```
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;
using Microsoft.Win32;
using System.Media;

namespace RememberEachWord
{
    public partial class Form1 : Form
    {
        public int WordCount = 6;
        private int roundCounter = 0;
        private REWRound currentRound = null;
        private Font visFont = null;
        private Font unvFont = null;
        private Color visCol = Color.Black;
        private Color unvCol = Color.Gray;
        public static bool saved = true;
        private string filename = "";
        private bool secondScreen = false;
        private SoundPlayer sp;

        private string FileName
        {
            get
            {
                return filename;
            }
            set
            {
                filename = value;
                wf.Text = "Editor - " + filename;
            }
        }
        private bool Saved
        {
            get
            {
                return saved;
            }
            set
            {
                saved = value;
                if (saved)
                {
                    wf.Text = "Editor - " + filename;
                }
                else
                {
                    wf.Text = "Editor - " + filename + "*";
                }
            }
        }

        WordsForm wf = new WordsForm();
        WordsForm pf = new WordsForm();
        public Form1()
        {
            InitializeComponent();

            RegistryLoad();

            WordCount = wf.WordsCount;
            currentRound = new REWRound("current", WordCount);

            wf.Edit = false;
            wf.Move += wf_Move;
            wf.WordFieldChanged += wf_WordFieldChanged;
            wf.KeyDown += wf_KeyDown;
            wf.Show();
            wf.StoreOrigParameters();

            pf.Edit = false;
            pf.Text = "Show";
            pf.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Sizable;
            pf.BackColor = Color.Black;
            pf.KeyDown += pf_KeyDown;
            pf.Show();
            pf.StoreOrigParameters();

            timer1.Start();

            sp = new SoundPlayer(Properties.Resources.Ding);
        }

        private void RegistryLoad()
        {
            //------------------------  Загрузка из реестра!!!!   -----------------------
            RegistryKey currentUserKey = Registry.CurrentUser;
            RegistryKey softwareKey = currentUserKey.OpenSubKey("Software", true);
            RegistryKey subSoftwareKey = softwareKey.OpenSubKey("RememberEachWord", true);

            if (subSoftwareKey == null)
            {
                subSoftwareKey = softwareKey.CreateSubKey("RememberEachWord");

                subSoftwareKey.SetValue("VisibleFontName", "Impact");
                subSoftwareKey.SetValue("VisibleFontSize", 28.0f);
                subSoftwareKey.SetValue("VisibleFontStyle", (int)FontStyle.Bold);
                subSoftwareKey.SetValue("VisibleFontColor", Color.Black.ToArgb());

                subSoftwareKey.SetValue("UnvisibleFontName", "Impact");
                subSoftwareKey.SetValue("UnvisibleFontSize", 28.0f);
                subSoftwareKey.SetValue("UnvisibleFontStyle", (int)FontStyle.Regular);
                subSoftwareKey.SetValue("UnvisibleFontColor", Color.Gray.ToArgb());
            }

            string name = (string)subSoftwareKey.GetValue("VisibleFontName");
            float size = float.Parse(subSoftwareKey.GetValue("VisibleFontSize").ToString());
            FontStyle fs = (FontStyle)(int)subSoftwareKey.GetValue("VisibleFontStyle");
            Color cl = Color.FromArgb((int)subSoftwareKey.GetValue("VisibleFontColor"));
            visFont = new Font(name, size, fs);
            visCol = cl;

            name = (string)subSoftwareKey.GetValue("UnvisibleFontName");
            size = float.Parse(subSoftwareKey.GetValue("UnvisibleFontSize").ToString());
            fs = (FontStyle)(int)subSoftwareKey.GetValue("UnvisibleFontStyle");
            cl = Color.FromArgb((int)subSoftwareKey.GetValue("UnvisibleFontColor"));
            unvFont = new Font(name, size, fs);
            unvCol = cl;

            subSoftwareKey.Close();
            softwareKey.Close();

            wf.VisFont = visFont;
            wf.UnvFont = unvFont;
            wf.VisColor = visCol;
            wf.UnvColor = unvCol;
            pf.VisFont = visFont;
            pf.UnvFont = unvFont;
            pf.VisColor = visCol;
            pf.UnvColor = unvCol;
            //-------------------------------------------------------------------------
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (!saved)
            {
                System.Windows.Forms.DialogResult dr = MessageBox.Show("Save this project?", "Project is not ↙
    saved!", MessageBoxButtons.YesNoCancel);
```

```
                if (dr == System.Windows.Forms.DialogResult.Yes)
                {
                    saveGameToolStripMenuItem_Click(sender, e);
                }
                else if (dr == System.Windows.Forms.DialogResult.Cancel)
                {
                    e.Cancel = true;
                    return;
                }
            }

            RegistrySave();
        }

        private void RegistrySave()
        {
            RegistryKey currentUserKey = Registry.CurrentUser;
            RegistryKey softwareKey = currentUserKey.OpenSubKey("Software", true);
            RegistryKey subSoftwareKey = softwareKey.OpenSubKey("RememberEachWord", true);

            subSoftwareKey = softwareKey.CreateSubKey("RememberEachWord");

            subSoftwareKey.SetValue("VisibleFontName", visFont.Name);
            subSoftwareKey.SetValue("VisibleFontSize", visFont.Size);
            subSoftwareKey.SetValue("VisibleFontStyle", (int)visFont.Style);
            subSoftwareKey.SetValue("VisibleFontColor", visCol.ToArgb());

            subSoftwareKey.SetValue("UnvisibleFontName", unvFont.Name);
            subSoftwareKey.SetValue("UnvisibleFontSize", unvFont.Size);
            subSoftwareKey.SetValue("UnvisibleFontStyle", (int)unvFont.Style);
            subSoftwareKey.SetValue("UnvisibleFontColor", unvCol.ToArgb());

            subSoftwareKey.Close();
            softwareKey.Close();
        }

        void pf_KeyDown(object sender, KeyEventArgs e)
        {
            Form1_KeyDown(sender, e);
        }

        void wf_KeyDown(object sender, KeyEventArgs e)
        {
            Form1_KeyDown(sender, e);
        }

        void wf_WordFieldChanged(object sender, WordEditorFieldEventArgs e)
        {
            // Изменения в редакторе
            currentRound.SetAt(e.Text, e.State, e.Index);
            if (autosaveChangesToolStripMenuItem.Checked)
            {
                int sel = listBox1.SelectedIndex;

                if (sel >= 0)
                {
                    (listBox1.Items[sel] as REWRound).SetAt(e.Text, e.State, e.Index);
                }
            }

            RefreshWordForm();

            if (toolStripMenuItem4.Checked)
            {
                sp.Play();
            }

            Saved = false;
        }

        void wf_Move(object sender, EventArgs e)
        {
            Point loc = (sender as WordsForm).Location;
            this.Location = new Point(loc.X - this.Width, loc.Y);
        }

        private void newRoundToolStripMenuItem_Click(object sender, EventArgs e)
        {
            roundCounter++;
            REWRound rr = new REWRound("Round " + roundCounter.ToString(), WordCount);
            listBox1.Items.Add(rr);
            listBox1.SelectedIndex = listBox1.Items.Count - 1;

            Saved = false;
        }

        private void insertRoundToolStripMenuItem_Click(object sender, EventArgs e)
        {
            roundCounter++;
            REWRound rr = new REWRound("Round " + roundCounter.ToString(), WordCount);
            if (listBox1.SelectedIndex >= 0)
            {
                int sel = listBox1.SelectedIndex;
                listBox1.Items.Insert(sel, rr);
                listBox1.SelectedIndex = sel;
            }
            else
            {
                listBox1.Items.Add(rr);
                listBox1.SelectedIndex = listBox1.Items.Count - 1;
            }

            Saved = false;
        }

        private void deleteRoundToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (listBox1.SelectedIndex >= 0)
            {
                int sel = listBox1.SelectedIndex;
                listBox1.Items.RemoveAt(sel);
                if (sel >= listBox1.Items.Count)
                {
                    sel--;
                }
                if (sel >=0)
                {
                    listBox1.SelectedIndex = sel;
                }
            }

            Saved = false;
        }

        private void storeRoundToolStripMenuItem_Click(object sender, EventArgs e)
        {
            int sel = listBox1.SelectedIndex;

            if (sel >= 0)
            {
                //(listBox1.Items[sel] as REWRound).SetAt(e.Text, e.State, e.Index);
                for (int i = 0; i < WordCount; i++)
                {
                    (listBox1.Items[sel] as REWRound).SetAt(currentRound.GetTxtAt(i), currentRound.GetStateAt↙
    (i), i);
                }
            }
        }

        private void listBox1_MouseDoubleClick(object sender, MouseEventArgs e)
        {
            if (listBox1.SelectedIndex >= 0)
            {
                int sel = listBox1.SelectedIndex;
                RoundNameDialog rnd = new RoundNameDialog();
                rnd.RoundName = listBox1.Items[sel].ToString();
                if (rnd.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
```

```
                REWRound rr = (REWRound)listBox1.Items[sel];
                rr.RoundName = rnd.RoundName;
                listBox1.Items[sel] = rr;
            }

            Saved = false;
        }
    }


    // ------------------------ Drag & Drop --------------------------------
    private Rectangle dragBoxFromMouseDown;
    private int rowIndexFromMouseDown;
    private int rowIndexOfItemUnderMouseToDrop;

    private void listBox1_MouseMove(object sender, MouseEventArgs e)
    {
        if ((e.Button & MouseButtons.Left) == MouseButtons.Left)
        {
            // If the mouse moves outside the rectangle, start the drag.
            if (dragBoxFromMouseDown != Rectangle.Empty &&
                !dragBoxFromMouseDown.Contains(e.X, e.Y))
            {

                // Proceed with the drag and drop, passing in the list item.
                DragDropEffects dropEffect = listBox1.DoDragDrop(
                listBox1.Items[rowIndexFromMouseDown],
                DragDropEffects.Move);
            }
        }
    }

    private void listBox1_MouseDown(object sender, MouseEventArgs e)
    {
        // Get the index of the item the mouse is below.
        rowIndexFromMouseDown = listBox1.IndexFromPoint(e.X, e.Y);
        if (rowIndexFromMouseDown != -1)
        {
            // Remember the point where the mouse down occurred.
            // The DragSize indicates the size that the mouse can move
            // before a drag event should be started.
            Size dragSize = SystemInformation.DragSize;

            // Create a rectangle using the DragSize, with the mouse position being
            // at the center of the rectangle.
            dragBoxFromMouseDown = new Rectangle(new Point(e.X - (dragSize.Width / 2),
                                                           e.Y - (dragSize.Height / 2)),
                                dragSize);
        }
        else
            // Reset the rectangle if the mouse is not over an item in the ListBox.
            dragBoxFromMouseDown = Rectangle.Empty;
    }

    private void listBox1_DragOver(object sender, DragEventArgs e)
    {
        e.Effect = DragDropEffects.Move;
    }

    private void listBox1_DragDrop(object sender, DragEventArgs e)
    {
        // The mouse locations are relative to the screen, so they must be
        // converted to client coordinates.
        Point clientPoint = listBox1.PointToClient(new Point(e.X, e.Y));

        // Get the row index of the item the mouse is below.
        rowIndexOfItemUnderMouseToDrop = listBox1.IndexFromPoint(clientPoint.X, clientPoint.Y);
        if (rowIndexOfItemUnderMouseToDrop < 0) rowIndexOfItemUnderMouseToDrop = listBox1.Items.Count - 1↙
;
        // If the drag operation was a move then remove and insert the row.
        if (e.Effect == DragDropEffects.Move)
        {
            REWRound rowToMove = e.Data.GetData(
                typeof(REWRound)) as REWRound;
            listBox1.Items.RemoveAt(rowIndexFromMouseDown);
            listBox1.Items.Insert(rowIndexOfItemUnderMouseToDrop, rowToMove);
            listBox1.SelectedIndex = rowIndexOfItemUnderMouseToDrop;
        }

        Saved = false;
    }

    //-------------------------------------------------------------------------

    private void showHideToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (pf.Visible)
        {
            pf.Hide();
        }
        else
        {
            pf.Show();
        }
    }

    private void Form1_Move(object sender, EventArgs e)
    {
        wf.Location = new Point(this.Location.X + this.Width, this.Location.Y);
    }

    private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        int sel = listBox1.SelectedIndex;

        if (sel >= 0)
        {
            wf.Edit = true;

            // Загрузка данных раунда
            REWRound rr = listBox1.Items[sel] as REWRound;

            for (int i = 0; i < WordCount; i++)
            {
                currentRound.SetAt(rr.GetTxtAt(i), rr.GetStateAt(i), i);
            }
        }
        else
        {
            wf.Edit = false;

            // Очистка данных раунда
            for (int i = 0; i < WordCount; i++)
            {
                currentRound.SetAt("", false, i);
            }
        }

        RefreshEditor();
        RefreshWordForm();
    }

    private void RefreshEditor()
    {
        for (int i = 0; i < WordCount; i++)
        {
            wf.SetItemState(currentRound.GetStateAt(i), i);
            wf.SetItemText(currentRound.GetTxtAt(i), i);
        }
    }

    private void RefreshWordForm()
    {
        // RefreshWordForm()

        for (int i = 0; i < WordCount; i++)
        {
            pf.SetItemState(currentRound.GetStateAt(i), i);
            pf.SetItemText(currentRound.GetTxtAt(i), i);
        }
    }

    private void visibleFontToolStripMenuItem_Click(object sender, EventArgs e)
    {
        FontDialog fd = new FontDialog();
        fd.Font = visFont;
        fd.Color = visCol;
        fd.ShowColor = true;
        if (fd.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        {
            visFont = fd.Font;
            wf.VisFont = visFont;
            pf.VisFont = visFont;

            visCol = fd.Color;
            wf.VisColor = visCol;
            pf.VisColor = visCol;

            Saved = false;
        }
    }

    private void unvisibleFontToolStripMenuItem_Click(object sender, EventArgs e)
    {
        FontDialog fd = new FontDialog();
        fd.Font = unvFont;
        fd.Color = unvCol;
        fd.ShowColor = true;
        if (fd.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        {
            unvFont = fd.Font;
            wf.UnvFont = unvFont;
            pf.UnvFont = unvFont;

            unvCol = fd.Color;
            wf.UnvColor = unvCol;
            pf.UnvColor = unvCol;

            Saved = false;
        }
    }

    //_____________________ MENU File _____________________________________
    private void newGameToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (!saved)
        {
            System.Windows.Forms.DialogResult dr = MessageBox.Show("Save this project?", "Project is not ↙
saved!", MessageBoxButtons.YesNoCancel);
            if (dr == System.Windows.Forms.DialogResult.Yes)
            {
                saveGameToolStripMenuItem_Click(sender, e);
            }
            else if (dr == System.Windows.Forms.DialogResult.Cancel) return;
        }

        roundCounter = 0;
        listBox1.Items.Clear();
        wf.ClearForm();
        pf.ClearForm();
        wf.Edit = false;
        pf.Edit = false;

        //RegistryLoad();

        Saved = true;
    }

    private void saveGameToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if ((filename == "") || (!File.Exists(filename)))
        {
            saveGameAsToolStripMenuItem_Click(sender, e);
        }
        else
        {
            SaveProject();
        }
    }

    private void saveGameAsToolStripMenuItem_Click(object sender, EventArgs e)
    {
        SaveFileDialog sfd = new SaveFileDialog();
        sfd.Filter = "RemEWord files (*.rew)|*.rew";
        sfd.DefaultExt = "rew";
        sfd.AddExtension = true;
        if ((filename != "") && (File.Exists(filename)))
        {
            sfd.FileName = filename;
        }
        if (sfd.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        {
            FileName = sfd.FileName;
            SaveProject();
        }
    }

    private void openGameToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (!saved)
        {
            System.Windows.Forms.DialogResult dr = MessageBox.Show("Save this project?", "Project is not ↙
saved!", MessageBoxButtons.YesNoCancel);
            if (dr == System.Windows.Forms.DialogResult.Yes)
            {
                saveGameToolStripMenuItem_Click(sender, e);
            }
            else if (dr == System.Windows.Forms.DialogResult.Cancel) return;
        }

        OpenFileDialog ofd = new OpenFileDialog();
        ofd.Filter = "RemEWord files (*.rew)|*.rew";
        if (ofd.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        {
            FileName = ofd.FileName;
            OpenProject();
        }
    }

    private void reloadCurrentGameCtrlRToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if ((filename != "") && (File.Exists(filename)))
        {
            OpenProject();
        }
    }

    private void SaveProject()
    {
        ProjFileStream pfs = new ProjFileStream(filename, FileMode.Create, FileAccess.Write);

        WriteFont(visFont, visCol, pfs); // Видимый шрифт
        WriteFont(unvFont, unvCol, pfs); // Невидимый шрифт
        pfs.WriteInt(WordCount); // Количество слов. Должно совпадать при загрузке!
        pfs.WriteInt(listBox1.Items.Count); // Количество заданий
        for (int i = 0; i < listBox1.Items.Count; i++ )
        {
            REWRound.SaveRound((REWRound)listBox1.Items[i], pfs);
        }

        pfs.Close();

        Saved = true;
    }

    private void OpenProject()
    {
        listBox1.Items.Clear();
        wf.ClearForm();
        pf.ClearForm();

        ProjFileStream pfs = new ProjFileStream(filename, FileMode.Open, FileAccess.Read);

        // Видимый шрифт
        visFont = ReadFont(pfs, ref visCol);
        wf.VisFont = visFont;
        pf.VisFont = visFont;
        wf.VisColor = visCol;
        pf.VisColor = visCol;

        // Невидимый шрифт
        unvFont = ReadFont(pfs, ref unvCol);
        wf.UnvFont = unvFont;
        pf.UnvFont = unvFont;
        wf.UnvColor = unvCol;
        pf.UnvColor = unvCol;

        // Количество слов. Должно совпадать!
        int localWC = pfs.ReadInt();

        // Количество заданий
        int localCNT = pfs.ReadInt();
        roundCounter = localCNT;
        for (int i = 0; i < localCNT; i++)
        {
            listBox1.Items.Add(REWRound.LoadRound(pfs, localWC, WordCount));
        }

        if (listBox1.Items.Count > 0) listBox1.SelectedIndex = 0;

        pfs.Close();

        Saved = true;
    }

    private void WriteFont(Font fnt, Color cl, ProjFileStream pfs)
    {
        pfs.WriteString(fnt.Name);
        pfs.WriteFloat(fnt.Size);
        pfs.WriteInt((int)fnt.Style);
        pfs.WriteInt((int)fnt.Unit);
        pfs.WriteByte(fnt.GdiCharSet);
        pfs.WriteBool(fnt.GdiVerticalFont);
        pfs.WriteColor(cl);
    }

    private Font ReadFont(ProjFileStream pfs, ref Color col)
    {
        string name = pfs.ReadString();
        float size = pfs.ReadFloat();
        FontStyle fs = (FontStyle)pfs.ReadInt();
        GraphicsUnit gu = (GraphicsUnit)pfs.ReadInt();
        byte cs = pfs.ReadByte();
        bool vf = pfs.ReadBool();
        col = pfs.ReadColor();
        return new System.Drawing.Font(name, size, fs, gu, cs, vf);
    }
    //________________________________________________________________________

    //----------------------------  Hot Keys  --------------------------------
    private void Form1_KeyDown(object sender, KeyEventArgs e)
    {
        //int sel;
        //MessageBox.Show(e.KeyCode.ToString());
        if (e.Control)
        {
            switch (e.KeyCode)
            {
                case Keys.N:
                    newGameToolStripMenuItem_Click(sender, new EventArgs());
                    break;
                case Keys.S:
                    if (e.Shift)
                    {
                        saveGameAsToolStripMenuItem_Click(sender, new EventArgs());
                    }
                    else
                    {
                        saveGameToolStripMenuItem_Click(sender, new EventArgs());
                    }
                    break;
                case Keys.O:
                    openGameToolStripMenuItem_Click(sender, new EventArgs());
                    break;
                case Keys.R:
                    reloadCurrentGameCtrlRToolStripMenuItem_Click(sender, new EventArgs());
                    break;
                case Keys.Add:
                    newRoundToolStripMenuItem_Click(sender, new EventArgs());
                    break;
                case Keys.Insert:
                    insertRoundToolStripMenuItem_Click(sender, new EventArgs());
                    break;
                case Keys.Delete:
                    deleteRoundToolStripMenuItem_Click(sender, new EventArgs());
                    break;
                case Keys.M:
                    storeRoundToolStripMenuItem_Click(sender, new EventArgs());
                    break;
                case Keys.A:
                    autosaveChangesToolStripMenuItem.Checked = !autosaveChangesToolStripMenuItem.Checked;
                    string answ = (autosaveChangesToolStripMenuItem.Checked) ? "YES" : "NO";
                    MessageBox.Show("Autosave Round Parameter was switched to - " + answ);
                    break;
                case Keys.Enter:
                    toolStripMenuItem3.Checked = !toolStripMenuItem3.Checked;
                    break;
                case Keys.P:
                    toolStripMenuItem4.Checked = !toolStripMenuItem4.Checked;
                    if (toolStripMenuItem4.Checked)
                    {
                        MessageBox.Show("Sound is On");
                    }
                    else
                    {
                        MessageBox.Show("Sound is Off");
                    }
                    break;
            }
        }
        else
        {
            switch (e.KeyCode)
            {
                case Keys.Oemtilde:
                    if (pf.Visible)
                    {
                        pf.Hide();
                    }
                    else
                    {
                        pf.Show();
                    }
                    break;
                //case Keys.Up:
                //    sel = listBox1.SelectedIndex;
                //    if (sel >= 0)
                //    {
                //        sel--;
                //        if (sel >= 0) listBox1.SelectedIndex = sel;
                //    }
                //    break;
                //case Keys.Down:
                //    sel = listBox1.SelectedIndex;
                //    if (sel >= 0)
                //    {
                //        sel++;
                //        if (sel < listBox1.Items.Count) listBox1.SelectedIndex = sel;
                //    }
                //    break;
                case Keys.F1:
                    visibleFontToolStripMenuItem_Click(sender, new EventArgs());
                    break;
                case Keys.F2:
                    unvisibleFontToolStripMenuItem_Click(sender, new EventArgs());
                    break;
            }
        }
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        if (Screen.AllScreens.Length > 1)                   // Следим за состоянием второго экрана
        {
            if (!secondScreen)
            {
                secondScreen = true;
                toolStripMenuItem3.Enabled = true;
            }
        }
        else
        {
            if (secondScreen)
            {
                secondScreen = false;
                toolStripMenuItem3.Enabled = false;

                if (toolStripMenuItem3.Checked)
                {
                    toolStripMenuItem3.Checked = false;
                    //
                    Point p = new Point(0, 0);
                    Size s = new Size(400, 300);
                    pf.FormBorderStyle = FormBorderStyle.Sizable;
                    pf.Location = p;
                    pf.Size = s;
                }
            }
        }
    }

    private void toolStripMenuItem3_CheckedChanged(object sender, EventArgs e)
    {
        if (secondScreen && toolStripMenuItem3.Checked)
        {
            Screen scr = Screen.AllScreens[1];
            Point p = new Point(scr.Bounds.Location.X, scr.Bounds.Location.Y);
            Size s = scr.Bounds.Size;
            pf.FormBorderStyle = FormBorderStyle.None;
            pf.Location = p;
            pf.Size = s;
        }
        else
        {
            Point p = new Point(0, 0);
            Size s = new Size(400, 300);
            pf.FormBorderStyle = FormBorderStyle.Sizable;
            pf.Location = p;
            pf.Size = s;
        }
    }

}
}
```