GUI Programming
Program #1 – **Door Prize**
Due Mon, Oct. 1
100 Points

Overview

You are to write a program using C# for picking a door prize winner at the Computer Seminar. The program will allow a user to enter a listing of names and then choose 4-6 names from the list to "compete" against each other. Some type of animation with sound effects will be used in a competition to determine a winner. This program should be similar in nature to the Battleship and Boats programs which are currently being used for seminar.

This program will improve your ability to write C# programs and give you the skills to produce top-quality animation. These skills will be in high demand in your cap-stone class, the Software Development Project.

The most entertaining programs from this assignment will be used in the Computer Seminar next semester!

Requirements

The following specifications must be met in order to receive all 100 points:

1. The program should allow the user to enter a list of names into a text box (1 unique name per line- your program does *not* need to enforce this) and save them to disk. The names may have already been entered into a text file using Notepad or some other program, so your program should also be able to load a text file from disk. A .txt default extension should be used for opening and saving files, and the user should also be able to select "All files (*.*)" when opening a file.

2. The user should be able to undo, redo, cut, copy, and paste into the text box.

3. At least 4 random names need to be selected from the name list to participate in the competition. Identify these names in a dialog box before continuing on to the competition.

4. The competition can be of your own choosing, but it should take on the average 5 – 60 seconds to complete (from the beginning of the competition to the display of the winner). There should be no need for operator input once the competition has begun. The winner should be randomly chosen from the competitors. Be as creative as you can with the competition.

5. The competition must involve animation using the smooth animation algorithm given in class. The animation could be as simple as a boat that travels from the left side of the screen to the right, but a more elaborate animation will distinguish your program from the rest. Although not required, it would be nice to have a random event take place that "kills" one of the competitors. For example, have a shark eat one of the boats.

6. When the competition is complete, the winner should be clearly identified. After the winner is identified, the user should be able to start a new competition or enter new names, load a new file, etc.

7. The user should be able to close the application during the competition. He/she **should not** be forced to wait until the competition has completed before being able to close the application.

8. Your program should have its own custom icon.

9. There should be an About dialog box that identifies you as the programmer.

10. Your program should be extremely robust- I'll try my best to get it to crash or hang! That means checking the name list for at least 5 names before trying to obtain 5 different names from the list, guarding against crashes when the user tries to close the program in the middle of the competition, etc.

11. Your program must play at least one sound file (WAV, MIDI, or MP3) during the competition.

12. Your program should be very forgiving. That means if the names have not been saved yet, your application should not quit before giving the user the chance to save the names. Study the Notepad application and mimic its responses to unsaved data.

Keep in mind that this program should not have an extremely complicated heuristic- a simple animation path from left to right or top to bottom is sufficient. Be creative, but don't be overly ambitious; don't run the risk of turning in a late program. **Don't waste a whole lot of time designing elaborate graphics or finding wave files.** Create a basic program that completes from start to finish, and add-on as time permits.

Grading

Make your application as intuitive as possible, use helpful user feedback, and make sure your animation is smooth and controlled.

All files required for execution (executable, sound files, bitmaps, etc.) of your program should be in the same directory.

Submit your program *before* class on the due date with **Easel** as a zip file. Zip up all your files together along with all your source code. I will not re-build your program from scratch, but I will look through your source code.

Everyone who competes in the games goes into strict training.
They do it to get a crown that will not last;
but we do it to get a crown that will last forever.

1 Corinthians 9:25

Implementation

Before writing the code for an object-oriented application you'll want to devote some time into deciding what objects your program will need. One necessary object for the Door Prize program will be for the contestants. You will need to create a player class that contains properties like the name and image for the player. You will use an array of player objects to represent the contestants who will compete for the door prize.

Below is the implementation for a Player class. Put your class in a file named Player.cs and add it to your project. You can declare an array of Players like so:

```
private Player[] players;
```

and allocate the players in the main form's constructor like this:

```
players = new Player[NUM_PLAYERS];    // NUM_PLAYERS is a constant.
```

Now you can create a player with happy1.png as an image (make sure the image is added to the project and the Build Action is set to Embedded Resource) and name him Bob:

```
players[0] = new Player(new Bitmap(GetType(), "happy1.png"));
players[0].Name = "Bob";
```

Bob's image could be drawn on a form or another bitmap:

```
g.DrawImage(players[0].Image, x, y);
```

or

```
players[0].Draw(g, x, y);  // Draws player's image on g at (x,y)
players[0].Draw(g);        // Draws player's image on g at current (x,y)
```

```
using System;
using System.Drawing;

namespace DoorPrize
{
        /// <summary>
        /// The Player class represents a single competitor.
        /// </summary>
        public class Player
        {
                private string name;        // Name of player
                private Image image;        // Image representing player
                private int x, y;           // Location of player on form
                private Bitmap workspace;   // Used as off-screen buffer for smooth anim


                public Player(Bitmap bitmap, int maxXIncrement)
                {
                        image = bitmap;
                        workspace = new Bitmap(image.Width + maxXIncrement, image.Height);
                }

                public string Name
                {
                        set { name = value;    }
                        get { return name;     }
                }
```

```
        public Image Image
        {
                get { return image;    }
        }

        public int X
        {
                set { x = value; }
                get { return x;  }
        }

        public int Y
        {
                set { y = value; }
                get { return y;  }
        }

        public void Draw(Graphics g)
        {
                g.DrawImage(image, X, Y);
        }

        public void Draw(Graphics g, int x, int y)
        {
                g.DrawImage(image, x, y);
        }

        public void Move(Graphics g, Image background, int incx)
        {
                // Move player from current location to next one using double buffering.

                Write your code here…

                // Update player's location
                x += incx;
        }
    }
}
```

Adding Sound

The appropriate use of sound in your program will make it much more exciting. You could use a short "BANG!" wave file that you play when your competitors collide, or maybe use a song wave file to play in the background while the competitors compete.

Sound files can be downloaded off the Internet or obtained from various applications on your computer. The most popular types of sound files are:

1. WAV files (.wav) – Windows specific file format capable of playing realistic sounds. Recorded with 8-bit or 16-bit sample resolutions, mono or stereo, and various sample rates. In general, the larger the file, the better it sounds.
2. MIDI files (.midi) – Stores each note's pitch, length, and volume in a musical score. Only used for music. Usually much smaller than WAV files, but lacks quality of sound that WAV files offer.
3. MP3 files (.mp3) – Compressed file format ideal for storing music. File sizes are generally smaller than WAV files.

There are two ways of playing sound files in a C# application: using a Windows Media Player COM component or using the PlaySound Win32 API function.

Using the Windows Media Player:

The Windows Media Player can play WAV, MIDI, and MP3 files. WMP must first be installed on the computer. The following instructions describe how to add the WMP COM component to your project:

1. Open your .NET project.
2. Select Tools → Choose Toolbox Items… from the menu.

3. Select the "COM Components" tab, check "Windows Media Player" from the list, and press OK. It will now appear as a component in the Toolbox.
4. Drop the media player component onto your form and change its Visible property to False so it won't be visible when running your program.
5. Place a WAV, MIDI, or MP3 file into the directory where your C# executable is located (project\bin\Debug folder). You will notice AxInterop.WMPLib.dll and Interop.WMPLib.dll in the same directory; they were placed there when you added the media player. If you expand "References" in the Solution Explorer, and you will also notice the addition of "AxWMPLib" and "WMPLib".

To open and play a sound file: `axWindowsMediaPlayer1.URL = "test.mp3";`

If you prefer to open the file and not play it until later, set the media player's autoStart property to false *before* you load the media file: `axWindowsMediaPlayer1.settings.autoStart = false;`

Once a file has been opened, it can be replayed using Play: `axWindowsMediaPlayer1.Ctlcontrols.play();`
You can stop the player by calling Stop: `axWindowsMediaPlayer1.Ctlcontrols.stop();`

Using PlaySound:

PlaySound API function is only capable of playing WAV files which is sufficient for your project. To use the PlaySound function, you must import the function from winmm.dll. Place the using line at the top of your class file:

```
using System.Runtime.InteropServices;  // for DllImport
```

In your class's method declaration section, add:

```
[DllImport("winmm.dll")]
private static extern int PlaySound(String filename, int handle, int mode);
```

Now place a WAV file in the same directory as your executable, and play it like so:

```
PlaySound("test.wav", 0, 0x0001);  // 0x0001 means to play asynchronously
```

Asynchronous play tells Windows to play the sound and immediately return control to the program. This enables the animation to continue while the sound is being played. To play a sound synchronously, change the last argument to 0x20000.

To stop a sound while playing: `PlaySound(null, 0, 0);`

You can use PlaySound to play a sound while the MediaPlayer is also playing a sound. Combining the two methods will allow your app to have a "BANG!" at the same time a song is playing.

Random Moves

To randomly select contestants and make the player move randomly, you will need to use the Random class. The following code shows how to get a random number from 1-10:

```
Random random = new Random();
int num = random.Next(10) + 1;
```

Each of the contestants should move a random amount throughout the race. You may want to experiment with different increments to see which one produces the best effect.