Cancel");
          break;
        }
      }

      // Поиск в словаре выбранного типа области печати и запись
      // найденного значения в переменную ptResult
      OdaX.AcPlotType ptResult;
      dictionary.TryGetValue(result.StringResult, out ptResult);

      // Назначение выбранной области печати активному листу
      activeLayout.PlotType = ptResult;
      ed.WriteMessage("Листу '{0}' назначена область печати '{1}'", activeLayout.Name, result.StringResult);

      }
      else ed.WriteMessage("Область печати не изменена");
    }
}
```

После запуска команды SetPlotArea пользователь увидит запрос о выборе типа области печати, дальше все зависит от выбора пользователя.
Если выбрана "Рамка", необходимо указать левый нижний и правый верхний углы прямоугольной области печати. После того как пользователь указал две точки, создается область печати – методом SetWindowToPlot(), а в консоль nanoCAD выводится сообщение об успешном выполнении команды (рис. 3).

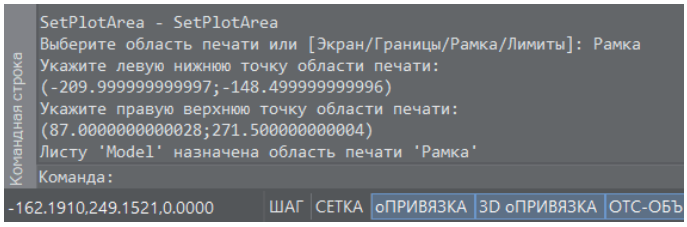

Рис. 3

Через интерфейс nanoCAD назначенную область печати можно увидеть, вызвав окно *Параметры листа*. Область будет обозначена в пространстве чертежа красной пунктирной линией (рис. 4).

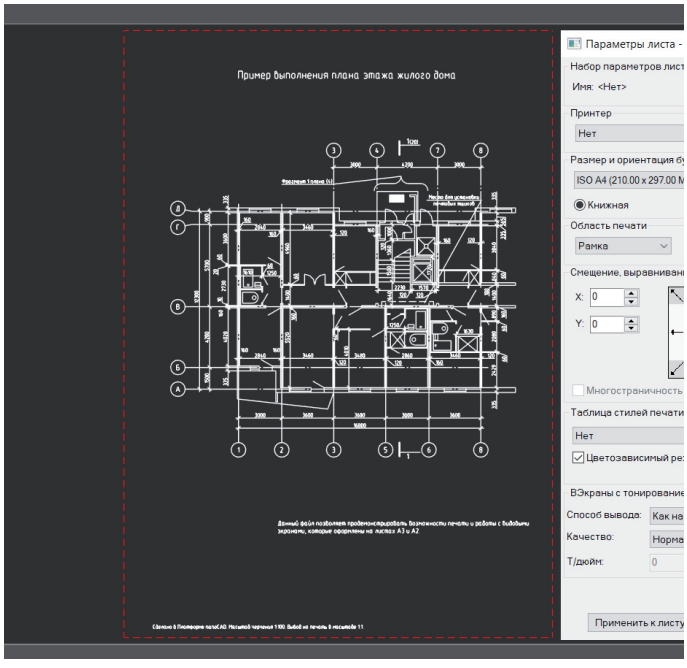

Рис. 4

При повторном запуске команды SetPlotArea и выборе типа области печати "Рамка" назначенная в первый раз область печати будет заменена новой.
Если у активного листа есть именованные виды, пользователь может выбрать тип области печати "Вид". После этого команда запросит, какой именно именованный вид назначить (рис. 5).

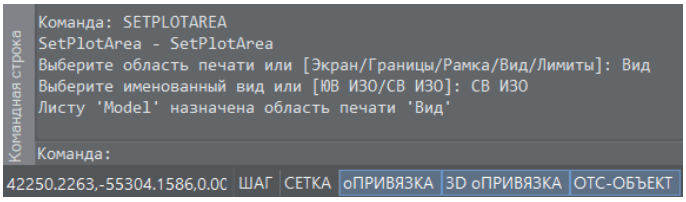

Рис. 5

Итак, в этой статье мы подробно разобрали шаги, необходимые для создания .NET-приложения, работающего с областями печати документа nanoCAD.
В приложении был создан фильтр доступных для выбора областей печати, который учитывает тип активного листа и его текущие настройки. Также был подробно представлен алгоритм работы с областями печати "Рамка" и "Вид", более сложными с точки зрения использования через API nanoCAD, чем остальные области печати. В следующей статье подробнее рассмотрим изменение расширенных настроек печати через API nanoCAD.

Светлана Мирончик,
Клуб разработчиков nanoCAD
ООО "Нанософт разработка"

ReClouds – приложение для обработки данных 3D-сканирования в интеграции с Платформой nanoCAD 22

С июля 2022 года цифровая модульная платформа ReClouds для обработки данных 3D-сканирования (разработчик – "СиСофт Девелопмент") доступна для массового корпоративного использования в виде отдельного специализированного приложения к российской САПР-платформе nanoCAD.

ReClouds представляет собой программный продукт, построенный на открытой архитектуре и предназначенный для создания инженерной экосистемы приложений, выполняющих обработку данных 3D-сканирования в области геодезии, строительства, машиностроения, инфраструктурного и метрологического мониторинга.

Полнофункциональная работа ReClouds возможна в интеграции с Платформой nanoCAD и дополнительным модулем "3D", что позволит создать единую САПР-среду и решать на ее основе следующие прикладные инженерные и информационные задачи:

- строительство и эксплуатация инженерных сооружений, зданий, коммуникаций;
- создание 2,5D-чертежей и планов, в том числе топографических;
- мониторинг чрезвычайных ситуаций и экологической обстановки;
- трехмерное моделирование, в том числе имитационное;
- выполнение оперативных расчетов и измерений;
- моделирование задач транспортировки;
- проектирование машин и механизмов;
- поиск коллизий, авторский надзор;
- наполнение ГИС-систем;
- работы в области метрологии.