using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BracketsAndVenues

{

class Team

{

public string Name{ get; set;}

public int Seed {get; set;}

public int Wins{get; private set;}

public bool isTop { get; set; }

public Team(string name, int seed, int wins)

{

Name = name;

Seed = seed;

Wins = wins;

}

public void Win()

{

Wins++;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BracketsAndVenues

{

class Bracket

{

public List<Team> teams { get; set; }

public bool IsDirty { get; set; }

public List<Team[]> matchups { get; set; }

public Bracket(List<Team> tList)

{

teams = tList;

IsDirty = false;

matchups = new List<Team[]>();

//LINQ to sort list by seed value

var seedSortedList =

(from t in teams

select t).OrderBy(x => x.Seed);

List<Team> sorted = seedSortedList.ToList();

//use the sorted list to pair every team with its nearest seed neighbor

for(int i = 0; i < sorted.Count - 1; i +=2)

{

matchups.Add(new Team[] { sorted.ElementAt(i), sorted.ElementAt(i + 1) });

}

}

public Bracket()

{

teams = new List<Team>();

IsDirty = false;

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace BracketsAndVenues

{

public partial class Form1 : Form

{

private Bracket mainBracket { get; set; }

private string FilePath { get; set; }

private List<int[]> TextPositions { get; set; }

public Form1()

{

InitializeComponent();

mainBracket = new Bracket();

TextPositions = new List<int[]>();

}

//load in team file. right now it is plain text. this is subject to change

private void loadToolStripMenuItem\_Click(object sender, EventArgs e)

{

OpenFileDialog load = new OpenFileDialog();

load.Filter = "Text Files (\*.txt) | \*.txt";

load.ShowDialog();

if (load.FileName != "")

{

StreamReader sr = new StreamReader(load.FileName);

StringBuilder sb = new StringBuilder(sr.ReadToEnd());

sr.Close();

//parse the file

string[] lines = sb.ToString().Split(new string[] { Environment.NewLine }, StringSplitOptions.RemoveEmptyEntries);

List<Team> teams = new List<Team>();

for (int i = 0; i < lines.Length; i++)

{

string[] line = lines[i].Split(',');

string name = line[0];

int seed = Int32.Parse(line[1]);

int wins = Int32.Parse(line[2]);

teams.Add(new Team(name, seed, wins));

}

mainBracket = new Bracket(teams);

saveToolStripMenuItem.Enabled = true;

FilePath = load.FileName;

int numberOfTeams = lines.Length;

//load the appropriate bracket image based on number of teams

switch (numberOfTeams)

{

case 4: pictureBox1.ImageLocation = Environment.CurrentDirectory + "\\4\_team\_bracket.png"; break;

case 8: pictureBox1.ImageLocation = Environment.CurrentDirectory + "\\8\_team\_bracket.png"; break;

case 16: pictureBox1.ImageLocation = Environment.CurrentDirectory + "\\16\_team\_bracket.png"; break;

default: pictureBox1.ImageLocation = Environment.CurrentDirectory + "\\32\_team\_bracket.png"; break;

}

pictureBox1.Refresh();

//all positions are stored in def files

string positionPath;

switch (numberOfTeams)

{

case 4: positionPath = Environment.CurrentDirectory + "\\4\_locations.txt"; break;

case 8: positionPath = Environment.CurrentDirectory + "\\8\_locations.txt"; break;

case 16: positionPath = Environment.CurrentDirectory + "\\16\_locations.txt"; break;

default: positionPath = Environment.CurrentDirectory + "\\32\_locations.txt"; break;

}

//parse the def file

sr = new StreamReader(positionPath);

sb = new StringBuilder(sr.ReadToEnd());

string[] posLines = sb.ToString().Split(new string[] { Environment.NewLine }, StringSplitOptions.RemoveEmptyEntries);

for (int i = 0; i < posLines.Length; i++)

{

string[] halfLine = posLines[i].Split(':');

string[] coords = halfLine[1].Split(',');

TextPositions.Add(new int[] { Int32.Parse(coords[0]) + 147, Int32.Parse(coords[1]) + 45 });

}

//draw names on bracket

Graphics g = Graphics.FromImage(pictureBox1.Image);

using (Font myFont = new Font("Arial", 14))

{

for (int i = 0; i < TextPositions.Count; i++)

{

if (mainBracket.teams.Count > i )

{

g.DrawString(mainBracket.teams.ElementAt(i).Name, myFont, Brushes.Black, new Point(TextPositions.ElementAt(i)[0], TextPositions.ElementAt(i)[1]));

}

}

}

}

}

private void pictureBox1\_Paint(object sender, PaintEventArgs e)

{

}

private void saveToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (FilePath == "")

{

Save(true);

}

else

Save(false);

}

//Exit button Clicked

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

//check if any changes have been made, offer to save if they have

if (mainBracket.IsDirty == false)

{

this.Close();

}

else

{

DialogResult saveOption = MessageBox.Show("Do you want to save first?", "Exit", MessageBoxButtons.YesNoCancel);

if (saveOption == DialogResult.Yes)

{

if (FilePath != "")

{

if (!Save(false))

this.Close();

}

else

{

Save(true);

this.Close();

}

}

else if(saveOption == DialogResult.No)

{

this.Close();

}

}

}

//save function returns true if canceled, false otherwise

private bool Save(bool saveAs)

{

if (mainBracket.teams.Count > 0)

{

if (saveAs)

{

SaveFileDialog save = new SaveFileDialog();

save.Filter = "Text File (\*.txt) | \*.txt";

save.ShowDialog();

FilePath = save.FileName;

}

StringBuilder sb = new StringBuilder();

foreach (Team t in mainBracket.teams)

{

string line = t.Name + "," + t.Seed.ToString() + "," + t.Wins.ToString();

sb.AppendLine(line);

}

if(FilePath == "")

{

return true;

}

try

{

File.WriteAllText(FilePath, sb.ToString());

}

catch

{

MessageBox.Show("File cannot be overwritten because it is in use by another process");

}

return false;

}

return true;

}

//save as option clicked

private void saveAsToolStripMenuItem\_Click(object sender, EventArgs e)

{

Save(true);

}

}

}