nanoCAD и сбоку бантик

или Программируем "Реверси" под CAD-платформу

Некоторое время назад в блоге "Нанософта" на портале habrahabr.ru была опубликована статья о программировании под nanoCAD. В тот же день прошла информация о программировании "Реверси" на Python. Спустя неделю "Реверси" были написаны уже на Silverlight, чуть позже – на Tcl/Tk. Можно смело сказать, что игра "Реверси" пошла в народ.

Как раз в это время я занимался изучением возможностей скриптов под nanoCAD. Изучение требовало практики. И я подумал – а чем я хуже? И решил написать эдакий "нано-боян". Так появились 3D Reversi.

Скриптовые возможности CAD-платформ привлекали меня всегда. С помощью скрипта прикручиваешь к большой системе собственный функционал – и дальше используешь уже не только то, что "зашито" в систему разработчиками.

Для изучения симбиоза nanoCAD и JScript я стал писать "несерьезный" скрипт под САПР-систему.

Начало

Разработку игры я мысленно разделил на три части: поле игры (интерфейс), интерактив (взаимодействие с пользователем) и сами игровые алгоритмы.

Для разработки игрового поля необходимо было создавать объекты в пространстве модели nanoCAD. Без документации все попытки писать "что-то под что-то" обречены, так что первое, что я сделал, – стал искать описание объектов и функций. Я спросил у Яндекса (почти как в песне) и получил множество ссылок на разные сайты, описывающие объектную модель AutoCAD. К примеру, на сайт *vbamodel.narod.ru*.

Создание объектов

Добавление объектов происходит с помощью коллекции ThisDocument.ModelSpace. Сразу оговорюсь, что я писал на JScript потому как программирую на C++ и синтаксис JS мне ближе, чем синтаксис VB Script.

Поле представляет собой 64 клетки, сетку и фишки. Сетку и фишки я сделал объемными фигурами. А для клеток поля использовал штриховки.

```
//Создаем рамку для штриховки (переменная shag задает сторону клетки).
        var polyline;
        var RefPoly = new Array(0,0,0, shag,0,0, shag,shag,0, 0,shag,0);

        polyline = ms.AddPolyline(ut.CreateTypedArrayFromJSArray(5, RefPoly));
        polyline.Closed = true;

//После чего заштриховываем рамку:
        hatch = ms.AddHatch(1,"SOLID",false,0);
        hatch.AppendOuterLoop(polyline);
```

Потом я размножил клетки и расставил их по местам.

Дальше пришла очередь бордюров и игровых фишек. Полностью копировать игру, написанную на Питоне, было неинтересно – требовалась изюминка. Так как nanoCAD может отображать объекты в пространстве, такой изюминкой стала объемность доски и игровых фишек. Поэтому бордюры и фишки я сделал объектами 3DMesh.

Каждое ребро – это просто прямоугольник. А каждая фишка – сфера.

Объемные объекты – это объекты типа 3DMesh. Они создаются методом Add3DMesh(...) коллекции ModelSpace. Это сеть размером M на N клеток. Для ее создания нужно указать в массиве координаты всех вершин сети. По рядам: сперва точки первого ряда, потом второго – и так до M.

```
//Строим массив координат точек сети:
        var mesh = new Array();
        var n = 0;
        for (i=-1;i<2; ++i)
                for (j=0;j<9; ++j)
                {
                        mesh[n]  =0;
                        mesh[++n]=shag*j;
                        mesh[++n]=shag*i/10;
                        n++;
                }
// Создаем прямоугольник, заданный координатами, лежащий в плоскости YZ.
var mesh3d = ms.Add3DMesh(3,9,ut.CreateTypedArrayFromJSArray(5,mesh));
```

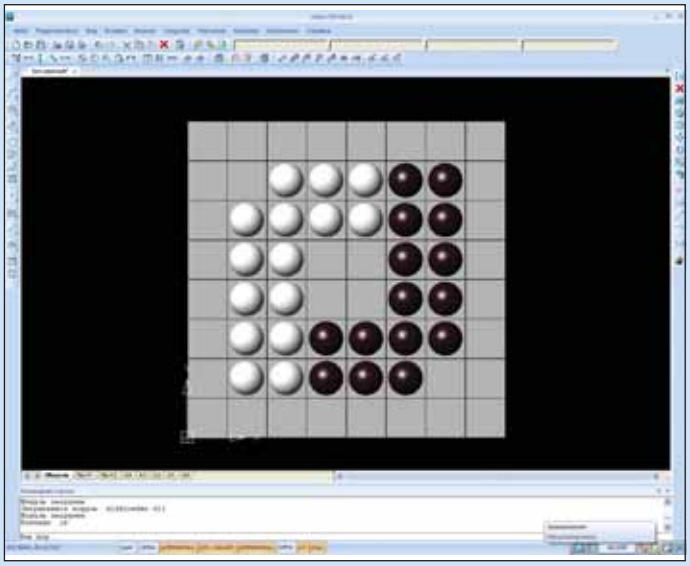

Рис. 1

Потом все же добавил ребрам толщину, чтобы они не исчезали при виде сверху.

Построить сферу оказалось посложнее, пришлось вспомнить аналитическую геометрию времен первого курса. Нет, в библиотеку я не пошел. Пошел в Википедию и узнал, как сферические координаты выражаются через декартовы.

```
        var Vertexs = new Array();
        var idx =-1;
        var r =22;
        for(j=0; j <= 20; ++j)
        for(i=0; i < 20; ++i)
        {
                beta = Math.PI / 19 * j;
                alfa = 2 * Math.PI / 20 * i;
                Vertexs[++idx] = 25 + r*(Math.cos(alfa)*Math.sin(beta)); Vertexs[++idx] = 25 + r*(Math.sin(alfa)*Math.sin(beta)); Vertexs[++idx]
                        = r*Math.cos(beta);
        }
        fishka_b = ms.Add3DMesh(20,20,ut.CreateTypedArrayFromJSArray(5,Vertexs));
```

В результате этих действий получилось такое игровое поле (рис. 1).

Взаимодействие с пользователем

Следующий шаг – организация взаимодействия скрипта с пользователем. Нужно спросить, куда игрок хочет поставить свою фишку. Этим занимается функция GetEntity().

```
// Смотрим, какие объекты выбраны, до тех пор пока не выбран один объект "Штриховка"

        while (hatch==null)
        {
                ut.GetEntity(RefPoly,null,"Ваш ход");
                if (RefPoly.length == 1 && RefPoly [0].EntityName == "AcDbHatch")
                {
                        // убеждаемся что эта штриховка – это клетка поля. Она хранит координаты клетки.
                        if (RefPoly[0].Hyperlinks.Count > 0)
                                hatch = RefPoly [0];
                }

                x = parseInt (hatch.Hyperlinks.Item(0).URLDescription);
                y = parseInt (hatch.Hyperlinks.Item(0).URLNamedLocation);
                }
```

Тут есть хитрость: чтобы не высчитывать координаты каждой из штриховок, я при создании заложил в каждую из них информацию о местоположении на поле.

```
        // здесь i, j – это координаты клетки
        hatch.Hyperlinks.Add("xy",i.toString(),j.toString());
```

Когда пользователь выбирает клетку для хода, я получаю эти координаты обратно:

```
x = parseInt (hatch.Hyperlinks.Item(0).URLDescription);
y = parseInt (hatch.Hyperlinks.Item(0).URLNamedLocation);
```

Подробнее останавливаться на алгоритмах не буду. Писать суперумный AI я не стал, а функцию принятия решения скопировал из исходника "Реверси" на Питоне. Надо сказать, что язык Питон я видел впервые. Всё вполне читается. Адаптация к JScript заняла у меня минут десять.

Нюансы и хитрости

Теперь о нюансах скриптостроения под nanoCAD.

Не знаю точно, как в VBS, а в JScript все переменные передаются в функции по значению, а не по ссылке. Что это значит? Это значит, что если вы указываете при вызове метода переменную JScript и ожидаете, что после выполнения метода вам вернется результат, вас ждет разочарование – этого не произойдет. Переменная не изменится. Так случится, например, с методом GetEntity(Object, PickedPoint [, Prompt]). Первым параметром в нем является переменная, через которую возвращается набор выбранных объектов. Но в JScript это не работает. Как обойти это ограничение? Нужно передать вместо переменной массив.

```
var RefPoly = new Array();
ut.GetEntity(RefPoly,null,"Выберите объект");
var entity = RefPoly[0]; // В массиве лежат выбранные элементы.
```

Второй важный момент заключается в том, что методы, связанные с координатами (например, создание объектов), не принимают скриптовые массивы. Так как JS и VBS – языки слабо типизированные, то их массивы могут содержать элементы разных типов. Это недопустимо. Для приведения к типизированному массиву служат функции объекта Utility CreateTypedArray(varArr,Type,ParamArray) и CreateTypedArrayFromJSArray(Type As Long, varJSArray). Здесь Type – это указание типа элементов массива, а JSArray – сам массив.

```
// Создаем js-массив с координатами вершин полилинии
var RefPoly = new Array(0,0,0, shag,0,0, shag,shag,0, 0,shag,0);

// Создаем полилинию, передавая в нее координаты вершин, преобразованные к TypedArray.
var polyline = ms.AddPolyline(ut.CreateTypedArrayFromJSArray(5,RefPoly));
```

Запуск скрипта

Для правильного отображения элементов игры нужно задать стиль освещения моделей. Стиль задается в меню *Вид ➔ Стиль*. Я задаю стиль "Точное" без показа ребер.

Чтобы запустить "Реверси", пишем в командной строке nanoCAD команду "JS". А потом указываем полный путь к скрипту nanoReversi.js.

Результат

Три вечера – и я достиг цели: игра заработала под nanoCAD (рис. 2).

В изомерии "Реверси" выглядят гораздо необычнее (рис. 3). Я был полностью доволен.

Несмотря на то что "Реверси" я написал что называется "just for fun", скрипт демонстрирует возможности ActiveX Automation nanoCAD. А если вспомнить, что в скриптах можно использовать и другие ActiveX-серверы, например ADO для подключения к базам данных, то скриптовые задачи уже не кажутся чем-то несерьезным.

Андрей Грачевский

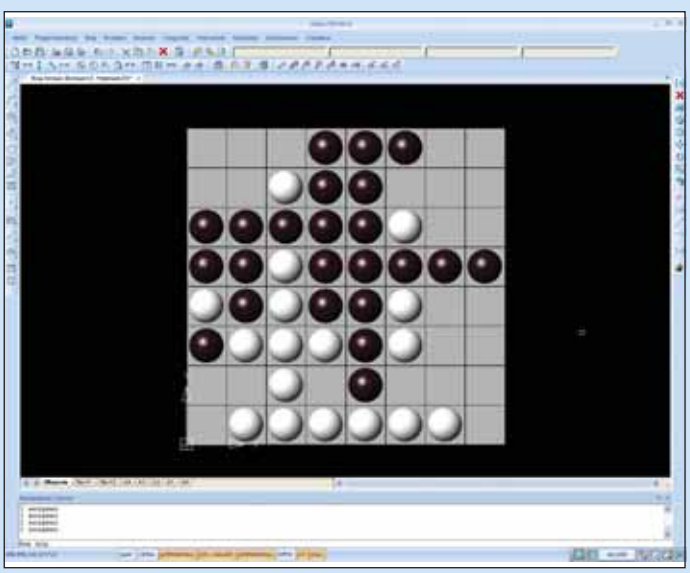

Рис. 2

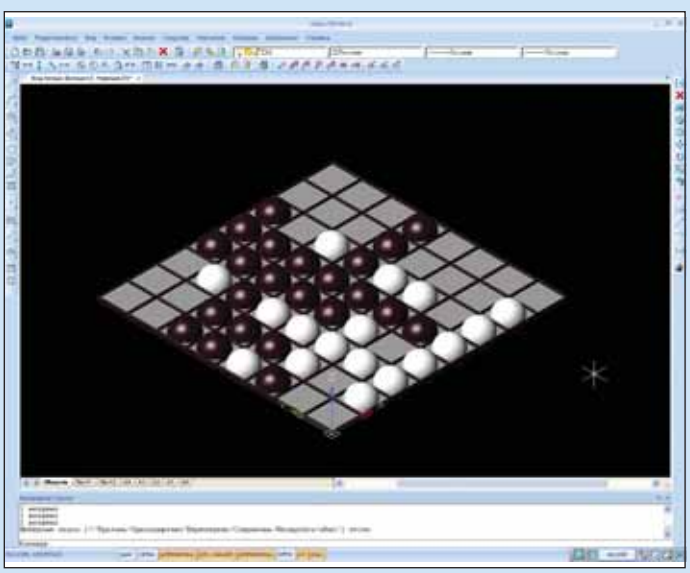

Рис. 3