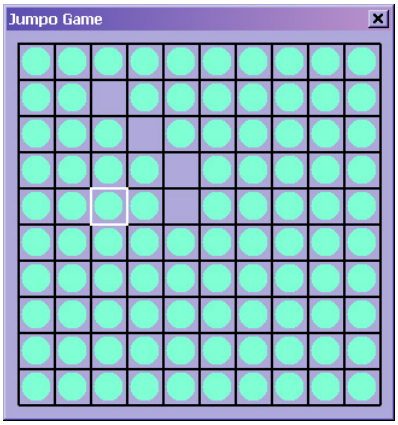
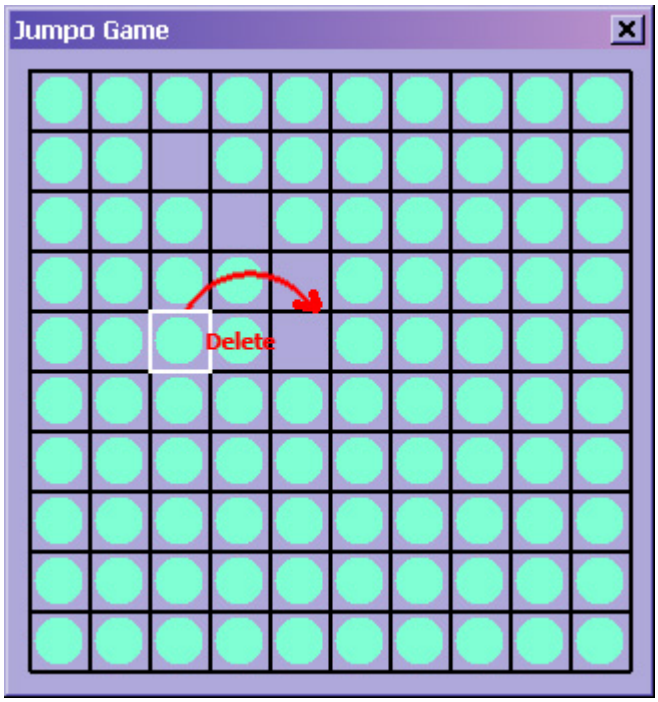
ساخت بازی با GDI+ در C#- قسمت اول
Jumpo در یک ساعت



بازی Jumpo

بازی Jumpo یک بازی صفحه ای (Board Game) است. اساس این بازی بدین صورت است که در ابتدا همه خانه های جدول به جز یک خانه که به شکل تصادفی انتخاب شده است را پر می کنیم (مهره ها را می چینیم). سپس هر کدام از مهره ها که دو مهره بعد از آن (افقی یا عمودی) خالی باشد می تواند از روی مهره بعد از خود پرش کرده و باعث حذف آن شود. خود مهره نیز در خانه خالی قرار می گیرد. به شکل زیر نگاه کنید:



مهره انتخاب شده به درون مهره ای که با فلش نشان داده شده است منتقل شده و مهره بعد از آن (در جهت حرکت) حذف می شود. در این شکل مهره ای که بعد از حرکت حذف می شود با کلمه Delete روی آن نمایش داده شده است.

مقدمه ای بر GDI+

- آشنایی با GDI+

GDI+ در NET Framework. به بخشی از NET Framework. گفته می شود که امکاناتی را برای گرافیکهای برداری دو بعدی، کار با تصاویر و چاپ را فراهم می کند.
پنج فضای نامی (Namespace) در GDI+ وجود دارد که عبارتند از:

```
System.Drawing;
System.Drawing.Drawing2D;
System.Drawing.Imaging;
System.Drawing.Printing;
System.Drawing.Text;
```

فضای نامی System.Drawing حاوی فضاهای نامی دیگر و همچنین اعضای مختلف در زمینه گرافیک می باشد. فضای نامی System.Drawing.Drawing2D حاوی اعضایی است که بصورت عمومی در گرافیک دو بعدی مورد استفاده است. فضای نامی System.Drawing.Imaging حاوی اعضایی برای کار با فایلهای تصویری می باشد. فضای نامی System.Drawing.Printing همانطور که از نام آن نیز مشخص است، حاوی اعضایی برای اعمال چاپ می باشد. فضای نامی System.Drawing.Text حاوی اعضایی برای کار با فونتها و متن می باشد.

کلاس Graphics

در GDI+ همه ترسیمات در کلاس Graphics انجام می شود. در حقیقت می توان کلاس Graphics را مانند بوم نقاشی تصور کرد. هر شی Graphics مرتبط با یک محتوای ابزار (DC) می باشد

یک Graphics را به دو روش می توان بدست آورد، می توان با استفاده از فراخوانی متد CreateGraphics هر کدام از کنترلها و یا مدیریت رویداد OnPaint کنترلها بدست آورد.

اعضای کلاس Graphics که در این مثال مورد استفاده قرار گرفته اند عبارتند از:

DrawLine: برای رسم یک خط بکار می رود. در این مثال برای رسم خطوط صفحه بازی استفاده شده است.
FillEllipse: برای رسم یک بیضی توپر استفاده می شود. در این مثال از این متد برای رسم مهره ها که به شکل دایره هستند استفاده شده است.
DrawRectangle: برای رسم چهارضلعی بکار می رود. در این مثال از این متد برای رسم یک چهارضلعی به دور خانه انتخاب شده استفاده شده است.

کلاس Pen

Pen کلاسی است که حالت و اندازه قلم برای رسم خط را مشخص می کند. به عنوان مثال کد زیر قلمی برای رسم خط با رنگ سفید و ضخامت 2 پیکسل را تعیین می کند:

```
System.Drawing.Pen RectPen = new System.Drawing.Pen(Color.White, 2);
```

کلاس Brush

Brush کلاسی است که مشخصات و نحوه پر کردن یک شکل را مشخص می کند. انواع Brush های مختلفی وجود دارد که ما در این مثال از کلاس SolidBrush استفاده کرده ایم. کلاس SolidBrush یک Brush تک رنگ را تعیین می کند. به عنوان مثال کد زیر یک SolidBrush را با رنگ زرد مشخص کرده است:

```
System.Drawing.SolidBrush EllipseBrush = new System.Drawing.SolidBrush(Color.Yellow);
```

ساختمان داده Point

این ساختمان داده برای ذخیره مختصات دو بعدی یک نقطه بکار می رود. به مثال زیر توجه کنید:

```
System.Drawing.Point BasePoint = new System.Drawing.Point(10, 10);
```

ساختمان داده Rectangle

این ساختمان داده برای ذخیره محل و اندازه چهارضلعی بکار می رود. مثال زیر یک چهارضلعی با موقعیت (10,30) و عرض 50 و طول 50 را تعریف می کند:

```
new System.Drawing.Rectangle(10, 30, 50,50)
```

بازسازی صفحه بازی

از آنجایی که در این بازی تا زمانی که کاربر حرکتی انجام نداده است نیازی به بازسازی صفحه نمی باشد بنابراین از مکانیزم Game Loop استفاده نمی کنیم. زمانی که ویندوز تشخیص بدهد که قسمتی و یا تمام پنجره ما نیاز به بازسازی دارد (مثلا زمانی که پنجره ما Minimize شده و دوباره به حالت عادی برگردد)، رویداد OnPaint رخ می دهد بنابراین ما باید در این رویداد کد مربوط به رسم صفحه و مهره ها و خانه انتخاب شده را بنویسیم.

وقتی که چیزی در Graphics رسم می کنیم (البته خارج از رویداد OnPaint) باید به پنجره پیغام دهیم که صفحه را بازسازی کند. برای این کار از متد Invalidate کنترل مورد نظر استفاده می کنیم.

اعداد تصادفی

برای تولید اعداد تصادفی از کلاس System.Random استفاده می کنیم. این کلاس متدی به نام Next دارد که با فراخوانی این متد باعث تولید یک عدد تصادفی می شود. فراخوانی این متد به شکل زیر:

```
row = rm.Next(10);
```

باعث تولید یک عدد تصادفی بین صفر تا 10 می شود (rm نام شی ای از کلاس Random می باشد).

توابع ریاضی

برای استفاده از توابع ریاضی باید از کلاس System.Math استفاده کرد. اکثر توابع این کلاس بصورت Static تعریف شده اند و بنابراین برای استفاده از آنها نیازی به تعریف شی از کلاس Math نمی باشد. مثال زیر قدر مطلق یک عدد را محاسبه می کند:

```
System.Math.Abs(_vDiff)
```

توضیح سورس کد

قبل از اینکه به سراغ توضیح سورس برویم این نکته را ذکر کنم که بدلیل حفظ جنبه آموزشی سورس، در این سورس همه کلاسها و فضاهای نامی بصورت کامل ذکر شده اند. شما می توانید در سورسهای خودتان فضاهای نامی مورد نیاز را با استفاده از using در ابتدای کد ذکر کنید تا مجبور به استفاده از عبارات و نامهای طولانی آنها نباشید. بدین شکل:

```
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
using System.Drawing.Printing;
using System.Drawing.Text;
```

در ابتدا آرایه arrBoard را تعریف کرده ایم که در حقیقت نگهدارنده اطلاعات صفحه بازی می باشد. هر تغییری در صفحه بازی در این آرایه منعکس می شود و بازسازی صفحه از روی این آرایه صورت می گیرد.

تابع FillBoard در ابتدای بازی آرایه را پر می کند. (در این آرایه عدد صفر نشاندهنده خالی بودن خانه و 1 نشاندهنده پر بودن خانه می باشد). سپس دو عدد تصادفی بین 0 تا 10 را به عنوان شماره ردیف و ستون خانه ای که باید در ابتدای بازی خالی باشد را بدست می آورد. دقت کنید که برای راندمان بهتر ابتدا همه خانه ها را با مقدار 1 پر می کنیم و سپس خانه بدست آمده به صورت تصادفی را با 0 مقداردهی می کنیم (تصور کنید اگر درون حلقه های for از دستور if استفاده می کردیم، 100 بار این دستور اجرا می شد!).

تابع DrawBoard کار رسم صفحه را از روی اطلاعات موجود در آرایه انجام می دهد. این تابع را در رویداد OnPaint فراخوانی می کنیم تا هر زمان که پنجره ما نیاز به رسم مجدد داشت، صفحه بازی نیز رسم شود.

در رویداد MouseUp فرم در صورتی که کلید سمت چپ ماوس در محدوده صفحه بازی کلیک شده باشد، شماره ردیف و ستون خانه کلیک شده را محاسبه می کنیم و سپس دور آن را یک مربع سفید برای نشان دادن خانه انتخاب شده رسم می کنیم (کار رسم مربع سفید در DrawSelect انجام و این تابع در رویداد Paint فراخوانی می شود). فراموش نکنید که بعد از رسم باید جهت بازسازی صفحه، به ویندوز اطلاع دهیم و این کار را با استفاده از متد Invalidate انجام می دهیم البته از آنجا که فقط قسمت کوچکی از صفحه تغییر کرده است و نیازی به بازسازی کل آن نیست مختصات قسمت تغییر داده شده را با استفاده از ساختمان داده Rectangle برای متد Invalidate تعیین می کنیم. با این کار سرعت بازسازی زیاد شده و هنگام بازسازی، فرم ما چشمک (blink) نمی زند.

تابع DrawSelected فقط وظیفه رسم مربع به دور خانه انتخاب شده را بر عهده دارد.
در رویداد MouseDown عملیات لازم برای بررسی شرایط حرکت انجام شده و در صورتی که همه شرایط برقرار باشد مطابق با فرمول بازی مهره انتخاب شده جای خود را عوض کرده و سپس مهره بعدی حذف می گردد. در صورت حرکت مهره ها سه خانه باید بازسازی شوند که این کار را با استفاده از متد Invalidate انجام می دهیم.

سورس کامل بازی به همراه این مقاله ارائه شده است که برای باز کردن آن نیاز به Visual Studio 2005 دارید. سورس مربوط به قسمتهایی که توصیح داده شد در ادامه نوشته شده است تا کسانی که فعلا دسترسی به VS 2005 ندارند بتوانند از آن استفاده کنند.

سورس بازی:

```
namespace JumpoGameTest
{
    public partial class MainForm : Form
    {
        private int[,] arrBoard;
        private int CurrentRow = 0;
        private int CurrentCol = 0;
        private int OldRow = 0;
        private int OldCol = 0;
        private bool dblClickState = false;

        public MainForm()
        {
            InitializeComponent();
            arrBoard= new int[10, 10];
            FillBoard();
        }

        private void FillBoard()
        {
            int row = 0;
            int col = 0;
            System.Random rm = new System.Random();
            row = rm.Next(10);
            col = rm.Next(10);
            CurrentRow = row;
            CurrentCol = col;

            arrBoard[row, col] = 0;
            for (int i = 0; i < 10; i++)
                for (int j = 0; j < 10; j++)
                    arrBoard[i, j] = 1;
            arrBoard[row, col] = 0;
        }

        private void DrawBoard(Graphics gr)
        {
            System.Drawing.Point BasePoint = new System.Drawing.Point(10, 10);
            System.Drawing.Pen LinePen = new System.Drawing.Pen(Color.Black, 2);
            System.Drawing.SolidBrush EllipseBrush = new
                    System.Drawing.SolidBrush(Color.Aquamarine);

            //Draw Table
            for (int i = BasePoint.X; i < (10 * 31 + BasePoint.X); i += 30)
                gr.DrawLine(LinePen, BasePoint.X, i, 10 * 30 + BasePoint.Y, i);
            for (int i = BasePoint.Y; i < (10 * 31 + BasePoint.X); i += 30)
                gr.DrawLine(LinePen, i, BasePoint.Y, i, 10 * 30 + BasePoint.X);
            //Draw Circles
            for (int i = 0; i < 10; i++)
                for (int j = 0; j < 10; j++)
                    if (arrBoard[i, j] == 1)
                        gr.FillEllipse(EllipseBrush,
                                 new System.Drawing.Rectangle(i * 30 + 12,
                                                              j * 30 + 12, 24, 24)
                                        );
        }

        private void MainForm_Paint(object sender, PaintEventArgs e)
```

```
        {
            DrawBoard(e.Graphics);
            DrawSelected(e.Graphics);
        }

        private void MainForm_MouseUp(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
                if (e.X > 10 && e.X < (10 * 30 + 10) &&
                                e.Y > 10 && e.Y < (10 * 30 + 10))
                {
                    OldCol = CurrentCol;
                    CurrentCol = (e.Y-10)/30;
                    OldRow = CurrentRow;
                    CurrentRow = (e.X-10)/30;
                    this.Invalidate(new System.Drawing.Rectangle(CurrentRow * 30 + 8,
                                                    CurrentCol * 30 + 8, 35, 35)
                                    );
                    this.Invalidate(new System.Drawing.Rectangle(OldRow * 30 + 8,
                                                    OldCol * 30 + 8, 35, 35)
                                    );
                }
        }

        private void DrawSelected(Graphics gr)
        {
            System.Drawing.Pen RectPen = new System.Drawing.Pen(Color.White, 2);
            System.Drawing.Pen ClearPen = new System.Drawing.Pen(Color.Black, 2);
            gr.DrawRectangle(ClearPen, new System.Drawing.Rectangle(OldRow * 30 + 10,
                                            OldCol * 30 + 10, 30, 30)
                                );
            gr.DrawRectangle(RectPen,
                                new System.Drawing.Rectangle(CurrentRow * 30 + 10,
                                                CurrentCol * 30 + 10, 30, 30)
                                );
        }

        private void MainForm_MouseDown(object sender, MouseEventArgs e)
        {
            dblClickState = !dblClickState;
            if (!dblClickState)
            {
                if (e.X > 10 && e.X < (10 * 30 + 10) &&
                                e.Y > 10 && e.Y < (10 * 30 + 10))
                {
                    int _col = 0;
                    int _row = 0;
                    int _vDiff = 0;
                    int _hDiff = 0;

                    _col = (e.Y - 10) / 30;
                    _row = (e.X - 10) / 30;
                    if (_col == CurrentCol + 2)
                        _vDiff = 1;
                    if (_col == CurrentCol - 2)
                        _vDiff = -1;
                    if (_row == CurrentRow + 2)
                        _hDiff = 1;
                    if (_row == CurrentRow - 2)
                        _hDiff = -1;
                    if (System.Math.Abs(_vDiff) == System.Math.Abs(_hDiff))
                        return;

                    if (arrBoard[_row, _col] == 0 &&
                            arrBoard[CurrentRow + _hDiff, CurrentCol + _vDiff]==1)
                    {
                        arrBoard[_row, _col] = arrBoard[CurrentRow, CurrentCol];
                        arrBoard[CurrentRow + _hDiff, CurrentCol + _vDiff] = 0;
                        arrBoard[CurrentRow , CurrentCol] = 0;
                        this.Invalidate(
                                new System.Drawing.Rectangle(CurrentRow * 30 + 8,
```

```
                                        CurrentCol * 30 + 8, 35, 35)
                                    );
                            this.Invalidate(
                                    new System.Drawing.Rectangle(_row * 30 + 8,
                                                    _col * 30 + 8, 35, 35)
                                    );
                            this.Invalidate(
                                    new System.Drawing.Rectangle((CurrentRow+_hDiff)*30+ 8,
                                        (CurrentCol + _vDiff) * 30 + 8, 35, 35)
                                    );
                        }
                    }
                }
            }
        }
    }
```

پیشنهاد

شما می توانید با توسعه این بازی، مهارت خود را در GDI+ افزایش دهید. برای توسعه بازی Jumpo اضافه کردن این قسمتها را پیشنهاد می دهیم:

1- Save و Load بازی
2- امکان تعریف تعداد سطرها و ستونها به صورت پارامتریک
3- اضافه کردن سیستم امتیاز در بازی
4- تعیین محدوده زمانی برای اتمام بازی
5- اضافه کردن امکان تشخیص GameOver در بازی

نویسنده: محمد صافدل
Email: msafdel@gmail.com

لینک فروشگاه سایت طراحان ایرانی: http://shop.persian-designers.com

سایت طراحان ایرانی با هدف آموزش ساخت بازیهای کامپیوتری به زبان فارسی طراحی شده است و تا کنون مقالات متعددی در زمینه های مختلف برنامه نویسی و ساخت بازی در آن قرار گرفته است .مدیریت سایت از تمامی عزیزان علاقمند به بازی های کامپیوتری ، برنامه نویسان ، طراحان و سایر کسانی که به نحوی با بازی ها در ارتباطند ، دعوت به همکاری به عمل می آورد تا بدینوسیله یک پایگاه علمی و موثق در زمینه صنعت ساخت بازیهای کامپیوتری در ایران ایجاد گردد.