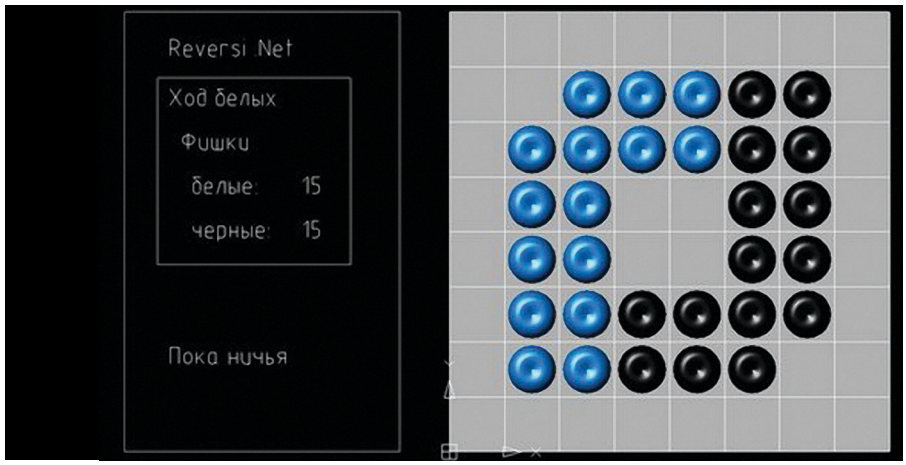
ВАШ ХОД, ТОВАРИЩ .NET, ИЛИ ОПЯТЬ "РЕВЕРСИ" ПОД NANOCAD



Некоторое время назад у нас произошло большое событие – выход релиза nanoCAD 3.5. Ключевым нововведением стало открытое API, о котором и пойдет речь в этой статье.

Как известно, лучший способ что-то изучить – это сделать его. Когда-то я писал "Реверси" под nanoCAD на скрипте. Теперь решил написать на .NET. В результате получилось кросс-САПР-платформенное приложение, способное работать не только под nanoCAD.

Программировать под nanoCAD можно было и раньше. dows писал на скриптах кривые Серпинского, я писал "Реверси" , есть и другие примеры. Все это, конечно, хорошо, но мало. Поэтому мой следующий ход – .NET.

Entry level

Первое, что нужно было сделать, – создать сборку, содержащую код, исполняемый в nanoCAD:

- создаем проект: Visual C#, Class Library;
- добавляем в References библиотеки .NET nanoCAD: hostdbmgd.dll, hostmgd.dll;
- регистрируем в nanoCAD команду.

Метод, который будет регистрироваться в качестве команды, должен иметь модификатор public и быть помечен специальным атрибутом CommandMethod.

Например, HelloWorld выглядит так:

```
[CommandMethod("HelloWorld")]
public void HelloWorld ()
{
  Editor ed =
Platform.ApplicationServices.Application.Do
cumentManager.MdiActiveDocument.Editor;

  // Выводим в командную строку сообщение
  ed.WriteMessage("Добро пожаловать в управляемый код nanoCAD!");
}
```

И ВСЁ!

Подробнее не пишу: об этом можно прочитать в nanoCAD SDK. Где взять? В Клубе разработчиков nanoCAD, регистрация открыта.

Структура

Игру я разделил на несколько классов: класс игры, класс игровой доски, класс информационной панели, класс игровой фишки:

- класс игры должен содержать алгоритмы проверки возможности сделать ход по определенным координатам, поиска хода компьютера, подсчета фишек игроков, решения о продолжении игры;
- класс доски – отрисовывать доску, хранить ее содержание;
- класс информационной панели – показывать результаты прохождения партии;
- класс фишки – отрисовывать фишку, уметь менять ее цвет, хранить всю информацию, касающуюся конкретной игровой ячейки.

Каждый класс должен быть максимально самостоятельным.

Дальше мне нужно было научиться создавать объекты, менять их и общаться с пользователем.

Создание объектов. Матчасть

Прежде чем рисовать "Реверси", требовалось понять – что делать, за что браться.

Для того чтобы создать объекты, нужно немного знать о структуре документа. В каждом документе есть база данных. В базе данных хранятся объекты, содержащиеся в чертеже, и их связи друг с другом. Здесь хранится всё: и линии с дугами, и пространство модели, и стили текстов, и многое другое. Добавляя новый объект в чертеж, нужно добавить его в базу данных. А где есть база данных, там есть и транзакции.

Транзакции нужны, чтобы защитить наш документ: если в результате выполнения кода случится сбой, объекты, добавленные этим кодом, не попадут в документ – транзакция будет отменена. Если все завершилось успешно – транзакция подтверждается и объекты будут добавлены.

```
Database db =
Application.DocumentManager.MdiActive
Document.Database;
TransactionManager tm =
db.TransactionManager;

using (Transaction tr =
tm.StartTransaction())
{
  ...
  tr.Commit();
}
```

Просто создать объект мало. Он останется никуда не присоединенным, висящим в воздухе. Объект нужно куда-то поместить. Обычно это модельное пространство. В скриптах было что-то похожее (сказал модельному пространству: "Сделай линию" – и она там появится). В .NET немного по-другому: созданный объект нужно добавить в модельное пространство и в транзакцию.

```
using (Transaction tr =
tm.StartTransaction())
{
  BlockTable bt =
tr.GetObject(db.BlockTableId,
OpenMode.ForRead, false) as BlockTable;
  BlockTableRecord ms =
tr.GetObject(bt[BlockTableRecord.ModelS
pace], OpenMode.ForWrite, false) as
BlockTableRecord;
```

[1] См.: CADmaster №2/2010, с. 15-17.

```
  Line line = new Line();
  ObjectId lid = ms.AppendEntity(line); //
добавляем в модельное пространство
  tr.AddNewlyCreatedDBObject(line, true);
// и в транзакцию
  tr.Commit();  // сохраняем изменения
}
```

Каждый объект, добавленный в базу данных, однозначно определяется по личному коду – ObjectId. Используя ObjectId, можно открывать объекты для чтения или записи.

Создание объектов-2. Полный вперед

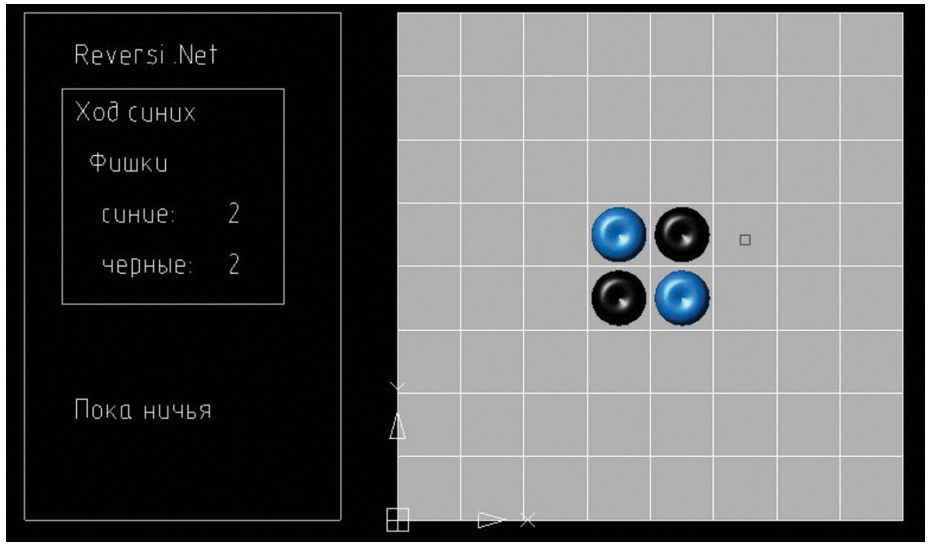
Вооружившись знаниями о внутренней кухне документа, можно, наконец, начинать разработку класса игровой доски. Нет доски – нет и партии. Поэтому первое, что я начал делать, – стал рисовать клетки в пространстве документа.

Клетки я делал из штриховок. Открыв в NCadSDK.chm описание объекта Hatch (документация входит в SDK, доступный членам Клуба разработчиков), я почерпнул нужные мне знания. Третий абзац сразу сообщил мне, что штриховка состоит из петель, а список методов объекта штриховки подсказал магическое слово AppendLoop(). "Вот то, что мне нужно!" – подумал я.

Итак, каждую клетку я строил из квадратной полилинии, которую закрашивала штриховка. Все штриховки вместе образовывали квадрат 8 на 8 клеток.

Дальше – по накатанной, всё как в прошлый раз: бордюры и фишки создаю из объектов 3Dmesh. Бордюр – это полигон две на две вершины. Вычисляю координаты вершин, создаю их, добавляю в сеть, сеть добавляю в модель.

```
using (Transaction tr =
tm.StartTransaction())
{
   // создаем сеть
  PolygonMesh mesh = new
PolygonMesh();
  mesh.NSize = 2;
  mesh.MSize = 2;
  ms.AppendEntity(mesh);
  tr.AddNewlyCreatedDBObject(mesh,
true);
   // создаем и добавляем вершины
  AddVertexToMesh(mesh, new
Point3d(col*gridstep, 0,-linehight), tr);
  AddVertexToMesh(mesh, new
Point3d(col*gridstep, 0, linehight), tr);
  AddVertexToMesh(mesh, new
Point3d(col*gridstep,8*gridstep,-line-
hight), tr);
  AddVertexToMesh(mesh, new
Point3d(col*gridstep,8*gridstep,line
hight), tr);
```




```
  tr.Commit();
}

// создаем вершину сети
private void
AddVertexToMesh(PolygonMesh
PolyMesh, Point3d Pt3d, Transaction
Trans)
{
   PolygonMeshVertex PMeshVer = new
PolygonMeshVertex(Pt3d);
  PolyMesh.AppendVertex(PMeshVer);

Trans.AddNewlyCreatedDBObject(PMesh
Ver, true);
}
```

Отлично. Клетки есть, разделители есть. Нарисовать фишку теперь тоже нетрудно. Формулы вычисления координат вершин шарика я взял из скриптовой версии игры. Правда, подправил их, чтобы объект больше походил на игровую фишку "Реверси".

Что у меня получилось в результате – смотрите на рисунке.

"Я полсотни третий, выхожу на квадрат"

Теперь нужно научиться реагировать на действия пользователя.

Тут снова понадобится вспомнить матчасть. Кроме базы данных, есть еще несколько объектов, которые относятся не к самому документу, а к приложению в целом. Это, например, объект Application – коллекция всех документов, открытых в приложении DocumentCollection. И объект взаимодействия с пользователем – Editor. Есть и другие, но их я сейчас не касаюсь.

У объекта Editor есть ряд методов для взаимодействия с пользователем: запрос объектов, запрос строки, числа, области. Запрос объекта осуществляется методом GetEntity(PromptEntityOptions). Объект PromptEntityOptions – это необязательные параметры. Через этот объект задаются строка приглашения, ключевые слова (если они нужны), выставляются ограничения на выбор объектов. Подобный объект принимают все методы ввода.

Принцип хода остался прежним: пользователь выбирает клетку, куда хочет пойти. Клетка – это объект "Штриховка". Поэтому указываю, что принимаем в качестве ввода только объекты штриховки, пустой выбор – запретить, обязательно должен быть объект. И пишем строку приглашения.

```
Editor ed =
Application.DocumentManager.MdiActive
Document.Editor;

ObjectId selectObj = ObjectId.Null;
PromptEntityOptions opts = new
PromptEntityOptions("Ваш ход.Укажите
ячейку");
opts.SetRejectMessage("\nТолько ячейка
может быть выбрана");
opts.AddAllowedClass(typeof(Hatch),
false);
PromptEntityResult pr =
ed.GetEntity(opts);
```

По клетке определяется, куда именно пользователь хочет поставить свою фишку. Далее алгоритм проверяет, можно ли это сделать. Если да – ход выполняется и нужные фишки переворачиваются.

Перекрашивание существующих фишек

Как уже сказано, все объекты живут внутри базы данных. Это значит, что для того чтобы прочитать или изменить свойства какого-либо объекта, этот объект нужно открыть. Открытие объектов происхо-

дит методом транзакции GetObject(). По завершении изменений транзакция подтверждается.

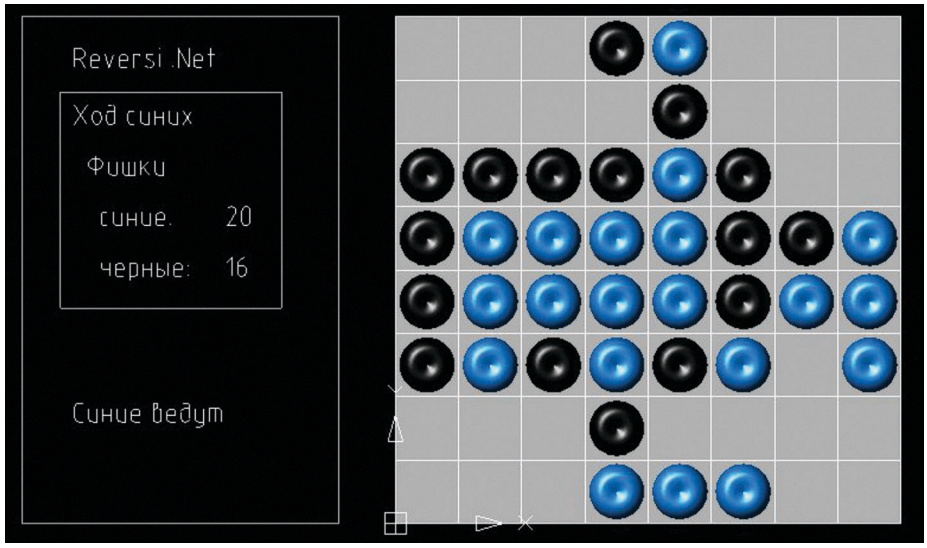
```
using (Transaction myT =
db.TransactionManager.StartTransaction())
{
 // pieceId – это id перекрашиваемой фишки в БД
 // открываем объект pieceId для изменений – OpenMode.ForWrite
 PolygonMesh piece =
myT.GetObject(this.pieceId,
OpenMode.ForWrite) as PolygonMesh;
 // присваиваем цвет в зависимости от того, чья фишка
 piece.Color = (player == ePlayer.Human)
? Constants.HumanColor:
Constants.PcColor;

 // подтверждаем транзакцию
 myT.Commit();
}
```

Вкусняшки

Для хранения игровой доски в памяти я сделал две структуры данных: массив и словарь. Массив хранит образ доски 8 на 8, а словарь соответствия: элемент клетки – ObjectId-штриховки. Обе структуры хранят ссылки на объекты игровой доски. При таком подходе можно не заботиться о синхронизации. Меняться будет только элемент Piece. А получить его всегда можно по ссылке. Не важно, из массива или из словаря.

```
Dictionary<ObjectId, Piece> GameDesc =
new Dictionary<ObjectId, Piece>();
Piece[,] GameDesc_xy = new Piece[8, 8];
```

На .NET многие вещи мне удалось сделать красивее и проще, чем на скриптах. Возможности фрэймворка несли приятные вкусности. К примеру, с использованием LINQ структуры данных обрабатывались почти сами собой. Подсчет количества фишек пользователя – в одну строку. Выбор клетки для хода компьютера – один запрос. Красота!




```
int GetCounterCount(ePlayer player)
{
 // подсчет фишек игрока player
  return gamedesk.GameDesc.Where(x =>
x.Value.Player == player).Count();
}
```

Компиляция и запуск игры

Исходники игры можно взять тут: *http://ftp.nanocad.ru/habr/reversiMgd/MgdReversi.zip*. Откройте проект в Visual Studio или SharpDeveloper и скомпилируйте. Пути проекта настроены с расчетом на то, что nanoCAD установлен в стандартную директорию.

Если исходники вам не нужны, а хочется просто посмотреть на "Реверси", можно скачать собранный нами модуль (*http://ftp.nanocad.ru/habr/reversiMgd/MgdReversi_dll.zip*).

Для запуска игры загрузите сборку *MgdReversi.dll* в nanoCAD командой NETLOAD. Теперь можно запускать игру командой PLAY.

Что не успел сделать

Было бы интересно, играя в nanoCAD, остановиться на середине партии, сохранить игру в файл, открыть этот файл в AutoCAD и доиграть – ведь формат файла в обеих системах один и тот же.

Но для этого понадобится переделать архитектуру приложения: сейчас информация о состоянии игры хранится в памяти команды, а нужно ее сохранять в объектах чертежа (поле, фишки), которые сохраняются в файл. Оставим это на будущее.

А до тех пор можно играть в "Реверси" не останавливаясь, от начала и до конца игры, что под AutoCAD, что под nanoCAD – и там и там игра работает одинаково. Достаточно лишь пересобрать "Реверси" под AutoCAD – используя его SDK, ObjectARX, это несложно.

Андрей Грачевский

AutoCAD WS будет работать с 3D-моделями, GPS-координатами и удаленной печатью

НОВОСТИ

На Autodesk University были представлены новые возможности облачной платформы для просмотра чертежей

Компания Autodesk объявила о том, что планирует добавить ряд новых функций в следующую версию AutoCAD WS – бесплатного приложения, позволяющего просматривать выполненные в AutoCAD проекты на мобильных устройствах, редактировать их и обмениваться ими в Интернете. Пользователи решения получат доступ к новым возможностям, благодаря которым, в частности, будет обеспечена работа с 3D-моделями. Кроме того, разработчики AutoCAD WS планируют включить в продукт поддержку GPS для сохранения в проектах информации об их географическом местоположении, а также возможность мгновенной облачной печати на любом принтере HP (в том числе семейства Designjet). На данный момент зарегистрировано уже более трех миллионов загрузок AutoCAD WS пользователями. Продукт помогает наладить успешное сотрудничество между архитекторами, инженерами и теми, кто не работает в САПР.

Используя AutoCAD WS, участники проектных групп и другие заинтересованные лица, даже находясь вне офиса, могут просматривать, редактировать и передавать друг другу чертежи. В новой версии функциональность AutoCAD WS будет расширена следующим образом:

Интерактивные 3D-функции. Пользователи AutoCAD WS получат возможность просматривать 3D-модели и обмениваться ими на мобильных устройствах.

Интеграция с GPS. Применяя GPS-возможности, которыми обладает большинство мобильных устройств, пользователи смогут лучше ориентироваться в чертежах – в частности, снабжать фрагменты чертежей комментариями об их географическом местоположении, находясь на проектируемых объектах.

Печать с помощью службы HP ePrint & Share. Регистрация в бесплатной онлайн-службе HP ePrint & Share предоставит пользователям AutoCAD WS возможность удобной печати проектов на подключенных к Интернету принтерах HP Designjet. При этом будет обеспечиваться полный контроль над стилями печати, чертежными форматами, компоновками листов и т.п.