```
1   using System;
2   using System.Drawing;
3   using System.IO.Ports;
4   using System.Text;
5   using System.Windows.Forms;
6   using MySql.Data.MySqlClient;
7   using ZedGraph;
8
9   namespace CDH_Controller
10  {
11      public partial class Form1 : Form
12      {
13          SerialPortController m_comPort = new SerialPortController(115200);
14          string m_portStr = "COM4";
15          StringBuilder str = new StringBuilder();
16          MySqlConnection conDatabase; // The "global" instance of MySqlConnection
17          int currScore;
18          int game = 0;
19
20          #region Form1 events
21
22          public Form1()
23          {
24              InitializeComponent();
25          }
26
27          private void Form1_Load(object sender, EventArgs e)
28          {
29              lbData.Items.Add("Opening connections...");
30
31              OpenCom();
32
33              OpenSql();
34
35              lbData.Items.Add("Done initializing. Check above for errors.");
36              lbData.Items.Add("-----------------------------------");
37
38              RefreshPlayers();
39
40              InitGraph(zgGraph);
41              SetSize();
42          }
43
```

```
44          private void Form1_FormClosed(object sender, FormClosedEventArgs e)
45          {
46              try
47              {
48                  conDatabase.Close();
49                  m_comPort.CloseConnection();
50              }
51              catch { }
52          }
53
54          private void Form1_Resize(object sender, EventArgs e)
55          {
56              SetSize();
57          }
58
59          #endregion
60
61          #region Connections
62
63          /// <summary>
64          /// Opens the COM port
65          /// </summary>
66          private void OpenCom()
67          {
68              try
69              {
70                  // Open the connection, using the global SerialPortController, m_comPort
71                  m_comPort.OpenConnection(m_portStr, new System.IO.Ports.SerialDataReceivedEventHandler(port_DataReceived));
72
73                  // Add a message to the log
74                  lbData.Items.Add(string.Format("{0} has been opened", m_portStr));
75
76                  // Enable buttons that should only be used if the COM port has been opened
77                  btnSend.Enabled = true;
78                  btnGame.Enabled = true;
79              }
80              catch // If an error occures, show a message in the log
81              {
82                  lbData.Items.Add(string.Format("Connecting to {0} failed!", m_portStr));
83              }
84          }
85
86          /// <summary>
```

```
87          /// Connects to the MySQL server
88          /// </summary>
89          private void OpenSql()
90          {
91              try // try-catch makes sure the program doesn't crash in case connecting failed
92              {
93                  // Initialize the instance of the MySqlConnection class.
94                  conDatabase = new MySqlConnection("server=localhost;" + "database=test;" + "uid=root;");
95                  // Open the connection
96                  conDatabase.Open();
97                  // In case all goes well, add a message to the log
98                  lbData.Items.Add("SQL connection established");
99                  // ..and enable btnAddPlayer, allowing the user to add players to the database
100                 btnAddPlayer.Enabled = true;
101             }
102             catch (Exception ex)
103             {
104                 // If something went wrong while trying to connect, show the error message in the log
105                 lbData.Items.Add(string.Format("SQL connection failed: {0}", ex.Message));
106             }
107         }
108
109         private void cbCOM_SelectedIndexChanged(object sender, EventArgs e)
110         {
111             m_portStr = cbCOM.Text;
112         }
113
114         private void btnReconnectCOM_Click(object sender, EventArgs e)
115         {
116             OpenCom();
117         }
118
119         private void btnReconnectSQL_Click(object sender, EventArgs e)
120         {
121             OpenSql();
122         }
123
124         #endregion
125
126         #region Score & game management
127
128         /// <summary>
129         /// When clicked, a "switch game"-command is sent to the master
```

```
130         /// </summary>
131         private void btnGame_Click(object sender, EventArgs e)
132         {
133             // The global variable game is set to the selected game in cbGame.
134             // The index of cbGame starts from 0, so add 1
135             game = cbGame.SelectedIndex + 1;
136             string gameStr;
137 
138             // If games with no. > 99 should be supported, a change in the master-software should be made
139             if (game > 99)
140                 throw new Exception("Games with no. > 99 is currently not supported!!");
141 
142             // If the game no. is < 10, add a leading 0
143             if (game < 10)
144                 gameStr = string.Format("G0{0}", game);
145             else
146                 gameStr = string.Format("G{0}", game);
147 
148             // Send the command
149             m_comPort.SendString(gameStr);
150             addToLb(string.Format("Sent: {0}", gameStr));
151         }
152 
153         /// <summary>
154         /// Fetches the Id of a player
155         /// </summary>
156         /// <param name="name">The name of the player</param>
157         /// <returns>The Id of the player</returns>
158         private int getPlayerId(string name)
159         {
160             try
161             {
162                 // Create an instance of the MySqlCommand class
163                 MySqlCommand cmd = conDatabase.CreateCommand();
164                 // Build the sql command
165                 cmd.CommandText = string.Format("SELECT Id FROM players WHERE Name='{0}'", name);
166                 // Create an instance of the MySqlDataReader class
167                 MySqlDataReader rdr = cmd.ExecuteReader();
168                 // Read the first row and save the result in the int id
169                 rdr.Read();
170                 int id = rdr.GetInt32(0);
171                 // Close the reader and return the id
172                 rdr.Close();
```

```
173                 return id;
174             }
175             catch
176             {
177                 // In case something goes wrong, return a 0
178                 return 0;
179             }
180         }
181 
182         /// <summary>
183         /// Saves the score of a person playing a game
184         /// NOTE: Uses the global variable "game"
185         /// </summary>
186         /// <param name="score">The score</param>
187         /// <param name="name">The name of the player</param>
188         private void SaveScore(int score, string name)
189         {
190             // If the connection is opened...
191             if (conDatabase.State == System.Data.ConnectionState.Open)
192             {
193                 // Assemble the SQL-command
194                 // string.format combines a string with a number of substrings
195                 MySqlCommand cmd = new MySqlCommand(
196                         string.Format("INSERT INTO scores (Score, PlayerId, GameId) VALUES ({0}, {1}, {2})", score,
197 getPlayerId(lbPlayers.SelectedItem.ToString()), game),
198                         conDatabase);
199                 // Execute the command, inserting the score
200                 cmd.ExecuteNonQuery();
201             }
202         }
203 
204         #endregion
205 
206         #region Graphing
207 
208         /// <summary>
209         /// Collects scores from the database related to the game w/ ID cbGame.SelectedIndex+1 and the player selected in lbPlayers
210         /// </summary>
211         private void btnGraph_Click(object sender, EventArgs e)
212         {
213             // Create a new instance of the PointParList class
214             PointPairList list = new PointPairList();
215 
```

```
216             //Create the MySqlCommand and start the DataReader
217             MySqlCommand cmd = conDatabase.CreateCommand();
218             cmd.CommandText = string.Format("SELECT Score FROM scores WHERE PlayerId={0} AND GameId={1} ORDER BY Id",
219 getPlayerId(lbPlayers.SelectedItem.ToString()), cbGame.SelectedIndex+1);
220             MySqlDataReader rdr = cmd.ExecuteReader();
221 
222             // While there's still new data to read, add the data to the PointParList
223             int i = 0;
224             while (rdr.Read())
225             {
226                 list.Add(i, rdr.GetInt32(0));
227                 i++;
228             }
229             rdr.Close();
230 
231             // Finally, put the data in the list on the graph
232             AddToGraph(zgGraph, list, string.Format("{0}'s scores", lbPlayers.SelectedItem.ToString()), colorDialog.Color,
233 SymbolType.Diamond);
234         }
235 
236         /// <summary>
237         /// Adds a PointParList of data to the zgc graph
238         /// </summary>
239         /// <param name="zgc">The graph to add the data to</param>
240         /// <param name="list">The list of data to add</param>
241         /// <param name="title">The title of the dataset</param>
242         /// <param name="color">The color of the dataset</param>
243         /// <param name="symbol">The symbol used to illustrate the dataset</param>
244         private void AddToGraph(ZedGraphControl zgc, PointPairList list, string title, Color color, SymbolType symbol)
245         {
246             // Create an instance of the GraphPane class, from the graph (zgc)
247             GraphPane myPane = zgc.GraphPane;
248 
249             // Add a curve to the graph with the specified title, data, color and symbol
250             LineItem curve = myPane.AddCurve(title, list, color, symbol);
251 
252             //curve.Symbol.Fill = new Fill(Color.White);
253             // Fill the background with a slightly golden gradient
254             myPane.Chart.Fill = new Fill(Color.White, Color.LightGoldenrodYellow, 45F);
255 
256             // Fill the pane background with a color gradient
257             //myPane.Fill = new Fill(Color.White, Color.FromArgb(220, 220, 255), 45F);
258 
```

```
259             // Refresh the graph
260             zgc.AxisChange();
261             zgc.Refresh();
262         }
263
264         /// <summary>
265         /// Init the graph window with default values
266         /// </summary>
267         /// <param name="zgc">The graphwindow to init</param>
268         private void InitGraph(ZedGraphControl zgc)
269         {
270             GraphPane myPane = zgc.GraphPane;
271
272             // Set the titles and axis labels
273             myPane.Title.Text = "Scores";
274             myPane.XAxis.Title.Text = "Spil nr.";
275             myPane.YAxis.Title.Text = "Score";
276         }
277
278         /// <summary>
279         /// Resizes the graphwindow to fit the fom
280         /// </summary>
281         private void SetSize()
282         {
283             zgGraph.Location = new Point(10, 260);
284             // Leave a small margin around the outside of the control
285             zgGraph.Size = new Size(this.ClientRectangle.Width - (30 + lbPlayers.Size.Width), this.ClientRectangle.Height - 270);
286         }
287
288         private void btnSetColor_Click(object sender, EventArgs e)
289         {
290             colorDialog.ShowDialog();
291             btnSetColor.BackColor = colorDialog.Color;
292         }
293
294         private void btnResetGraph_Click(object sender, EventArgs e)
295         {
296             // Goes here
297             zgGraph.GraphPane.CurveList.Clear();
298             zgGraph.Refresh();
299         }
300
301         #endregion
```

```
302
303         #region Serial communication
304         private void addToLb(string str)
305         {
306             Invoke(new MethodInvoker(
307                             delegate { lbData.Items.Add(string.Format("{0}", str));
308                                        lbData.SelectedIndex = lbData.Items.Count - 1;
309                                        lbData.SelectedIndex = -1; }
310                             ));
311         }
312
313         /// <summary>
314         /// Triggered when data is received on the COM-port
315         /// </summary>
316         private void port_DataReceived(object sender, SerialDataReceivedEventArgs e)
317         {
318             // Collect the message waiting in the input buffer
319             string inp = m_comPort.port.ReadExisting();
320
321             // The message is appended to the stringbuilder str
322             str.Append(inp);
323
324             // If there's a new line character, the message is complete, and we continue processing
325             // if not, the semi-completed message will wait until the next part of the message is received
326             if (inp.Contains("\n"))
327             {
328                 // The message is added to the log
329                 addToLb("Received: " + str.ToString());
330
331                 // If the message contains "SCORE:", its a score-entry, that should be added to the database
332                 if (str.ToString().Contains("SCORE:"))
333                 {
334                     // Catch the actual score (int), and cut the "SCORE:"-part off
335                     currScore = Convert.ToInt32(str.ToString().Split(':')[1]);
336
337                     // Save the score to the database
338                     SaveScore(currScore, lbPlayers.SelectedItem.ToString());
339                 }
340                 // If the message contains "STATUS", its a STATUS-request from the master
341                 else if (str.ToString().Contains("STATUS"))
342                 {
343                     // This message means the master is trying to verify the connection - we reply with "OK"
344                     m_comPort.SendString("OK\n");
```

```
345
346                 // And lets add the data to the log
347                 addToLb("Sent: OK");
348                 addToLb("-Connection to Master verified-");
349             }
350
351             // Since the message is over and processed, clear the stringbuilder
352             str.Clear();
353         }
354     }
355
356     private void btnSend_Click(object sender, EventArgs e)
357     {
358         m_comPort.SendString(tbSend.Text);
359         addToLb(string.Format("Sent: {0}", tbSend.Text));
360         tbSend.Text = "";
361         tbSend.Focus();
362     }
363     #endregion
364
365     #region Player-management
366
367     /// <summary>
368     /// Fetches players from the SQL-server and adds them to lbPlayers
369     /// </summary>
370     private void RefreshPlayers()
371     {
372         try
373         {
374             // Clear lbPlayers, first
375             lbPlayers.Items.Clear();
376             // Initialize and set the command
377             MySqlCommand cmd = conDatabase.CreateCommand();
378             cmd.CommandText = "SELECT Name FROM players ORDER BY Id";
379             // Start the DataReader
380             MySqlDataReader rdr = cmd.ExecuteReader();
381             while (rdr.Read()) // While there's more data to read
382             {
383                 // Add the string in column 0 of the current row to lbPlayers
384                 lbPlayers.Items.Add(rdr.GetString(0));
385             }
386             // Close the reader
387             rdr.Close();
```

```
388             }
389             catch { } // If something goes wrong nothing currently happens. Just ignore the error.
390         }
391
392         /// <summary>
393         /// Handles a click on btnAddPlayer
394         /// </summary>
395         private void btnAddPlayer_Click(object sender, EventArgs e)
396         {
397             // If the MySQL-database connection has been opened...
398             if (conDatabase.State == System.Data.ConnectionState.Open)
399             {
400                 // Initialize the command to insert players into the database
401                 MySqlCommand cmd = new MySqlCommand(
402                         string.Format("INSERT INTO players (Name) VALUES ('{0}')", tbPlayername.Text),
403                         conDatabase);
404                 // Execute the command, thereby inserting the player
405                 cmd.ExecuteNonQuery();
406                 // Refresh lbPlayers
407                 RefreshPlayers();
408
409                 // Empty tbPlayername and get ready for a new entry
410                 tbPlayername.Text = "";
411                 tbPlayername.Focus();
412             }
413         }
414
415         private void tbPlayername_Enter(object sender, EventArgs e)
416         {
417             tbPlayername.Text = "";
418         }
419
420         private void lbPlayers_SelectedIndexChanged(object sender, EventArgs e)
421         {
422             btnGraph.Enabled = true;
423             labelPlayer.Text = string.Format("Spiller: {0}", lbPlayers.SelectedItem.ToString());
424         }
425
426         #endregion
427     }
428 }
429
430
```