

```
1 using Microsoft.Win32;
2 using System;
3 using System.Collections.Generic;
4 using System.Drawing;
5 using System.IO;
6 using System.Linq;
7 using System.Runtime.InteropServices;
8 using System.Windows.Forms;
9
10 //-----\\
11 // Author: Nikola Nejedlý \\
12 // Kybersoft (c) 2017 \\
13 // Web: http://nejedniko.tk \\
14 // TUL | FM:IL - Project \\
15 // .NET 3.5 - support winXP \\
16 //-----\\
17
18
19 namespace GifSlider
20 {
21     public partial class Form1 : Form
22     {
23         double volume = 100;
24
25         string version = "v1.0201";
26
27         WMPLib.WindowsMediaPlayer wplayer = new WMPLib.WindowsMediaPlayer();
28
29         string songtoplay = "None";
30         public static Slideshow slides = new Slideshow();
31         Image img;
32         int time = 3000;
33         public static List<string> imglist = new List<string>();
34         Tooltip tip = new Tooltip();
35         bool playguide = true;
36         string[] options = { "Always on top", "Play song", "Next image on timer", "Stretch images", "Fit images", "1:1 images", "No border fullscreen", "Random order", "> Random with repetition", "Start from selected", "Prevent Windows from going IDLE ", "Play 'Help' audio"};
37
38
39         public Form1()
40         {
41             InitializeComponent();
42             labelversion.Text = version;
43
44             if (!UacHelper.IsUacEnabled || !UacHelper.IsProcessElevated)
45             {
46                 checkBox_associate.Enabled = false;
47             }
48
49             using (RegistryKey Key = Registry.ClassesRoot.OpenSubKey("GIFSlider_File"))
50             {
51                 if (Key != null)
52                 {
53                     
```

```

52         checkBox_associate.Checked = true;
53     }
54
55 }
56
57 //START -- Association | .gs
58 public static void SetAssociation(string Extension, string KeyName,
59     string OpenWith, string FileDescription)
60 {
61     if (!UacHelper.IsUacEnabled || !UacHelper.IsProcessElevated)
62         return;
63
64     RegistryKey BaseKey;
65     RegistryKey OpenMethod;
66     RegistryKey Shell;
67     RegistryKey CurrentUser;
68
69     BaseKey = Registry.ClassesRoot.CreateSubKey(Extension);
70     BaseKey.SetValue("", KeyName);
71
72     OpenMethod = Registry.ClassesRoot.CreateSubKey(KeyName);
73     OpenMethod.SetValue("", FileDescription);
74     OpenMethod.CreateSubKey("DefaultIcon").SetValue("", "\\\" +
75         OpenWith + "\\\", 0);
76     Shell = OpenMethod.CreateSubKey("Shell");
77     Shell.CreateSubKey("edit").CreateSubKey("command").SetValue("",
78         "\\\" + OpenWith + "\\\" + \"%1\");
79     Shell.CreateSubKey("open").CreateSubKey("command").SetValue("",
80         "\\\" + OpenWith + "\\\" + \"%1\");
81     BaseKey.Close();
82     OpenMethod.Close();
83     Shell.Close();
84
85     CurrentUser = Registry.CurrentUser.OpenSubKey("Software\\Microsoft
86         \\Windows\\CurrentVersion\\Explorer\\FileExts\\" + Extension,
87         true);
88     CurrentUser.DeleteSubKey("UserChoice", false);
89     CurrentUser.Close();
90
91     SHChangeNotify(0x08000000, 0x0000, IntPtr.Zero, IntPtr.Zero);
92 }
93
94 [DllImport("shell32.dll", CharSet = CharSet.Auto, SetLastError =
95     true)]
96 //END -- Association | .gs
97
98 public static extern void SHChangeNotify(uint wEventId, uint uFlags,
99     IntPtr dwItem1, IntPtr dwItem2);
100
101 private string ReadLineF(int line, string file)
102 {
103     if (!File.Exists(file))
104         return "";
105
106     string stringf = "";
107 }

```

```
100     List<string> lines = new List<string>();
101     using (var reader = new StreamReader(file))
102     {
103         while (!reader.EndOfStream)
104             lines.Add(reader.ReadLine());
105     }
106
107     int i = 0;
108     foreach (var lin in lines)
109     {
110         if (i == line)
111             stringf = lin;
112         i++;
113     }
114     return stringf;
115 }
116
117 private void UpdateSettings(string file)
118 {
119     if (File.Exists(file))
120         File.Delete(file);
121
122     string text = "";
123     int i = 0;
124     foreach (var item in checkedListBox1.Items)
125     {
126         if (checkedListBox1.GetItemChecked(i) == true)
127             text += "1";
128         else
129             text += "0";
130         i++;
131     }
132
133     using (StreamWriter outputFile = new StreamWriter(file))
134     {
135         outputFile.WriteLine(text);
136         outputFile.WriteLine(time.ToString());
137         outputFile.WriteLine(checkboxLoopmp3.Checked.ToString());
138         outputFile.WriteLine(songtoplay);
139         outputFile.WriteLine(volume.ToString());
140     }
141 }
142
143 private void InitSettings()
144 {
145
146     try
147     {
148         string linefile = "";
149         if (File.Exists(Application.StartupPath + "\\settings.ini"))
150         {
151             linefile = ReadLineF(0, Application.StartupPath + "\\
152                                     \settings.ini");
153             if (Int32.TryParse(ReadLineF(1, Application.StartupPath +
154                                     "\\settings.ini"), out int res))
155                 time = Int32.Parse(ReadLineF(1,
```

```
Application.StartupPath + "\\settings.ini"));
154     }
155
156     int i = 0;
157     foreach (string opt in options)
158     {
159         checkedListBox1.Items.Add(opt);
160
161         if (File.Exists(Application.StartupPath + "\\settings.ini")) ➤
162         {
163             if (linefile.Substring(i, 1) == "0")
164                 checkedListBox1.SetItemCheckState(i, ➤
165                 CheckState.Unchecked);
166             else
167                 checkedListBox1.SetItemCheckState(i, ➤
168                 CheckState.Checked);
169         }
170         if (!File.Exists(Application.StartupPath + "\\settings.ini")) ➤
171             checkedListBox1.SetItemCheckState(i, ➤
172             CheckState.Unchecked);
173         i++;
174     }
175
176     if (File.Exists(Application.StartupPath + "\\settings.ini"))
177     {
178         linefile = ReadLineF(2, Application.StartupPath + "\\settings.ini"); ➤
179         checkBox_loopmp3.Checked = bool.Parse(linefile);
180     }
181
182     if (File.Exists(Application.StartupPath + "\\settings.ini"))
183     {
184         linefile = ReadLineF(3, Application.StartupPath + "\\settings.ini"); ➤
185         songtoplay = linefile;
186         if (songtoplay == "" || songtoplay == "None")
187         {
188             slides.playsong = false;
189             songtoplay = "None";
190         }
191     }
192
193     if (File.Exists(Application.StartupPath + "\\settings.ini"))
194     {
195         linefile = ReadLineF(4, Application.StartupPath + "\\settings.ini"); ➤
196         volume = double.Parse(linefile);
197     }
198     trackBar_volume.SetRange(0,100);
199     trackBar_volume.Value = (int)volume;
200     label_vol.Text = volume.ToString();
201 }
202 catch
```

```

201     {
202         MessageBox.Show("An initiation error happened. Try starting
the application again. \r\nSome of your settings might get
reset.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
203         volume = 50;
204         this.Close();
205     }
206
207     if (Environment.GetCommandLineArgs() != null)
208     {
209         string[] s = Environment.GetCommandLineArgs();
210
211         if (s.Length >= 2)
212         {
213             if (s[1].Substring(s[1].Length - 3) == ".gs")
214                 using (var reader = new StreamReader(s[1]))
215                 {
216                     while (!reader.EndOfStream)
217                     {
218                         string tmp = reader.ReadLine();
219                         if (File.Exists(tmp))
220                         {
221                             imglist.Add(tmp);
222                             listBox1.Items.Add(Path.GetFileName(tmp));
223                         }
224                     }
225                 }
226             else
227                 if (Directory.Exists(s[1]))
228                 {
229                     foreach (string f in Directory.GetFiles(s[1]))
230                     {
231                         if (File.Exists(f))
232                             if (f.Substring(f.Length - 4).ToLower
().ToLower() == ".jpg" || f.Substring(f.Length -
5).ToLower() == ".jpeg" ||
233                                 f.Substring(f.Length - 4).ToLower() ==
".png" || f.Substring(f.Length - 4).ToLower() == ".gif" ||
234                                 f.Substring(f.Length - 4).ToLower() ==
".bmp")
235                         {
236                             imglist.Add(f);
237                             listBox1.Items.Add(Path.GetFileName(f));
238                         }
239                     }
240                 }
241
242                 foreach (string f in s)
243                 {
244                     if (File.Exists(f))
245                         if (f.Substring(f.Length - 4).ToLower() == ".jpg" ||
f.Substring(f.Length - 5).ToLower() == ".jpeg" ||
246                             f.Substring(f.Length - 4).ToLower() == ".png" ||
f.Substring(f.Length - 4).ToLower() == ".gif" ||
247                             f.Substring(f.Length - 4).ToLower() == ".bmp")
248                     {

```

```
249         imglist.Add(f);
250         listBox1.Items.Add(Path.GetFileName(f));
251     }
252 }
253 }
254 }
255
256     listBox1.SelectedIndex = listBox1.Items.Count - 1;
257     listBox1.Focus();
258 }
259
260 private void Form1_Load(object sender, EventArgs e)
261 {
262     InitSettings();
263     listBox1.SelectedIndex = -1;
264     button_movedown.Enabled = false;
265     button_moveup.Enabled = false;
266     checkedListBox1.CheckOnClick = true;
267     button_minus.Enabled = false;
268
269     Refresh();
270     if (imglist.Count == 0)
271         button_start.Enabled = false;
272     else
273         button_start.Enabled = true;
274
275     imageTimer.Text = time.ToString();
276     ApplySettings();
277
278     int htemp = Screen.PrimaryScreen.WorkingArea.Size.Height;
279     int wtemp = Screen.PrimaryScreen.WorkingArea.Size.Width;
280     this.SetDesktopLocation(wtemp / 2 - this.Width / 2, htemp / 2 -
        this.Height / 2);
281
282 }
283
284 private void Form1_Paint(object sender, PaintEventArgs e)
285 {
286     Pen p = new Pen(Color.White);
287     Brush b = new SolidBrush(Color.Black);
288     Brush b1 = new SolidBrush(Color.White);
289     e.Graphics.FillRectangle(b, 316, 12, 300, 200);
290     e.Graphics.DrawRectangle(p, 315, 11, 301, 201);
291
292     Font f = new Font(FontFamily.GenericSerif, 8);
293
294     if (img == null || img.Size.IsEmpty)
295         e.Graphics.DrawString("Preview (static)", f, b1, 320, 13);
296     else
297         e.Graphics.DrawString("Preview", f, b1, 320, 13);
298
299     if (img != null)
300         if (!img.Size.IsEmpty)
301         {
302             double h; double w;
303             int x = 0; int y = 0;
```

```

304         Point pp = new Point(img.Size.Width, img.Size.Height);
305
306         if (img.Size.Height > 200 || img.Size.Width > 300)
307             pp = fit(img, 300, 200);
308
309         w = pp.X;
310         h = pp.Y;
311
312         if (w < 300)
313             x = 300 / 2 - (int)(w / 2);
314         if (h < 200)
315             y = 200 / 2 - (int)(h / 2);
316
317         e.Graphics.DrawImage(img, 316+x, 12+y, (int)w, (int)h);
318     }
319 }
320
321 private void button_plus_Click(object sender, EventArgs e)
322 {
323     Stream myStream;
324     OpenFileDialog thisDialog = new OpenFileDialog();
325     //thisDialog.InitialDirectory = "c:\\";
326     thisDialog.Filter = "Image files (*.bmp, *.jpeg, *.jpg, *.png)|
        *.bmp;*.jpeg;*.jpg;*.png|GIF files (*.gif)|*.gif|GIFSlider files
        (*.gs)|*.gs|All Supported (*.gs, *.bmp, *.jpeg, *.jpg, *.png,
        *.gif)|*.gs; *.bmp;*.jpeg;*.jpg;*.png;*.gif";
327     thisDialog.FilterIndex = 4;
328     thisDialog.RestoreDirectory = false;
329     thisDialog.Multiselect = true;
330     thisDialog.Title = "Please Select Desired Image File(s)";
331
332     if (thisDialog.ShowDialog() == DialogResult.OK)
333     {
334         foreach (String file in thisDialog.FileNames)
335         {
336             try
337             {
338                 if ((myStream = thisDialog.OpenFile()) != null)
339                 {
340                     using (myStream)
341                     {
342                         if (file.Substring(file.Length - 3) == ".gs")
343                             using (var reader = new StreamReader
344                                 (file))
345                             {
346                                 while (!reader.EndOfStream)
347                                 {
348                                     string tmp = reader.ReadLine();
349                                     if (File.Exists(tmp))
350                                     {
351                                         imglist.Add(tmp);
352                                         listBox1.Items.Add
353                                             (Path.GetFileName(tmp));
354                                     }
355                                 }
356                             }
357                         }
358                     }
359                 }
360             }
361         }
362     }
363 }

```


```
355         else
356         {
357             imglist.Add(file);
358             listBox1.Items.Add(Path.GetFileName
359                 (file));
360         }
361     }
362 }
363
364 catch (Exception ex)
365 {
366     MessageBox.Show("Error: Could not read file. Error: "
367         + ex.Message);
368 }
369 }
370 listBox1.SelectedIndex = imglist.Count - 1;
371 listBox1.Focus();
372 }
373
374 private void button_minus_Click(object sender, EventArgs e)
375 {
376     if (listBox1.SelectedIndex >= 0 && listBox1.SelectedIndex <=
377         listBox1.Items.Count)
378     {
379         int index = listBox1.SelectedIndex;
380         if (index == -1) return;
381
382         string addr = imglist[index];
383         imglist.RemoveAt(index);
384         listBox1.Items.RemoveAt(index);
385
386         if (index > -1)
387         {
388             if(index > 0)
389                 listBox1.SelectedIndex = index - 1;
390             else
391                 if(index == 0 && listBox1.Items.Count > 1)
392                     listBox1.SelectedIndex = index;
393         }
394         img = null;
395     }
396 }
397
398 private void button_start_Click(object sender, EventArgs e)
399 {
400     ApplySettings();
401     stopMp3();
402     slides.timer.Enabled = true;
403     slides.TheParent = this;
404     slides.time = time;
405     slides.Show();
406     this.Hide();
407     this.Enabled = false;
```



```
408
409     private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
410     {
411
412         if (imglist.Count == 0)
413         {
414             button_minus.Enabled = false;
415             button_start.Enabled = false;
416             img = null;
417             Refresh();
418         }
419         else
420         {
421             button_minus.Enabled = true;
422             button_start.Enabled = true;
423         }
424         int ind = listBox1.SelectedIndex;
425         if (ind == -1)
426         {
427             button_movedown.Enabled = false;
428             button_moveup.Enabled = false;
429             return;
430         }
431
432         if (ind >= imglist.Count - 1 || ind == -1)
433             button_movedown.Enabled = false;
434         else
435             button_movedown.Enabled = true;
436         if (ind == 0 || ind == -1)
437             button_moveup.Enabled = false;
438         else
439             button_moveup.Enabled = true;
440
441         string addr = imglist[ind];
442
443         if (!File.Exists(addr))
444         { imglist.RemoveAt(ind); listBox1.Items.RemoveAt(ind); return; }
445
446         if(img != null)
447             img.Dispose();
448
449         try
450         {
451             img = Image.FromFile(addr);
452         }
453         catch
454         {
455             img = null;
456         }
457
458         Refresh();
459     }
460
461     private void button_clean_Click(object sender, EventArgs e)
462     {
463         imglist.Clear();
```

```
464         listBox1.ClearSelected();
465         listBox1.Items.Clear();
466         button_start.Enabled = false;
467         img = null;
468     }
469
470     private void listBox1_DragDrop(object sender, DragEventArgs e)
471     {
472         string[] FileList = (string[])e.Data.GetData(DataFormats.FileDrop,
473             false);
474         foreach (var file in FileList) // 1. level
475         {
476             if (file.Substring(file.Length - 4).ToLower() == ".jpg" ||
477                 file.Substring(file.Length - 5).ToLower() == ".jpeg" ||
478                 file.Substring(file.Length - 4).ToLower() == ".png" ||
479                 file.Substring(file.Length - 4).ToLower() == ".gif" ||
480                 file.Substring(file.Length - 4).ToLower() == ".bmp")
481             {
482                 imglist.Add(file);
483                 listBox1.Items.Add(Path.GetFileName(file));
484             }
485             else
486             if (Directory.Exists(file))
487             {
488                 foreach (var ffile in Directory.GetFiles(file)) // 2.
489                     level
490                     {
491                         if (ffile.Substring(ffile.Length - 4).ToLower() ==
492                             ".jpg" || ffile.Substring(ffile.Length - 5).ToLower() ==
493                             ".jpeg" ||
494                             ffile.Substring(ffile.Length - 4).ToLower() ==
495                             ".png" || ffile.Substring(ffile.Length - 4).ToLower() ==
496                             ".gif" ||
497                             ffile.Substring(ffile.Length - 4).ToLower() ==
498                             ".bmp")
499                         {
500                             imglist.Add(ffile);
501                             listBox1.Items.Add(Path.GetFileName(ffile));
502                         }
503                     }
504                 foreach (string dffile in Directory.GetDirectories(file))
505                 if (Directory.Exists(dffile))
506                     foreach (var fffile in Directory.GetFiles
507                         (dffile)) // 3. level
508                     {
509                         if (fffile.Substring(fffile.Length -
510                             4).ToLower() == ".jpg" || fffile.Substring(fffile.Length -
511                             5).ToLower() == ".jpeg" ||
512                             fffile.Substring(fffile.Length -
513                             4).ToLower() == ".png" || fffile.Substring(fffile.Length -
514                             4).ToLower() == ".gif" ||
515                             fffile.Substring(fffile.Length -
516                             4).ToLower() == ".bmp")
517                         {
518                             imglist.Add(fffile);
519                         }
520                     }
521             }
522         }
523     }
524 }
```

```
505         listBox1.Items.Add(Path.GetFileName
                    (fffile));
506             }
507         }
508     }
509 }
510 else
511 if (file.Substring(file.Length - 3) == ".gs")
512 {
513     using (var reader = new StreamReader(file))
514     {
515         while (!reader.EndOfStream)
516         {
517             string tmp = reader.ReadLine();
518             if (File.Exists(tmp))
519             {
520                 imglist.Add(tmp);
521                 listBox1.Items.Add(Path.GetFileName(tmp));
522             }
523         }
524     }
525 }
526 }
527
528
529 listBox1.SelectedIndex = imglist.Count - 1;
530 listBox1.Focus();
531 }
532
533 private void listBox1_DragEnter(object sender, DragEventArgs e)
534 {
535     if (e.Data.GetDataPresent(DataFormats.FileDrop))
536     {
537         e.Effect = DragDropEffects.Copy;
538     }
539     else
540     {
541         e.Effect = DragDropEffects.None;
542     }
543 }
544
545 private void button_moveup_Click(object sender, EventArgs e)
546 {
547     int index = listBox1.SelectedIndex;
548     int newindex = index - 1;
549     if (newindex <= -1)
550         return;
551
552     List<string> intmp = listBox1.Items.OfType<string>().ToList();
553     List<string> tmp = Swap(intmp, index, newindex);
554     listBox1.Items.Clear();
555     foreach (string item in tmp)
556     {
557         listBox1.Items.Add(item);
558     }
559 }
```

```
560         tmp = Swap(imglist, index, newIndex);
561         imglist = tmp;
562
563         listBox1.SelectedIndex = newIndex;
564     }
565
566     private void button_movedown_Click(object sender, EventArgs e)
567     {
568         int index = listBox1.SelectedIndex;
569         int newIndex = index + 1;
570         if (newIndex >= listBox1.Items.Count)
571             return;
572
573         List<string> intmp = listBox1.Items.OfType<string>().ToList();
574         List<string> tmp = Swap(intmp, index, newIndex);
575         listBox1.Items.Clear();
576         foreach (string item in tmp)
577         {
578             listBox1.Items.Add(item);
579         }
580
581         tmp = Swap(imglist, index, newIndex);
582         imglist = tmp;
583
584         listBox1.SelectedIndex = newIndex;
585     }
586
587     private List<string> Swap(List<string> l, int index1, int index2)
588     {
589         string A = l[index1];
590         string B = l[index2];
591         l[index1] = B;
592         l[index2] = A;
593
594         return l;
595     }
596
597     private void checkedListBox1_SelectedIndexChanged(object sender, 
598         EventArgs e)
599     {
600         ApplySettings();
601         checkedListBox1.ClearSelected();
602     }
603
604     private void ApplySettings()
605     {
606         if (checkedListBox1.GetItemChecked(0)) // topmost
607         {
608             slides.TopMost = true;
609             slides.topm = true;
610         }
611         else
612         {
613             slides.TopMost = false;
614             slides.topm = false;
```

```

615     }
616
617     if (checkedListBox1.GetItemChecked(1)) // play song
618     {
619         button_selectsong.Enabled = true;
620         checkBox_loopmp3.Enabled = true;
621         trackBar_volume.Enabled = true;
622         textBox_song.Text = songtoplay;
623
624         if (File.Exists(songtoplay) && songtoplay.Substring
625             (songtoplay.Length - 4) != ".mp3")
626         {
627             MessageBox.Show("Your file is not mp3, aborting.", "File
628                 not mp3", MessageBoxButtons.OK, MessageBoxIcon.Warning);
629             return;
630         }
631         if (File.Exists(songtoplay))
632         {
633             slides.playsong = true;
634             slides.song = songtoplay;
635         }
636         else
637         {
638             //MessageBox.Show("Your mp3 file was not found and the
639                 desired mp3 will not play.", "File not
640                 found", MessageBoxButtons.OK, MessageBoxIcon.Warning);
641             slides.playsong = false;
642         }
643     }
644     else
645     {
646         slides.playsong = false;
647         textBox_song.Text = "None";
648         button_selectsong.Enabled = false;
649         checkBox_loopmp3.Enabled = false;
650         trackBar_volume.Enabled = false;
651     }
652
653     if (checkedListBox1.GetItemChecked(2)) // next image on timer
654         slides.nextontime = true;
655     else
656         slides.nextontime = false;
657
658     if (checkedListBox1.SelectedIndex == 3 || (!
659         checkedListBox1.GetItemChecked(3) && !
660         checkedListBox1.GetItemChecked(4) && !
661         checkedListBox1.GetItemChecked(5))) // stretch
662     {
663         checkedListBox1.SetItemCheckState(3, CheckState.Checked);
664         checkedListBox1.SetItemCheckState(4, CheckState.Unchecked); // fit
665         checkedListBox1.SetItemCheckState(5,
666             CheckState.Unchecked); //1:1
667     }

```

```
662
663     if (checkedListBox1.SelectedIndex == 4) //fit
664     {
665         checkedListBox1.SetItemCheckState(4, CheckState.Checked);
666         checkedListBox1.SetItemCheckState(3, CheckState.Unchecked); // 7
667         stretch
668         checkedListBox1.SetItemCheckState(5,
669         CheckState.Unchecked); //1:1
670     }
671
672     if (checkedListBox1.SelectedIndex == 5) //1:1
673     {
674         checkedListBox1.SetItemCheckState(5, CheckState.Checked);
675         checkedListBox1.SetItemCheckState(3, CheckState.Unchecked); // 7
676         stretch
677         checkedListBox1.SetItemCheckState(4, CheckState.Unchecked); // 7
678         fit
679     }
680
681     if (checkedListBox1.GetItemChecked(3)) slides.imagefit = 0;
682     if (checkedListBox1.GetItemChecked(4)) slides.imagefit = 1;
683     if (checkedListBox1.GetItemChecked(5)) slides.imagefit = 2;
684
685     if (checkedListBox1.GetItemChecked(6)) // no border fullscreen
686     {
687         slides.FormBorderStyle = FormBorderStyle.None;
688         slides.WindowState = FormWindowState.Maximized;
689     }
690     else
691     {
692         slides.FormBorderStyle = FormBorderStyle.Sizable;
693         slides.WindowState = FormWindowState.Normal;
694     }
695
696     if (checkedListBox1.GetItemChecked(7)) // Random order
697     {
698         slides.random = true;
699         checkedListBox1.SetItemCheckState(7, CheckState.Checked);
700     }
701     else
702     {
703         slides.random = false;
704         slides.repeatrand = false;
705         checkedListBox1.SetItemCheckState(8, CheckState.Unchecked);
706         checkedListBox1.SetItemCheckState(7, CheckState.Unchecked);
707     }
708
709     if (checkedListBox1.GetItemChecked(8)) // Random order w
710     repetition
711     {
712         slides.random = true;
713         slides.repeatrand = true;
714         checkedListBox1.SetItemCheckState(7, CheckState.Checked);
715         checkedListBox1.SetItemCheckState(8, CheckState.Checked);
716     }
717     else
```

```
713     {
714         slides.repeatrand = false;
715         checkedListBox1.SetItemCheckState(8, CheckState.Unchecked);
716     }
717
718     if (checkedListBox1.GetItemChecked(9) && listBox1.Items.Count > 0) // Start from selected
719     {
720         slides.startat = listBox1.SelectedIndex;
721     }
722     else
723     {
724         slides.startat = 0;
725         listBox1.SelectedIndex = 0;
726     }
727
728     if (checkedListBox1.GetItemChecked(10)) // Stop from sleeping/IDLE
729     {
730         slides.prevent_idle = true;
731     }
732     else
733     {
734         slides.prevent_idle = false;
735     }
736
737     if (checkedListBox1.GetItemChecked(11)) // play Guide audio
738     {
739         playguide = true;
740     }
741     else
742     {
743         playguide = false;
744         stopMp3();
745     }
746
747     slides.vol = (int)volume;
748
749 }
750
751 private void imageTimer_KeyPress(object sender, KeyPressEventArgs e)
752 {
753     if (e.KeyChar == (char)Keys.Return || e.KeyChar == (char)Keys.Enter)
754     {
755         int t = time;
756         int ts = time;
757         if (Int32.TryParse(imageTimer.Text, out ts))
758             if (ts >= 300)
759                 t = Int32.Parse(imageTimer.Text);
760         time = t;
761         imageTimer.Text = time.ToString();
762         e.Handled = true;
763         this.ActiveControl = null;
764     }
765 }
766
```

```
767     private void Form1_FormClosing(object sender, FormClosingEventArgs e)
768     {
769         UpdateSettings(Application.StartupPath + "\\settings.ini");
770     }
771
772     private void textBox_song_Enter(object sender, EventArgs e)
773     {
774         this.ActiveControl = null;
775     }
776
777     private void button_selectsong_Click(object sender, EventArgs e)
778     {
779         Stream myStream;
780         OpenFileDialog thisDialog = new OpenFileDialog();
781         thisDialog.Filter = "MP3 files (*.mp3)|*.mp3";
782         thisDialog.FilterIndex = 1;
783         thisDialog.RestoreDirectory = false;
784         thisDialog.Multiselect = false;
785         thisDialog.Title = "Please Select Desired MP3 File";
786
787         if (thisDialog.ShowDialog() == DialogResult.OK)
788         {
789             String file = thisDialog.FileName;
790             try
791             {
792                 if ((myStream = thisDialog.OpenFile()) != null)
793                 {
794                     using (myStream)
795                     {
796                         songtoplay = file;
797                         textBox_song.Text = songtoplay;
798                     }
799                 }
800             }
801             catch (Exception ex)
802             {
803                 MessageBox.Show("Error: Could not read file. Error: " +
804                     ex.Message);
805             }
806         }
807     }
808
809     private void button_savelist_Click(object sender, EventArgs e)
810     {
811         if (imglist.Count <= 0)
812         {
813             MessageBox.Show("You cannot save an empty list. Sorry.", "List
814                 is empty", MessageBoxButtons.OK, MessageBoxIcon.Warning);
815             return;
816         }
817         SaveFileDialog thisDialog = new SaveFileDialog();
818         thisDialog.Filter = "GIFSlider files (*.gs)|*.gs";
819         thisDialog.FilterIndex = 1;
820         thisDialog.RestoreDirectory = true;
```



```
821         thisDialog.Title = "Save as ... ";
822
823         if (thisDialog.ShowDialog() == DialogResult.OK)
824         {
825             try
826             {
827                 using (StreamWriter outputFile = new StreamWriter      ↗
828                     (thisDialog.FileName))
829                 {
830                     foreach (var file in imglist)
831                         outputFile.WriteLine(file);
832                 }
833             }
834             catch (Exception ex)
835             {
836                 MessageBox.Show("Error: Could not write file. Error: " + ↗
837                     ex.Message);
838             }
839         }
840
841         private void button_help_Click(object sender, EventArgs e)
842         {
843             stopMp3();
844
845             if(playguide)
846                 if (File.Exists(Application.StartupPath + "\\Guide.mp3"))
847                     playMp3(Application.StartupPath + "\\Guide.mp3", (int) ↗
848                         volume, false);
849
850                 MessageBox.Show("HELP ? Did I hear a call for help? \r\n" +
851                     "Hello, friend ! Call me a MessageBoxInformationGuide™. I ↗
852                     will be your guide for today (or any time you click that ↗
853                     button). \r\n" +
854                     "I am very pleased to introduce you to this beautiful ↗
855                     piece of application. It's an app that makes slideshows! " ↗
856                     +
857                     "Not usual slideshows, but very - to the detail - ↗
858                     adjustable slideshows! And you are about to experience the ↗
859                     magic! \r\n" +
860                     "First, I would suggest clicking the '+' button on the ↗
861                     right side just below the blue-ish listbox. Or you can ↗
862                     also " +
863                     "drop your files onto that listbox, that works too! \r\n" ↗
864                     +
865                     "Your next step should probably be looking to the left and ↗
866                     adjusting the s*BEEP*t out of your slideshow. \r\n\r\n " ↗
867                     +
868                     "* OMG, JERRY!! THIS IS A FAMILY FRIENDLY SHOW! WE HAD TO ↗
869                     BEEP YOU OUT!!! * \r\n\r\n" +
870                     "Sorry for that there. Didn't mean to be rude. Let's ↗
871                     continue our journey. Since you have probably set ↗
872                     everything up that you" +
873                     " needed, you are ready to set up the timer. In the middle ↗
874                     below the preview box. What that does is that it let's ↗
875                     you control " +
```

```

858         "for how long every image stays up before changing to the
            next one. Set up your desired delay and press Enter to
            confirm your " +
859         "sweet numbers. Don't forget that it's in milliseconds and
            we won't allow less than 300. And now, you are ready " +
860         "to start your slideshow by pressing the button
            'Slideshow!'. \r\n" +
861         "\r\nBut keep in mind that you can control the slideshow
            by arrow keys and exit by ESC in any setup.\r\n\r\n" +
862         "I hope that helped you. You can call me back any time you
            please. MessageBoxInformationGuide™ out. PEACE!",
863         "A friend needs help!", MessageBoxButtons.OK,
            MessageBoxIcon.None);
864     }
865
866     private void checkBox_loopmp3_CheckedChanged(object sender, EventArgs e)
867     {
868         slides.loopsong = checkBox_loopmp3.Checked;
869     }
870
871     private void trackBar_volume_ValueChanged(object sender, EventArgs e)
872     {
873         volume = trackBar_volume.Value;
874         label_vol.Text = volume.ToString();
875     }
876
877     private void checkBox_associate_CheckedChanged(object sender,
            EventArgs e)
878     {
879         if (checkBox_associate.Checked)
880             SetAssociation(".gs", "GIFSlider_File",
            Application.ExecutablePath, "GIFSlider File");
881         else
882         {
883             Registry.ClassesRoot.DeleteSubKeyTree("GIFSlider_File");
884             SHChangeNotify(0x08000000, 0x0000, IntPtr.Zero, IntPtr.Zero);
885         }
886     }
887
888     private void checkBox_associate_MouseHover(object sender, EventArgs e)
889     {
890         Point pos = this.PointToClient(MousePosition);
891         tip.Show("You need to start this application under Admin
            privileges to do this", this, pos.X, pos.Y, 2000);
892     }
893
894     private Point fit(Image origin, int wi, int he)
895     {
896         int sourceWidth = origin.Width;
897         int sourceHeight = origin.Height;
898         double destX = 0;
899         double destY = 0;
900
901         double nScale = 0;
902         double nScaleW = 0;

```

```
903         double nScaleH = 0;
904
905         nScaleW = ((double)wi / (double)sourceWidth);
906         nScaleH = ((double)he / (double)sourceHeight);
907         nScale = Math.Min(nScaleH, nScaleW);
908         destY = (he - sourceHeight * nScale) / 2;
909         destX = (wi - sourceWidth * nScale) / 2;
910
911         int destWidth = (int)Math.Round(sourceWidth * nScale);
912         int destHeight = (int)Math.Round(sourceHeight * nScale);
913         return new Point(destWidth, destHeight);
914     }
915
916     private void playMp3(string file, int vol, bool repeat)
917     {
918         if (!File.Exists(file))
919             return;
920
921         wplayer.URL = file;
922         wplayer.settings.volume = vol;
923
924         if (repeat)
925             wplayer.settings.setMode("loop", true);
926         else
927             wplayer.settings.setMode("loop", false);
928
929         wplayer.controls.play();
930     }
931
932     private void stopMp3()
933     {
934         wplayer.controls.stop();
935     }
936 }
937 }
938
```