Hatch Brush Menu

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;

class HatchBrushMenu: Form
{
HatchStyleMenuItem hsmiChecked;
const int iMin = 0, iMax = 52;
public static void Main()
{
Application.Run(new HatchBrushMenu());
}
public HatchBrushMenu()
{
ResizeRedraw = true;

Menu = new MainMenu();
Menu.MenuItems.Add(“&Hatch-Style”);

for (HatchStyle hs = (HatchStyle)iMin; hs <= (HatchStyle)iMax;hs++) { HatchStyleMenuItem hsmi = new HatchStyleMenuItem(); hsmi.HatchStyle = hs; hsmi.Click += new EventHandler(MenuHatchStyleOnClick); if ((int)hs % 8 == 0) hsmi.BarBreak = true; Menu.MenuItems[0].MenuItems.Add(hsmi); } hsmiChecked = (HatchStyleMenuItem) Menu.MenuItems[0].MenuItems[0]; hsmiChecked.Checked = true; } void MenuHatchStyleOnClick(object obj, EventArgs ea) { hsmiChecked.Checked = false; hsmiChecked = (HatchStyleMenuItem) obj; hsmiChecked.Checked = true; Invalidate(); } protected override void OnPaint(PaintEventArgs pea) { Graphics grfx = pea.Graphics; HatchBrush hbrush = new HatchBrush(hsmiChecked.HatchStyle, Color.White, Color.Black); grfx.FillEllipse(hbrush, ClientRectangle); } } class
```

HatchStyleMenuItem: MenuItem { const int cxImage = 32, cyImage = 32, iMargin = 2; readonly int cxCheck, cyCheck; public HatchStyle HatchStyle; public HatchStyleMenuItem() { OwnerDraw = true; cxCheck = SystemInformation.MenuCheckSize.Width; cyCheck = SystemInformation.MenuCheckSize.Height; } protected override void OnMeasureItem(MeasureItemEventArgs miea) { miea.ItemWidth = 2 * cxImage + 3 * iMargin - cxCheck; miea.ItemHeight = cyImage + 2 * iMargin; } protected override void OnDrawItem(DrawItemEventArgs diea) { diea.DrawBackground(); if ((diea.State & DrawItemState.Checked) != 0) { ControlPaint.DrawMenuGlyph(diea.Graphics, diea.Bounds.Location.X + iMargin, diea.Bounds.Location.Y + iMargin, cxImage, cyImage, MenuGlyph.Checkmark); } HatchBrush hbrush = new HatchBrush(HatchStyle, Color.White, Color.Black); diea.Graphics.FillRectangle(hbrush, diea.Bounds.X + 2 * iMargin + cxImage, diea.Bounds.Y + iMargin, cxImage, cyImage); } } [/csharp]