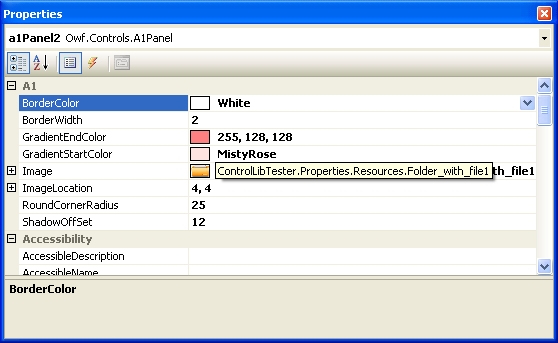
**Introduction**

This article will explain how we can create cool looking gradient panel with round corner in .net winforms using GDI+. This panel supports various customizable properties. Below are some cool samples.  
  


**Features**

You can customize follwing things in this panel

* Gradient Start Color
* Gradient End Color
* Border Color
* Border Width
* Shadow Offset
* Round corner radius
* Panel Image
* Panel Image location



**Source Code** [(Download here)](http://www.openwinforms.com/Cool_Gradient_Panel.zip)

using System;

using System.ComponentModel;

using System.Collections.Generic;

using System.Diagnostics;

using System.Text;

using System.Windows.Forms;

using System.Drawing;

using System.Drawing.Drawing2D;

namespace Owf.Controls

{

// A1Panel class

public partial class A1Panel : Panel

{

int \_borderWidth = 1;

[Browsable(true), Category(A1PanelGlobals.A1Category)]

[DefaultValue(1)]

public int BorderWidth

{

get { return \_borderWidth; }

set { \_borderWidth = value; Invalidate(); }

}

int \_shadowOffSet = 5;

[Browsable(true), Category(A1PanelGlobals.A1Category)]

[DefaultValue(5)]

public int ShadowOffSet

{

get

{

return \_shadowOffSet;

}

set { \_shadowOffSet = Math.Abs(value); Invalidate(); }

}

int \_roundCornerRadius = 4;

[Browsable(true), Category(A1PanelGlobals.A1Category)]

[DefaultValue(4)]

public int RoundCornerRadius

{

get { return \_roundCornerRadius; }

set { \_roundCornerRadius = Math.Abs(value); Invalidate(); }

}

Image \_image;

[Browsable(true), Category(A1PanelGlobals.A1Category)]

public Image Image

{

get { return \_image; }

set { \_image = value; Invalidate(); }

}

Point \_imageLocation = new Point(4, 4);

[Browsable(true), Category(A1PanelGlobals.A1Category)]

[DefaultValue("4,4")]

public Point ImageLocation

{

get { return \_imageLocation; }

set { \_imageLocation = value; Invalidate(); }

}

Color \_borderColor = Color.Gray;

[Browsable(true), Category(A1PanelGlobals.A1Category)]

[DefaultValue("Color.Gray")]

public Color BorderColor

{

get { return \_borderColor; }

set { \_borderColor = value; Invalidate(); }

}

Color \_gradientStartColor = Color.White;

[Browsable(true), Category(A1PanelGlobals.A1Category)]

[DefaultValue("Color.White")]

public Color GradientStartColor

{

get { return \_gradientStartColor; }

set { \_gradientStartColor = value; Invalidate(); }

}

Color \_gradientEndColor = Color.Gray;

[Browsable(true), Category(A1PanelGlobals.A1Category)]

[DefaultValue("Color.Gray")]

public Color GradientEndColor

{

get { return \_gradientEndColor; }

set { \_gradientEndColor = value; Invalidate(); }

}

public A1Panel()

{

this.SetStyle(ControlStyles.DoubleBuffer, true);

this.SetStyle(ControlStyles.AllPaintingInWmPaint, true);

this.SetStyle(ControlStyles.ResizeRedraw, true);

this.SetStyle(ControlStyles.UserPaint, true);

this.SetStyle(ControlStyles.SupportsTransparentBackColor, true);

InitializeComponent();

}

protected override void OnPaintBackground(PaintEventArgs e)

{

base.OnPaintBackground(e);

int tmpShadowOffSet = Math.Min(Math.Min(\_shadowOffSet, this.Width - 2), this.Height - 2);

int tmpSoundCornerRadius = Math.Min(Math.Min(\_roundCornerRadius, this.Width - 2), this.Height - 2);

if (this.Width > 1 && this.Height > 1)

{

e.Graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

Rectangle rect = new Rectangle(0, 0, this.Width - tmpShadowOffSet - 1,

this.Height - tmpShadowOffSet - 1);

Rectangle rectShadow = new Rectangle(tmpShadowOffSet, tmpShadowOffSet,

this.Width - tmpShadowOffSet - 1, this.Height - tmpShadowOffSet - 1);

GraphicsPath graphPathShadow = A1PanelGraphics.GetRoundPath(rectShadow, tmpSoundCornerRadius);

GraphicsPath graphPath = A1PanelGraphics.GetRoundPath(rect, tmpSoundCornerRadius);

if (tmpSoundCornerRadius > 0)

{

using (PathGradientBrush gBrush = new PathGradientBrush(graphPathShadow))

{

gBrush.WrapMode = WrapMode.Clamp;

ColorBlend colorBlend = new ColorBlend(3);

colorBlend.Colors = new Color[]{Color.Transparent,

Color.FromArgb(180, Color.DimGray),

Color.FromArgb(180, Color.DimGray)};

colorBlend.Positions = new float[] { 0f, .1f, 1f };

gBrush.InterpolationColors = colorBlend;

e.Graphics.FillPath(gBrush, graphPathShadow);

}

}

// Draw backgroup

LinearGradientBrush brush = new LinearGradientBrush(rect,

this.\_gradientStartColor,

this.\_gradientEndColor,

LinearGradientMode.BackwardDiagonal);

e.Graphics.FillPath(brush, graphPath);

e.Graphics.DrawPath(new Pen(Color.FromArgb(180, this.\_borderColor), \_borderWidth), graphPath);

// Draw Image

if (\_image != null)

{

e.Graphics.DrawImageUnscaled(\_image, \_imageLocation);

}

}

}

}

// A1PanelGraphics class

internal class A1PanelGraphics

{

public static GraphicsPath GetRoundPath(Rectangle r, int depth)

{

GraphicsPath graphPath = new GraphicsPath();

graphPath.AddArc(r.X, r.Y, depth, depth, 180, 90);

graphPath.AddArc(r.X + r.Width - depth, r.Y, depth, depth, 270, 90);

graphPath.AddArc(r.X + r.Width - depth, r.Y + r.Height - depth, depth, depth, 0, 90);

graphPath.AddArc(r.X, r.Y + r.Height - depth, depth, depth, 90, 90);

graphPath.AddLine(r.X, r.Y + r.Height - depth, r.X, r.Y + depth / 2);

return graphPath;

}

}

// A1PanelGlobals class

internal class A1PanelGlobals

{

public const string A1Category = "A1";

}

}

**Some Other Cool samples**

