

```

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using System.Management;
using Ripper;
using System.Drawing.Drawing2D;
using Yeti.MMedia;
using Yeti.MMedia.Mp3;
using WaveLib;

namespace SB54_CSAI.CDRip
{
    #region uctCDRipper CLASS
    /// <summary>
    /// uctCDRipper is a sub class of a <see cref="System.Windows.Forms.UserControl">User
    Control </see>
    /// that provides an Audio CD ripper windows form control. The ripper control enables
    /// the user to rip a single file or an entire audio CD. In both cases the user
    /// must specify the type of rip operation, WAV or MP3 rip and they must also specify
    the
    /// output path for the ripped file(s) to be stored.
    /// <list type="bullet">
    /// <item>
    /// <description>This code makes use of a free audio CD ripper library, for C#, that I
    found
    /// on the internet within a forum at www.c-sharpcorner.com, by a one Idael Cardoso
    /// (yetiicb@hotmail.com), I have communicated with Idael directly about the usage of
    his
    /// CD ripper library, and Idael has stated that he is perfectly happy with me using his
    /// ripper library in my project. In fact I think he was very happy to hear that
    someone
    /// found a use for his ripper library. Idael will receive FULL acknowledgments for the
    use
    /// of his Dll in my ReMP3 project.
    /// <br></br>
    /// <br></br>
    /// <b>NOTE : </b>That even with the use of Idaels CD ripper library there
    /// are still some 1200 or so lines of code to create a suitable wrapper/GUI interface,
    so there
    /// was still substantial ammounts of effort required to get the library to work the way
    I wanted
    /// it to work.
    /// <br></br>This uctCDRipper user control represents the wrapper around Idaels CD
    ripper
    /// library.<br></br>
    /// </description>
    /// </item>
    /// </list>
    /// </summary>
    public class uctCDRipper : System.Windows.Forms.UserControl
    {
        #region Instance fields
        //Instance fields
        private System.ComponentModel.IContainer components;
        private System.Windows.Forms.Panel oPnlFill;
        private System.Windows.Forms.Panel oPnlBottom;
        private System.Windows.Forms.Label oLblTask;
        private System.Windows.Forms.Label olblStatus;
        private System.Windows.Forms.Panel oPnlTop;
        private System.Windows.Forms.Button oBtnSaveFile;
        private System.Windows.Forms.Button oBtnLoadCD;
        private System.Windows.Forms.Button oBtnEjectCD;
        private System.Windows.Forms.Button oBtnSaveAll;
        private System.Windows.Forms.Button oBtnCancelRip;
        private System.Windows.Forms.Panel oPnlLeft;
        private System.Windows.Forms.Panel oPnlMain;
        private System.Windows.Forms.ProgressBar oProg;
        private System.Windows.Forms.ListView oLvTracks;
        private System.Windows.Forms.ColumnHeader oLvColTrack;
    }
}

```

```

private System.Windows.Forms.ColumnHeader oLvColSize;
private System.Windows.Forms.ColumnHeader oLvColType;
private System.Windows.Forms.Button oBtnOpenCD;
private System.Windows.Forms.ComboBox oCmbDrives;
private System.Windows.Forms.Label oLblDrives;
private System.Windows.Forms.Panel oPnlRipper;
private System.Windows.Forms.ImageList oImgListCD;
private System.Windows.Forms.Label oLblCDTOC;
private System.Windows.Forms.Label oLblProgress;
private System.Windows.Forms.Label oLblTrackSmall;
private System.Windows.Forms.Label oLblTitle1;
private System.Windows.Forms.Label oLblCDCtrls;
private System.Windows.Forms.Label oLblRipCtrls;
private System.Windows.Forms.SaveFileDialog oSaveFileDialog;
private System.Windows.Forms.ToolTip otoolTip;
private System.Windows.Forms.ImageList oImgListSmall;
private System.Windows.Forms.FolderBrowserDialog oFolderBrowserDialog;
private AudioWriter oWavWriter = null;
private Ripper.CDDrive oCDDrive;
private bool ripping = false;
private bool cancelRipping = false;
/// <summary>
/// ripType enumeration : 0 = Rip to WAV format, 1 = Rip to MP3 format
/// </summary>
public enum ripType : int
{
    RIP_WAV = 0,
    RIP_MP3 = 1
}
/// <summary>
/// the current rip operation type, WAV or MP3
/// </summary>
public static ripType RipOperation;

#endregion
#region Public Constructor
/// <summary>
/// Constructor, simply initalises all GUI components
/// </summary>
public uctCDRipper()
{
    // This call is required by the Windows.Forms Form Designer.
    InitializeComponent();
}
#endregion
#region Public Methods
/// <summary>
/// Clean up any resources being used.
/// </summary>
/// <param name="disposing">true indicates the object should be disposed</param>
protected override void Dispose( bool disposing )
{
    //if being asked to dispose
    if( disposing )
    {
        //clean up the held resources
        if (components != null) { components.Dispose(); }
        if (oCDDrive != null) { oCDDrive.Dispose(); }
    }
    //call base class Dispose
    base.Dispose( disposing );
}
#endregion
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.components = new System.ComponentModel.Container();
    System.Resources.ResourceManager resources = new System.Resources.

```

```

ResourceManager(typeof(uctCDRipper));
this.oSaveFileDialog = new System.Windows.Forms.SaveFileDialog();
this.oToolTip = new System.Windows.Forms.ToolTip(this.components);
this.oBtnLoadCD = new System.Windows.Forms.Button();
this.oBtnEjectCD = new System.Windows.Forms.Button();
this.oLvTracks = new System.Windows.Forms.ListView();
this.oLvColTrack = new System.Windows.Forms.ColumnHeader();
this.oLvColSize = new System.Windows.Forms.ColumnHeader();
this.oLvColType = new System.Windows.Forms.ColumnHeader();
this.oImgListSmall = new System.Windows.Forms.ImageList(this.components);
this.oBtnOpenCD = new System.Windows.Forms.Button();
this.oCmbDrives = new System.Windows.Forms.ComboBox();
this.oBtnSaveFile = new System.Windows.Forms.Button();
this.oBtnSaveAll = new System.Windows.Forms.Button();
this.oBtnCancelRip = new System.Windows.Forms.Button();
this.oFolderBrowserDialog = new System.Windows.Forms.FolderBrowserDialog();
this.oPnlFill = new System.Windows.Forms.Panel();
this.oPnlMain = new System.Windows.Forms.Panel();
this.oPnlRipper = new System.Windows.Forms.Panel();
this.oLblProgress = new System.Windows.Forms.Label();
this.oLblCDTOC = new System.Windows.Forms.Label();
this.oLblDrives = new System.Windows.Forms.Label();
this.oProg = new System.Windows.Forms.ProgressBar();
this.oPnlLeft = new System.Windows.Forms.Panel();
this.oLblRipCtrls = new System.Windows.Forms.Label();
this.oLblCDCtrls = new System.Windows.Forms.Label();
this.oPnlTop = new System.Windows.Forms.Panel();
this.oLblTrackSmall = new System.Windows.Forms.Label();
this.oLblTitle1 = new System.Windows.Forms.Label();
this.oPnlBottom = new System.Windows.Forms.Panel();
this.oLblStatus = new System.Windows.Forms.Label();
this.oLblTask = new System.Windows.Forms.Label();
this.oImgListCD = new System.Windows.Forms.ImageList(this.components);
this.oPnlFill.SuspendLayout();
this.oPnlMain.SuspendLayout();
this.oPnlRipper.SuspendLayout();
this.oPnlLeft.SuspendLayout();
this.oPnlTop.SuspendLayout();
this.oPnlBottom.SuspendLayout();
this.SuspendLayout();
//
// oSaveFileDialog
//
this.oSaveFileDialog.DefaultExt = "wav";
this.oSaveFileDialog.Title = "Save track to:";
//
// oBtnLoadCD
//
this.oBtnLoadCD.BackColor = System.Drawing.Color.White;
this.oBtnLoadCD.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.oBtnLoadCD.Image = ((System.Drawing.Image)(resources.GetObject("oBtnLoadCD.
Image")));
this.oBtnLoadCD.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.oBtnLoadCD.Location = new System.Drawing.Point(12, 80);
this.oBtnLoadCD.Name = "oBtnLoadCD";
this.oBtnLoadCD.Size = new System.Drawing.Size(127, 36);
this.oBtnLoadCD.TabIndex = 10;
this.oBtnLoadCD.Text = "Load CD";
this.oToolTip.SetToolTip(this.oBtnLoadCD, "Load (Close) the CD drive");
this.oBtnLoadCD.Click += new System.EventHandler(this.oBtnLoadCD_Click);
//
// oBtnEjectCD
//
this.oBtnEjectCD.BackColor = System.Drawing.Color.White;
this.oBtnEjectCD.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.oBtnEjectCD.Image = ((System.Drawing.Image)(resources.GetObject(
"oBtnEjectCD.Image")));
this.oBtnEjectCD.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.oBtnEjectCD.Location = new System.Drawing.Point(12, 35);
this.oBtnEjectCD.Name = "oBtnEjectCD";
this.oBtnEjectCD.Size = new System.Drawing.Size(127, 36);
this.oBtnEjectCD.TabIndex = 9;
this.oBtnEjectCD.Text = "Eject";

```

```

        this.oToolTip.SetToolTip(this.oBtnEjectCD, "Eject (Open) the CD drive");
        this.oBtnEjectCD.Click += new System.EventHandler(this.oBtnEjectCD_Click);
        //
        // oLvTracks
        //
        this.oLvTracks.Columns.AddRange(new System.Windows.Forms.ColumnHeader[] {
this.oLvColTrack,
this.oLvColSize,
this.oLvColType});
        this.oLvTracks.FullRowSelect = true;
        this.oLvTracks.GridLines = true;
        this.oLvTracks.Location = new System.Drawing.Point(30, 86);
        this.oLvTracks.Name = "oLvTracks";
        this.oLvTracks.Size = new System.Drawing.Size(306, 201);
        this.oLvTracks.SmallImageList = this.oImgListSmall;
        this.oLvTracks.TabIndex = 18;
        this.oToolTip.SetToolTip(this.oLvTracks, "Select the track that you want to save");
    };

    this.oLvTracks.View = System.Windows.Forms.View.Details;
    this.oLvTracks.EnabledChanged += new System.EventHandler(this.
oLvTracks_EnabledChanged);
    this.oLvTracks.SelectedIndexChanged += new System.EventHandler(this.
oLvTracks_SelectedIndexChanged);
    //
    // oLvColTrack
    //
    this.oLvColTrack.Text = "Track";
    this.oLvColTrack.Width = 62;
    //
    // oLvColSize
    //
    this.oLvColSize.Text = "Size (bytes)";
    this.oLvColSize.Width = 105;
    //
    // oLvColType
    //
    this.oLvColType.Text = "Type";
    this.oLvColType.Width = 115;
    //
    // oImgListSmall
    //
    this.oImgListSmall.ColorDepth = System.Windows.Forms.ColorDepth.Depth24Bit;
    this.oImgListSmall.ImageSize = new System.Drawing.Size(17, 17);
    this.oImgListSmall.ImageStream = ((System.Windows.Forms.ImageListStreamer)
(resources.GetObject("oImgListSmall.ImageStream")));
    this.oImgListSmall.TransparentColor = System.Drawing.Color.Transparent;
    //
    // oBtnOpenCD
    //
    this.oBtnOpenCD.BackColor = System.Drawing.Color.Transparent;
    this.oBtnOpenCD.Enabled = false;
    this.oBtnOpenCD.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
    this.oBtnOpenCD.Image = ((System.Drawing.Image)(resources.GetObject("oBtnOpenCD.
Image")));
    this.oBtnOpenCD.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
    this.oBtnOpenCD.Location = new System.Drawing.Point(240, 20);
    this.oBtnOpenCD.Name = "oBtnOpenCD";
    this.oBtnOpenCD.Size = new System.Drawing.Size(95, 36);
    this.oBtnOpenCD.TabIndex = 22;
    this.oBtnOpenCD.Text = "    Open";
    this.oToolTip.SetToolTip(this.oBtnOpenCD, "Open/Close the CD drive");
    this.oBtnOpenCD.Click += new System.EventHandler(this.oBtnOpenCD_Click);
    //
    // oCmbDrives
    //
    this.oCmbDrives.DrawMode = System.Windows.Forms.DrawMode.OwnerDrawFixed;
    this.oCmbDrives.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
    this.oCmbDrives.ItemHeight = 30;
    this.oCmbDrives.Location = new System.Drawing.Point(31, 20);
    this.oCmbDrives.Name = "oCmbDrives";
    this.oCmbDrives.Size = new System.Drawing.Size(195, 36);
    this.oCmbDrives.TabIndex = 21;
    this.oToolTip.SetToolTip(this.oCmbDrives, "Select the source CD drive to

```

```

retrieve data from");
    this.oCmbDrives.SelectedIndexChanged += new System.EventHandler(this.
oCmbDrives_SelectedIndexChanged);
    this.oCmbDrives.DrawItem += new System.Windows.Forms.DrawItemEventHandler(this.
oCmbDrives_DrawItem);
    //
    // oBtnSaveFile
    //
    this.oBtnSaveFile.BackColor = System.Drawing.Color.White;
    this.oBtnSaveFile.Enabled = false;
    this.oBtnSaveFile.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
    this.oBtnSaveFile.Image = ((System.Drawing.Image)(resources.GetObject(
"oBtnSaveFile.Image")));
    this.oBtnSaveFile.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
    this.oBtnSaveFile.Location = new System.Drawing.Point(12, 220);
    this.oBtnSaveFile.Name = "oBtnSaveFile";
    this.oBtnSaveFile.Size = new System.Drawing.Size(127, 36);
    this.oBtnSaveFile.TabIndex = 12;
    this.oBtnSaveFile.Text = "    Rip File";
    this.oToolTip.SetToolTip(this.oBtnSaveFile, "Rip a single file");
    this.oBtnSaveFile.Click += new System.EventHandler(this.oBtnSaveFile_Click);
    //
    // oBtnSaveAll
    //
    this.oBtnSaveAll.BackColor = System.Drawing.Color.White;
    this.oBtnSaveAll.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
    this.oBtnSaveAll.Image = ((System.Drawing.Image)(resources.GetObject(
"oBtnSaveAll.Image")));
    this.oBtnSaveAll.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
    this.oBtnSaveAll.Location = new System.Drawing.Point(12, 265);
    this.oBtnSaveAll.Name = "oBtnSaveAll";
    this.oBtnSaveAll.Size = new System.Drawing.Size(127, 36);
    this.oBtnSaveAll.TabIndex = 13;
    this.oBtnSaveAll.Text = "    Rip CD";
    this.oToolTip.SetToolTip(this.oBtnSaveAll, "Rip the entire CD");
    this.oBtnSaveAll.Click += new System.EventHandler(this.oBtnSaveAll_Click);
    //
    // oBtnCancelRip
    //
    this.oBtnCancelRip.BackColor = System.Drawing.Color.White;
    this.oBtnCancelRip.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
    this.oBtnCancelRip.Image = ((System.Drawing.Image)(resources.GetObject(
"oBtnCancelRip.Image")));
    this.oBtnCancelRip.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
    this.oBtnCancelRip.Location = new System.Drawing.Point(12, 175);
    this.oBtnCancelRip.Name = "oBtnCancelRip";
    this.oBtnCancelRip.Size = new System.Drawing.Size(127, 36);
    this.oBtnCancelRip.TabIndex = 11;
    this.oBtnCancelRip.Text = "    Cancel Rip";
    this.oToolTip.SetToolTip(this.oBtnCancelRip, "Cancel the current rip operation");
;

    this.oBtnCancelRip.Click += new System.EventHandler(this.oBtnCancelRip_Click);
    //
    // oPnlFill
    //
    this.oPnlFill.Controls.Add(this.oPnlMain);
    this.oPnlFill.Controls.Add(this.oPnlLeft);
    this.oPnlFill.Controls.Add(this.oPnlTop);
    this.oPnlFill.Dock = System.Windows.Forms.DockStyle.Fill;
    this.oPnlFill.Location = new System.Drawing.Point(0, 0);
    this.oPnlFill.Name = "oPnlFill";
    this.oPnlFill.Size = new System.Drawing.Size(1008, 587);
    this.oPnlFill.TabIndex = 6;
    this.oPnlFill.Resize += new System.EventHandler(this.oPnlFill_Resize);
    this.oPnlFill.Paint += new System.Windows.Forms.PaintEventHandler(this.
oPnlFill_Paint);
    //
    // oPnlMain
    //
    this.oPnlMain.BackColor = System.Drawing.Color.Transparent;
    this.oPnlMain.Controls.Add(this.oPnlRipper);
    this.oPnlMain.Dock = System.Windows.Forms.DockStyle.Fill;
    this.oPnlMain.Location = new System.Drawing.Point(183, 64);

```

```

this.oPnlMain.Name = "oPnlMain";
this.oPnlMain.Size = new System.Drawing.Size(825, 523);
this.oPnlMain.TabIndex = 13;
this.oPnlMain.Resize += new System.EventHandler(this.oPnlMain_Resize);
//
// oPnlRipper
//
this.oPnlRipper.BackColor = System.Drawing.Color.Transparent;
this.oPnlRipper.Controls.Add(this.oLblProgress);
this.oPnlRipper.Controls.Add(this.oLblCDTOC);
this.oPnlRipper.Controls.Add(this.oBtnOpenCD);
this.oPnlRipper.Controls.Add(this.oCmbDrives);
this.oPnlRipper.Controls.Add(this.oLblDrives);
this.oPnlRipper.Controls.Add(this.oLvTracks);
this.oPnlRipper.Controls.Add(this.oProg);
this.oPnlRipper.Location = new System.Drawing.Point(163, 15);
this.oPnlRipper.Name = "oPnlRipper";
this.oPnlRipper.Size = new System.Drawing.Size(385, 361);
this.oPnlRipper.TabIndex = 20;
//
// oLblProgress
//
this.oLblProgress.BackColor = System.Drawing.Color.Transparent;
this.oLblProgress.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.oLblProgress.Location = new System.Drawing.Point(31, 298);
this.oLblProgress.Name = "oLblProgress";
this.oLblProgress.Size = new System.Drawing.Size(189, 16);
this.oLblProgress.TabIndex = 24;
this.oLblProgress.Text = "Ripping Progress";
this.oLblProgress.Visible = false;
//
// oLblCDTOC
//
this.oLblCDTOC.BackColor = System.Drawing.Color.Transparent;
this.oLblCDTOC.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.oLblCDTOC.Location = new System.Drawing.Point(31, 67);
this.oLblCDTOC.Name = "oLblCDTOC";
this.oLblCDTOC.Size = new System.Drawing.Size(189, 16);
this.oLblCDTOC.TabIndex = 23;
this.oLblCDTOC.Text = "CD Contents";
//
// oLblDrives
//
this.oLblDrives.BackColor = System.Drawing.Color.Transparent;
this.oLblDrives.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.oLblDrives.Location = new System.Drawing.Point(31, 3);
this.oLblDrives.Name = "oLblDrives";
this.oLblDrives.Size = new System.Drawing.Size(189, 16);
this.oLblDrives.TabIndex = 20;
this.oLblDrives.Text = "Available CD Drives";
//
// oProg
//
this.oProg.Location = new System.Drawing.Point(30, 315);
this.oProg.Name = "oProg";
this.oProg.Size = new System.Drawing.Size(306, 23);
this.oProg.TabIndex = 17;
this.oProg.Visible = false;
//
// oPnlLeft
//
this.oPnlLeft.BackColor = System.Drawing.Color.Transparent;
this.oPnlLeft.Controls.Add(this.oLblRipCtrls);
this.oPnlLeft.Controls.Add(this.oLblCDCtrls);
this.oPnlLeft.Controls.Add(this.oBtnSaveFile);
this.oPnlLeft.Controls.Add(this.oBtnLoadCD);
this.oPnlLeft.Controls.Add(this.oBtnEjectCD);
this.oPnlLeft.Controls.Add(this.oBtnSaveAll);
this.oPnlLeft.Controls.Add(this.oBtnCancelRip);
this.oPnlLeft.Dock = System.Windows.Forms.DockStyle.Left;

```

```

this.oPnlLeft.Location = new System.Drawing.Point(0, 64);
this.oPnlLeft.Name = "oPnlLeft";
this.oPnlLeft.Size = new System.Drawing.Size(183, 523);
this.oPnlLeft.TabIndex = 12;
//
// oLblRipCtrls
//
this.oLblRipCtrls.BackColor = System.Drawing.Color.Transparent;
this.oLblRipCtrls.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.oLblRipCtrls.Location = new System.Drawing.Point(13, 158);
this.oLblRipCtrls.Name = "oLblRipCtrls";
this.oLblRipCtrls.Size = new System.Drawing.Size(139, 16);
this.oLblRipCtrls.TabIndex = 22;
this.oLblRipCtrls.Text = "Ripping Controls";
//
// oLblCDCtrls
//
this.oLblCDCtrls.BackColor = System.Drawing.Color.Transparent;
this.oLblCDCtrls.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.oLblCDCtrls.Location = new System.Drawing.Point(12, 18);
this.oLblCDCtrls.Name = "oLblCDCtrls";
this.oLblCDCtrls.Size = new System.Drawing.Size(139, 16);
this.oLblCDCtrls.TabIndex = 21;
this.oLblCDCtrls.Text = "CD Controls";
//
// oPnlTop
//
this.oPnlTop.BackColor = System.Drawing.Color.Transparent;
this.oPnlTop.Controls.Add(this.oLblTrackSmall);
this.oPnlTop.Controls.Add(this.oLblTitle1);
this.oPnlTop.Dock = System.Windows.Forms.DockStyle.Top;
this.oPnlTop.Location = new System.Drawing.Point(0, 0);
this.oPnlTop.Name = "oPnlTop";
this.oPnlTop.Size = new System.Drawing.Size(1008, 64);
this.oPnlTop.TabIndex = 10;
//
// oLblTrackSmall
//
this.oLblTrackSmall.BackColor = System.Drawing.Color.Transparent;
this.oLblTrackSmall.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)
(0)));
this.oLblTrackSmall.Location = new System.Drawing.Point(7, 28);
this.oLblTrackSmall.Name = "oLblTrackSmall";
this.oLblTrackSmall.Size = new System.Drawing.Size(495, 32);
this.oLblTrackSmall.TabIndex = 17;
this.oLblTrackSmall.Text = "Ripped files may be saved as WAV (*.wav) or MP3 (*.
mp3) files, in a location of y" +
    "our choosing";
//
// oLblTitle1
//
this.oLblTitle1.BackColor = System.Drawing.Color.Transparent;
this.oLblTitle1.Font = new System.Drawing.Font("Microsoft Sans Serif", 11.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.oLblTitle1.Location = new System.Drawing.Point(7, 4);
this.oLblTitle1.Name = "oLblTitle1";
this.oLblTitle1.Size = new System.Drawing.Size(374, 23);
this.oLblTitle1.TabIndex = 16;
this.oLblTitle1.Text = "Rip a single file or rip an entire CD";
//
// oPnlBottom
//
this.oPnlBottom.BackColor = System.Drawing.Color.White;
this.oPnlBottom.Controls.Add(this.oLblStatus);
this.oPnlBottom.Controls.Add(this.oLblTask);
this.oPnlBottom.Dock = System.Windows.Forms.DockStyle.Bottom;
this.oPnlBottom.ForeColor = System.Drawing.Color.White;
this.oPnlBottom.Location = new System.Drawing.Point(0, 563);
this.oPnlBottom.Name = "oPnlBottom";
this.oPnlBottom.Size = new System.Drawing.Size(1008, 24);

```

```

        this.oPnlBottom.TabIndex = 27;
        this.oPnlBottom.Paint += new System.Windows.Forms.PaintEventHandler(this.
oPnlBottom_Paint);
        //
        // olblStatus
        //
        this.olblStatus.BackColor = System.Drawing.Color.Transparent;
        this.olblStatus.Font = new System.Drawing.Font("Tahoma", 8.25F, System.Drawing.
FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
        this.olblStatus.ForeColor = System.Drawing.Color.White;
        this.olblStatus.Location = new System.Drawing.Point(14, 3);
        this.olblStatus.Name = "olblStatus";
        this.olblStatus.Size = new System.Drawing.Size(592, 18);
        this.olblStatus.TabIndex = 28;
        //
        // oLblTask
        //
        this.oLblTask.BackColor = System.Drawing.Color.Transparent;
        this.oLblTask.Font = new System.Drawing.Font("Tahoma", 8.25F, System.Drawing.
FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
        this.oLblTask.ForeColor = System.Drawing.Color.White;
        this.oLblTask.Location = new System.Drawing.Point(8, 3);
        this.oLblTask.Name = "oLblTask";
        this.oLblTask.Size = new System.Drawing.Size(592, 18);
        this.oLblTask.TabIndex = 27;
        //
        // oImgListCD
        //
        this.oImgListCD.ColorDepth = System.Windows.Forms.ColorDepth.Depth24Bit;
        this.oImgListCD.ImageSize = new System.Drawing.Size(30, 30);
        this.oImgListCD.ImageStream = ((System.Windows.Forms.ImageListStreamer)
(resources.GetObject("oImgListCD.ImageStream")));
        this.oImgListCD.TransparentColor = System.Drawing.Color.Transparent;
        //
        // uctCDRipper
        //
        this.BackColor = System.Drawing.SystemColors.Control;
        this.Controls.Add(this.oPnlBottom);
        this.Controls.Add(this.oPnlFill);
        this.Name = "uctCDRipper";
        this.Size = new System.Drawing.Size(1008, 587);
        this.Load += new System.EventHandler(this.MainWindow_Load);
        this.oPnlFill.ResumeLayout(false);
        this.oPnlMain.ResumeLayout(false);
        this.oPnlRipper.ResumeLayout(false);
        this.oPnlLeft.ResumeLayout(false);
        this.oPnlTop.ResumeLayout(false);
        this.oPnlBottom.ResumeLayout(false);
        this.ResumeLayout(false);
    }
    #endregion
    #region Private Methods

    #region Ripper.CDDrive operations

    /// <summary>
    /// This is an event that occurs when the uctCDRipper control is 1st loaded. This
would
    /// be when a form containing this control is 1st shown.
    /// This event sets up a Ripper.CDDrive (From Idaels CD ripper C# Library) an also
    /// creates CDInserted and CDRemoved event handler delegates.
    /// Then the Management namespace is used to retrieve all CD's on the users system
    /// to display in the controls CD drive combobox. Then the 1st CD drive on the user
    /// system is selected within the running uctCDRipper control.
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void MainWindow_Load(object sender, System.EventArgs e)
    {
        //set up new CD drive and evenet handler delegates
        oCDDrive = new CDDrive();
        oCDDrive.CDInserted += new EventHandler(oCDDrive_CDInserted);
    }

```

```

oCDDrive.CDRemoved += new EventHandler(oCDDrive_CDRemoved);

//CD type drive = 5
const int CD = 5;
// get a list of all drives on users system, then filter out CD Drives and add
//them to the CD drive combobox (oCmbDrives)
ManagementObjectSearcher query = new ManagementObjectSearcher("SELECT * From
Win32_LogicalDisk ");
ManagementObjectCollection queryCollection = query.Get();
foreach ( ManagementObject mo in queryCollection)
{
    int diskType = int.Parse(mo["DriveType"].ToString());
    if (diskType == CD)
    {
        oCmbDrives.Items.Add (mo["Name"] + " " + mo["Description"]);
    }
}
//if there was a least one CD drive, select the 1st one
if (oCmbDrives.Items.Count > 0 )
{
    oCmbDrives.SelectedIndex = 0;
}
}

/// <summary>
/// Simply updates the enabled status for all GUI components based
/// on whether a rip is in progress and the CD Drive status
/// </summary>
private void UpdateVisualControls()
{
    //enabled control based on current ripping progress and CD Drive status
oBtnOpenCD.Enabled = !ripping & (oCmbDrives.SelectedIndex >= 0);
oCmbDrives.Enabled = !ripping & (!oCDDrive.IsOpened);
oBtnEjectCD.Enabled = !ripping & (oCDDrive.IsOpened);
oBtnLoadCD.Enabled = !ripping & (oCDDrive.IsOpened);
oBtnCancelRip.Enabled = ripping;
oBtnSaveAll.Enabled = !ripping & (oLvTracks.Items.Count > 0);
//enable saveFile button if user has selected a track
if (oLvTracks.SelectedIndices.Count > 0)
{
    int track = oLvTracks.SelectedIndices[0]+1;
    oBtnSaveFile.Enabled = !ripping & oCDDrive.IsAudioTrack(track);
}
else
{
    oBtnSaveFile.Enabled = false;
}
}

/// <summary>
/// Event handler that occurs when Track list enabled state changes
/// Simply ensures that the GUI components are updated
/// to show the correct enabled status for each component
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oLvTracks_EnabledChanged(object sender, System.EventArgs e)
{
    UpdateVisualControls();
}

/// <summary>
/// Event handler that occurs when the oCDDrive.ReadTrack (From Idael's CD ripper C#
Library)
/// updates, and is used to update the GUI rip progress bar
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="ea">The event arguments</param>
private void CdReadProgress(object sender, ReadProgressEventArgs ea)
{
    //update the progress bar based on current bytes read by oCDDrive.ReadTrack
    ulong Percent = ((ulong)ea.BytesRead * 100)/ea.Bytes2Read;

```

```

        oProg.Value = (int)Percent;
        Application.DoEvents();
        //should the rip be cancel, do bitwise or to find out
        ea.CancelRead |= this.cancelRipping;
    }

    /// <summary>
    /// Event handler that occurs when the oCDDrive.ReadTrack (From Idaels CD ripper C#
Library)
    /// updates, and is to write to write to the output wavWriter stream
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="ea">The event arguments</param>
    private void WriteWaveData(object sender, DataReadEventArgs ea)
    {
        //if there is a current wavWriter object
        if ( oWavWriter != null)
        {
            //write to its stream
            oWavWriter.Write(ea.Data, 0, (int)ea.DataSize);
        }
    }

    /// <summary>
    /// Save an individual audio track (WAV or MP3) from CD to a disk file. This
    /// method is used by both the Save File and Save CD buttons. This method makes
    /// use of a WaveWriter and various event delegates to perform the ripping
    /// (From Idaels CD ripper C# Library)
    /// </summary>
    /// <param name="track">An int representing the current track numer, used for
updating status
    /// bar information</param>
    /// <param name="OutFileName">A string that represents the output file path</param>
    private void saveTrack(int track, string OutFileName)
    {
        //set up information required for waveformat
        WaveFormat Format = new WaveFormat(44100,16,2);

        //create a new file stream
        Stream WaveFile = new FileStream(OutFileName, FileMode.Create, FileAccess.Write)
;

        try
        {
            if (RipOperation == ripType.RIP_WAV)
            {
                //create a new WaveWriter (From free C# Library)
                oWavWriter = new WaveWriter(WaveFile, Format);
            }
            else if (RipOperation == ripType.RIP_MP3)
            {
                //create a new WaveWriter (From free C# Library)
                oWavWriter = new Mp3Writer(WaveFile, Format);
            }

            try
            {
                //update the status bar and GUI components
                lblStatus.Text = string.Format("Reading track {0}", track);
                UpdateVisualControls();
                DateTime InitTime = DateTime.Now;
                //start the read
                if ( oCDDrive.ReadTrack(track, new CdDataReadEventHandler(WriteWaveData)
, new CdReadProgressEventHandler(this.CdReadProgress)) > 0)
                {
                    TimeSpan Duration = DateTime.Now - InitTime;
                    double Speed = oCDDrive.TrackSize(track) / Duration.TotalSeconds /
Format.nAvgBytesPerSec;
                    lblStatus.Text = string.Format("Track {0} read at {1:0.00} X",
track, Speed);
                }
            }
            else

```

```

        { //there must have been a problem with the read, or the user
cancelled it
            if (! cancelRipping)
            {
                lblStatus.Text = string.Format("There was an error reading
track {0}", track);
            }
            //close resources
            oWavWriter.Close();
            WaveFile.Close();
            //and delete the faulty file, and set rip progress to 0
            if ( File.Exists(OutFileName))
            {
                File.Delete(OutFileName);
            }
            oProg.Value = 0;
        }
        if (cancelRipping)
        {
            ripping = false;
        }
    }
    //close resources
finally
    {
        oWavWriter.Close();
        oWavWriter = null;
    }
}
//close resources
finally
    {
        WaveFile.Close();
    }
}

/// <summary>
/// Event handler that occurs when the oCDDrive.CDInserted (From Idaels CD ripper C#
Library)
/// occurs, and is used to update the GUI rip progress bar and also to display
/// any Audio tracks that are contained on the CD
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oCDDrive_CDInserted(object sender, EventArgs e)
{
    //clear track list and progress status
    oLvTracks.Items.Clear();
    oProg.Value = 0;
    //if CD ready and has audio tracks, display them on the GUI component
    if ( oCDDrive.IsCDReady() )
    {
        lblStatus.Text = "CD inserted and ready";
        if ( oCDDrive.Refresh() )
        {
            int Tracks = oCDDrive.GetNumTracks();
            for (int i = 1; i <= Tracks; i++)
            {
                ListViewItem item = new ListViewItem(new string[] {i.ToString(),
oCDDrive.TrackSize(i).ToString(), oCDDrive.IsAudioTrack(i)?"audio":"data"});
                oLvTracks.Items.Add(item);
            }
        }
    }
    //CD was not ready
    else
    {
        lblStatus.Text = "CD inserted but not ready";
    }
    //update GUI status
    UpdateVisualControls();
}

```

```

    /// <summary>
    /// Event handler that occurs when the oCDDrive.CDRemoved (From Idaels CD ripper C# Library)
    /// occurs, and is used to update the GUI rip progress bar and also to clear and CD
    /// tracks that may be currently shown
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void oCDDrive_CDRemoved(object sender, EventArgs e)
    {
        oLvTracks.Items.Clear();
        oProg.Value = 0;
        olblStatus.Text = "CD Removed";
        UpdateVisualControls();
    }

#endregion
#region GUI User interactions

    /// <summary>
    /// Event handler that occurs when user selects different CD drive.
    /// When the user selects a new CD Drive the GUI components are updated
    /// to show the correct enabled status for each component
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void oCmbDrives_SelectedIndexChanged(object sender, System.EventArgs e)
    {
        UpdateVisualControls();
    }

    /// <summary>
    /// Event handler that occurs when user selects Open CD button.
    /// The open CD button has 2 actions it is really a toggle that
    /// can open and close the CD (From Idaels CD ripper C# Library).
    /// If it is opened the list of audio tracks will be displayed that
    /// the CD contains within the GUI. If it is closed the CD drive will
    /// be closed and all audio tracks shown will be cleared from the GUI
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void oBtnOpenCD_Click(object sender, System.EventArgs e)
    {
        //If CD was open, close it and clear any tracks that may have
        //been displayed
        if ( oCDDrive.IsOpened )
        {
            oCDDrive.Close();
            oBtnOpenCD.Text = "    Open";
            olblStatus.Text = "CD drive closed";
            oLvTracks.Items.Clear();
        }
        else
        {
            //CD was closed, so try and open it,and see if it is ready
            if ( oCDDrive.Open(oCmbDrives.Text[0]) )
            {
                olblStatus.Text = "CD drive opened";
                if ( oCDDrive.IsCDReady() )
                {
                    olblStatus.Text += " and ready";
                    if ( oCDDrive.Refresh() )
                    {
                        //open and ready so get its tracks
                        int Tracks = oCDDrive.GetNumTracks();
                        for (int i = 1; i <= Tracks; i++)
                        {
                            ListViewItem item = new ListViewItem(new string[] {i.
ToString(), oCDDrive.TrackSize(i).ToString(), oCDDrive.IsAudioTrack(i)?"audio":"data"},
0);
                            oLvTracks.Items.Add(item);
                        }
                    }
                }
            }
        }
    }

```

```

        }
        oBtnOpenCD.Text = "    Close";
    }
    //could not open CD drive
else
{
    lblStatus.Text = "CD drive could not be opened";
}
}
//init rip progress to 0
oProg.Value = 0;
//update GUI
UpdateVisualControls();
}

/// <summary>
/// Event handler that occurs when user selects Eject CD button.
/// Simply ejects the current CD (From Idaels CD ripper C# Library) and
/// clears all displayed audio tracks and sets rip progress to 0
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oBtnEjectCD_Click(object sender, System.EventArgs e)
{
    oLvTracks.Items.Clear();
    oProg.Value = 0;
    if ( oCDDrive.EjectCD() )
    {
        lblStatus.Text = "CD ejected";
    }
    else
    {
        lblStatus.Text = "CD could not be ejected";
    }
}

/// <summary>
/// Event handler that occurs when user selects Load CD button.
/// Simply loads the current CD (From Idaels CD ripper C# Library) and
/// clears all displayed audio tracks and sets rip progress to 0
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oBtnLoadCD_Click(object sender, System.EventArgs e)
{
    oLvTracks.Items.Clear();
    oProg.Value = 0;
    if ( oCDDrive.LoadCD() )
    {
        lblStatus.Text = "CD loaded";
    }
    else
    {
        lblStatus.Text = "CD could not be loaded";
    }
}

/// <summary>
/// Event handler that occurs when the user selects the Save File button.
/// Will prompt the user for the type of file WAV or MP3 and a file location.
/// Once the user enters these details the CD is locked (From Idaels CD ripper C#
Library)
/// and then the track is saved, and finally the CD is unlocked
/// (From Idaels CD ripper C# Library)
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oBtnSaveFile_Click(object sender, System.EventArgs e)
{
    //has user selected track
    if (oLvTracks.SelectedIndices.Count > 0 )
    {
        //if so ask them where to store it

```

```

int track = oLvTracks.SelectedIndices[0]+1;

//get rip type WAV or MP3 from modal dialog
FormRipType rt = new FormRipType();
rt.ShowDialog(this);

//This next line will not be processed until the user replies to the
previous //FormRipType form operation, this is because is it shown modally, so the
current //thread has to deal with that operation 1st, then comes back to here
if (RipOperation == ripType.RIP_WAV)
{
    oSaveFileDialog.FileName = string.Format("track{0:00}.wav", track);
    oSaveFileDialog.Filter = "Wave files (*.wav)|*.wav|All files (*.*)|*.*";
}
else if (RipOperation == ripType.RIP_MP3)
{
    oSaveFileDialog.FileName = string.Format("track{0:00}.mp3", track);
    oSaveFileDialog.Filter = "MP3 files (*.mp3)|*.mp3|All files (*.*)|*.*";
}

#region fileDialog == OK
//has the user selected OK, to save file dialog
if (oSaveFileDialog.ShowDialog() == DialogResult.OK)
{
    string fileName = oSaveFileDialog.FileName;
    //is the file name still valid
    if (fileName.Equals(string.Empty))
    {
        MessageBox.Show("There was an error with the path specified, it can
not be null",
        "Track Selection ERROR",MessageBoxButtons.OK,MessageBoxIcon.
Error);
        return;
    }
    //is the file still valid for the current rip operation
    else
    {
        if ( (!fileName.ToLower().EndsWith("wav") && RipOperation ==
ripType.RIP_WAV) ||
        (!fileName.ToLower().EndsWith("mp3") && RipOperation ==
ripType.RIP_MP3) )
        {
            MessageBox.Show("You have attempted to save the file using an
extension that is\n" +
            "incompatable with the current rip operation, please retry",
            "Track Selection ERROR",MessageBoxButtons.OK,MessageBoxIcon.
Error);
            return;
        }
        //file is valid and of correct rip type
        else
        {
            //start ripping
            ripping = true;
            try
            {
                oCDDrive.LockCD();
                try
                {
                    //save file
                    showprogress(true);
                    saveTrack(track,oSaveFileDialog.FileName);
                }
                finally
                {
                    oCDDrive.UnLockCD();
                }
            }
            //stop ripping
            finally
            {
                ripping = false;
            }
        }
    }
}

```

```

    }
    }
}
#endregion
}

//clear any cancelRipping and hide progress bar status
cancelRipping = false;
showprogress(false);
//update GUI status
UpdateVisualControls();
}

/// <summary>
/// Event handler that occurs when the user selects the Save All button.
/// Will prompt the user for the type of file WAV or MP3 and a folder location.
/// Once the user enters these details the CD is locked (From Idaels CD ripper C#
Library)
/// and then the tracks are saved, and finally the CD is unlocked
/// (From Idaels CD ripper C# Library)
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oBtnSaveAll_Click(object sender, System.EventArgs e)
{

    //deselect any previous items
    foreach (ListViewItem it in oLvTracks.SelectedItems)
    {
        it.Selected = false;
    }

    //get rip type WAV or MP3 from modal dialog
    FormRipType rt = new FormRipType();
    rt.ShowDialog(this);

    //This next line will not be processed until the user replies to the previous
    //FormRipType form operation, this is because is it shown modally, so the
current
    //thread has to deal with that operation 1st, then comes back to here

    //start at desktop
    oFolderBrowserDialog.SelectedPath = "Desktop";
    //has the user selected OK, to save file dialog
    if (oFolderBrowserDialog.ShowDialog() == DialogResult.OK)
    {
        //is the file name still valid
        if (oFolderBrowserDialog.SelectedPath.Equals(string.Empty))
        {
            MessageBox.Show("There was an error with the path specified, it can not
be null",
                "Track Selection ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        else
        {
            //start ripping
            ripping = true;
            try
            {
                oCDDrive.LockCD();
                try
                {
                    showprogress(true);
                    //save files while not cancelling
                    while (cancelRipping != true)
                    {
                        for (int i=0;i<oLvTracks.Items.Count;i++)
                        {
                            int track = i+1;
                            oLvTracks.Items[i].Selected = true;
                            string filename="";

```

```

        if (RipOperation == ripType.RIP_WAV)
        {
            filename = oFolderBrowserDialog.SelectedPath + @"\
Track_" + track.ToString() + ".wav";
        }
        else if (RipOperation == ripType.RIP_MP3)
        {
            filename = oFolderBrowserDialog.SelectedPath + @"\
Track_" + track.ToString() + ".mp3";
        }
        saveTrack(track, filename);
        oLvTracks.Items[i].Selected = false;
    }
}
}
finally
{
    oCDDrive.UnLockCD();
}
}
//stop ripping
finally
{
    ripping = false;
}
}
}
//clear any cancelRipping and hide progress bar status
cancelRipping = false;
showprogress(false);
//update GUI status
UpdateVisualControls();
}

/// <summary>
/// Simply cancels the current rip and sets the progress to 0
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oBtnCancelRip_Click(object sender, System.EventArgs e)
{
    cancelRipping = true;
    olblStatus.Text = "CD Rip aborted by user";
    oProg.Value = 0;
}

/// <summary>
/// Event handler that occurs when user selects a new Track Number
/// Simply ensures that the GUI components are updated
/// to show the correct enabled status for each component
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oLvTracks_SelectedIndexChanged(object sender, System.EventArgs e)
{
    UpdateVisualControls();
}

#endregion
#region GUI look and feel methods

/// <summary>
/// Help method to show/hide progress information
/// </summary>
/// <param name="state">>true to show progress information, false otherwise</param>
private void showprogress(bool state)
{
    oBtnCancelRip.Enabled = state;
    oProg.Visible = state;
    oLblProgress.Visible = state;
}
}

```

```

/// <summary>
/// Event handler that occurs when the main uctCDRipper panel paints
/// Simply paints the main uctCDRipper panel using a colorBlend
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oPnlFill_Paint(object sender, System.Windows.Forms.PaintEventArgs e)
{
    try
    {
        //call base class paint method
        base.OnPaint(e);

        //paint the panel using a color belnd
        Color[] myColors = {
            Color.Gainsboro,
            Color.White,
            Color.White };
        float[] myPositions = {0.0f,0.4f,1.0f};
        ColorBlend myBlend = new ColorBlend();
        myBlend.Colors = myColors;
        myBlend.Positions = myPositions;

        Rectangle r1 = new Rectangle(0,0,oPnlFill.Bounds.Width,oPnlFill.Bounds.
Height);
        LinearGradientBrush lgBrush2 = new LinearGradientBrush(r1,Color.Gainsboro,
Color.White,45.0f,true);
        lgBrush2.InterpolationColors = myBlend;

        e.Graphics.FillRectangle(lgBrush2,0,0,oPnlFill.Bounds.Width,oPnlFill.Bounds.
Height);
    }
    //nothing to do occurs when, going from Minimised to maxmise and rectange to
draw is effectievly
    //0 Wide 0 High, which causes an exception. Cant do anything about it.
    catch(Exception ex)
    {
    }

}

/// <summary>
/// Event handler that occurs when the main uctCDRipper panel resizes.
/// Simply used to refresh the main uctCDRipper panel area.
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oPnlFill_Resize(object sender, System.EventArgs e)
{
    oPnlFill.Invalidate(oPnlFill.Bounds);
    this.Refresh();
}

/// <summary>
/// Event handler that occurs when the main uctCDRipper bottom panel occurs
/// Simply paints the main uctCDRipper bottom panel using a brush
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oPnlBottom_Paint(object sender, System.Windows.Forms.PaintEventArgs e)
{
    try
    {
        //call base class paint method
        base.OnPaint(e);
    }
}

```

```

        Graphics g = e.Graphics;
        Pen penBlack = new Pen(Color.Black);
        Rectangle r = new Rectangle(0,0,oPnlBottom.Size.Width,oPnlBottom.Size.
Height);
        g.DrawRectangle(penBlack,r);
        Brush divBrush = new HatchBrush(HatchStyle.Divot,Color.Gray,Color.Black);
        g.FillRectangle(divBrush,r);
    }
    //nothing to do occurs when, going from Minimised to maxmise and rectange to
draw is effectievly
    //0 Wide 0 High, which causes an exception. Cant do anything about it.
    catch(Exception ex)
    {
    }
}

/// <summary>
/// Event handler that occurs when the main uctCDRipper panel resizes.
/// Simply reposition all the relevant GUI components based on the new
/// client window size.
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oPnlMain_Resize(object sender, System.EventArgs e)
{
    oPnlMain.Invalidate(oPnlMain.Bounds);
    oPnlRipper.Left = (oPnlMain.Width/2 - oPnlRipper.Width/2) - oPnlLeft.Width/2;
    this.Refresh();
}

/// <summary>
/// Event handler that occurs when the CD Drives comobo box items are drawn
/// Simply draw our own custom style combo box items.
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oCmbDrives_DrawItem(object sender, System.Windows.Forms.
DrawItemEventArgs e)
{
    //check index is valid
    if (e.Index != -1)
    {
        // Draw the background of the ComboBox control for each item.
        e.DrawBackground();

        Image img = oImgListCD.Images[0];
        e.Graphics.DrawImage(img, new Point(e.Bounds.X, e.Bounds.Y));

        // Draw the current item text based on the current Font and the custom brush
settings.
        e.Graphics.DrawString(oCmbDrives.Items[e.Index].ToString(), e.Font, Brushes.
Black,
            new Point(img.Width*2,e.Bounds.Y + 10));
        e.Graphics.DrawImage(img, new Point(e.Bounds.X, e.Bounds.Y));

        // If the ComboBox has focus, draw a focus rectangle around the selected
item.
        e.DrawFocusRectangle();
    }
}
#endregion
#endregion
}
#endregion
}

```