XNA 4.0 RPG Tutorials

Part 14B

Back to Editors

I'm writing these tutorials for the new XNA 4.0 framework. The tutorials will make more sense if they are read in order. You can find the list of tutorials on the XNA 4.0 RPG tutorials page of my web site. I will be making my version of the project available for download at the end of each tutorial. It will be included on the page that links to the tutorials.

This is the second part of tutorial 14 on adding more to the editors for the game. In this part I'm going to be concentrating on the forms that hold the list of data items. Right click **FormArmor** and select **View Code**. This is the code for **FormArmor**.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;

using RpgLibrary.CharacterClasses;
using RpgLibrary.ItemClasses;

namespace RpgEditor
{
    public partial class FormArmor : FormDetails
    {
        #region Field Region
        #endregion

        #region Property Region
        #endregion

        #region Constructor Region

        public FormArmor()
        {
            InitializeComponent();

            btnAdd.Click += new EventHandler(btnAdd_Click);
            btnEdit.Click += new EventHandler(btnEdit_Click);
            btnDelete.Click += new EventHandler(btnDelete_Click);
        }

        #endregion

        #region Button Event Handler Region

        void btnAdd_Click(object sender, EventArgs e)
        {
            using (FormArmorDetails frmArmorDetails = new FormArmorDetails())
            {
                frmArmorDetails.ShowDialog();
                if (frmArmorDetails.Armor != null)
                {
                    AddArmor(frmArmorDetails.Armor);
                }
            }
        }

        void btnEdit_Click(object sender, EventArgs e)
        {
            if (lbDetails.SelectedItem != null)
            {
                string detail = lbDetails.SelectedItem.ToString();
                string[] parts = detail.Split(',');
                string entity = parts[0].Trim();

                ArmorData data = itemManager.ArmorData[entity];
                ArmorData newData = null;

                using (FormArmorDetails frmArmorData = new FormArmorDetails())
                {
                    frmArmorData.Armor = data;
                    frmArmorData.ShowDialog();

                    if (frmArmorData.Armor == null)
                        return;

                    if (frmArmorData.Armor.Name == entity)
                    {
                        itemManager.ArmorData[entity] = frmArmorData.Armor;
                        FillListBox();
                        return;
                    }

                    newData = frmArmorData.Armor;
                }

                DialogResult result = MessageBox.Show(
                    "Name has changed. Do you want to add a new entry?",
                    "New Entry",
                    MessageBoxButtons.YesNo);

                if (result == DialogResult.No)
                    return;

                if (itemManager.ArmorData.ContainsKey(newData.Name))
                {
                    MessageBox.Show("Entry already exists. Use Edit to modify the entry.");
                    return;
                }

                lbDetails.Items.Add(newData);
                itemManager.ArmorData.Add(newData.Name, newData);
            }
        }

        void btnDelete_Click(object sender, EventArgs e)
        {
            if (lbDetails.SelectedItem != null)
            {
                string detail = (string)lbDetails.SelectedItem;
                string[] parts = detail.Split(',');
                string entity = parts[0].Trim();

                DialogResult result = MessageBox.Show(
                    "Are you sure you want to delete " + entity + "?",
                    "Delete",
                    MessageBoxButtons.YesNo);

                if (result == DialogResult.Yes)
                {
                    lbDetails.Items.RemoveAt(lbDetails.SelectedIndex);
                    itemManager.ArmorData.Remove(entity);

                    if (File.Exists(FormMain.ItemPath + @"\Armor\" + entity + ".xml"))
                        File.Delete(FormMain.ItemPath + @"\Armor\" + entity + ".xml");
                }
            }
        }

        #endregion

        #region Method Region

        public void FillListBox()
        {
            lbDetails.Items.Clear();

            foreach (string s in FormDetails.ItemManager.ArmorData.Keys)
                lbDetails.Items.Add(FormDetails.ItemManager.ArmorData[s]);
        }

        private void AddArmor(ArmorData armorData)
        {
            if (FormDetails.ItemManager.ArmorData.ContainsKey(armorData.Name))
            {
                DialogResult result = MessageBox.Show(
                    armorData.Name + " already exists. Overwrite it?",
                    "Existing armor",
                    MessageBoxButtons.YesNo);

                if (result == DialogResult.No)
                    return;

                itemManager.ArmorData[armorData.Name] = armorData;
                FillListBox();
                return;
            }

            itemManager.ArmorData.Add(armorData.Name, armorData);
            lbDetails.Items.Add(armorData);
        }

        #endregion
    }
}
```

The code should look familiar, it is pretty much the same code as **FormClasses**. It just works with armor instead of entity data. There is the using statement for the **System.IO** name space. Event handlers for the click event of the buttons are wired in the constructors. The **Click** event handler of **btnAdd** creates a form in a using block. I show the form. If the **Armor** property of the form is not null I call the **AddArmor** method passing in the **Armor** property. The **Click** event handler of **btnEdit** checks to see if the **SelectedItem** of **lbDetails** is not null. It parses the string to get the name of the armor. It then gets the **ArmorData** for the selected item and sets **newData** to null. In a using statement a form is created. The **Armor** property of the form is set to the **ArmorData** of **SelectedItem.** I call the **ShowDialog** method to display the form. If the **Armor** property of form is null I exit the method. If the name is the same as before I assign the entry in the item manager to be the new armor, call **FillListBox** to update the armor and exit the method. I then set **newData** to be the **Armor** property of the form. The name of the armor changed so I display a message box asking if the new armor should be added. If the result is no I exit the method. If there is armor with that name already I display a message box and exit. If there wasn't I add the new armor to the list box and the item manager. The **Click** event handler for **btnDelete** checks to make sure that the **SelectedItem** of the list box is not null. It parses the selected item and displays a message box asking if the armor should be deleted. If the result of the message box is Yes I remove the armor from the list box and I remove if from the item manager as well. I then delete the file, if it exists.

Right click **FormShield** now and select **View Code**. The code for **FormShield** is almost a carbon copy of **FormArmor**. In fact, I copied and pasted the code. I then renamed **Armor** to **Shield** and then **armor** to **shield**. This is the code for **FormShield**.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;

using RpgLibrary.CharacterClasses;
using RpgLibrary.ItemClasses;

namespace RpgEditor
{
    public partial class FormShield : FormDetails
    {
        #region Field Region
        #endregion

        #region Property Region
        #endregion

        #region Constructor Region

        public FormShield()
        {
            InitializeComponent();

            btnAdd.Click += new EventHandler(btnAdd_Click);
            btnEdit.Click += new EventHandler(btnEdit_Click);
            btnDelete.Click += new EventHandler(btnDelete_Click);
        }

        #endregion

        #region Button Event Handler Region

        void btnAdd_Click(object sender, EventArgs e)
        {
            using (FormShieldDetails frmShieldDetails = new FormShieldDetails())
            {
                frmShieldDetails.ShowDialog();

                if (frmShieldDetails.Shield != null)
                {
                    AddShield(frmShieldDetails.Shield);
                }
            }
        }

        void btnEdit_Click(object sender, EventArgs e)
        {
            if (lbDetails.SelectedItem != null)
            {
                string detail = lbDetails.SelectedItem.ToString();
                string[] parts = detail.Split(',');
                string entity = parts[0].Trim();

                ShieldData data = itemManager.ShieldData[entity];
                ShieldData newData = null;

                using (FormShieldDetails frmShieldData = new FormShieldDetails())
                {
                    frmShieldData.Shield = data;
                    frmShieldData.ShowDialog();

                    if (frmShieldData.Shield == null)
                        return;

                    if (frmShieldData.Shield.Name == entity)
                    {
                        itemManager.ShieldData[entity] = frmShieldData.Shield;
                        FillListBox();
                        return;
                    }

                    newData = frmShieldData.Shield;
                }

                DialogResult result = MessageBox.Show(
                    "Name has changed. Do you want to add a new entry?",
                    "New Entry",
                    MessageBoxButtons.YesNo);

                if (result == DialogResult.No)
                    return;

                if (itemManager.ShieldData.ContainsKey(newData.Name))
                {
                    MessageBox.Show("Entry already exists. Use Edit to modify the entry.");
                    return;
                }

                lbDetails.Items.Add(newData);
                itemManager.ShieldData.Add(newData.Name, newData);
            }
        }

        void btnDelete_Click(object sender, EventArgs e)
        {
            if (lbDetails.SelectedItem != null)
            {
                string detail = (string)lbDetails.SelectedItem;
                string[] parts = detail.Split(',');
                string entity = parts[0].Trim();

                DialogResult result = MessageBox.Show(
                    "Are you sure you want to delete " + entity + "?",
                    "Delete",
                    MessageBoxButtons.YesNo);

                if (result == DialogResult.Yes)
                {
                    lbDetails.Items.RemoveAt(lbDetails.SelectedIndex);
                    itemManager.ShieldData.Remove(entity);

                    if (File.Exists(FormMain.ItemPath + @"\Shield\" + entity + ".xml"))
                        File.Delete(FormMain.ItemPath + @"\Shield\" + entity + ".xml");
                }
            }
        }

        #endregion

        #region Method Region

        public void FillListBox()
        {
            lbDetails.Items.Clear();
```

```
            foreach (string s in FormDetails.ItemManager.ShieldData.Keys)
                lbDetails.Items.Add(FormDetails.ItemManager.ShieldData[s]);
        }

        private void AddShield(ShieldData shieldData)
        {
            if (FormDetails.ItemManager.ShieldData.ContainsKey(shieldData.Name))
            {
                DialogResult result = MessageBox.Show(
                    shieldData.Name + " already exists. Overwrite it?",
                    "Existing shield",
                    MessageBoxButtons.YesNo);

                if (result == DialogResult.No)
                    return;

                itemManager.ShieldData[shieldData.Name] = shieldData;
                FillListBox();
                return;
            }

            itemManager.ShieldData.Add(shieldData.Name, shieldData);
            lbDetails.Items.Add(shieldData);
        }

        #endregion
    }
}
```

As you can see, other than variable and class names, the code is the same. The code for **FormWeapon** is the same again. Right click **FormWeapon** and select **View Code**. This is the code for that form.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;

using RpgLibrary.CharacterClasses;
using RpgLibrary.ItemClasses;

namespace RpgEditor
{

    public partial class FormWeapon : FormDetails
    {
        #region Field Region
        #endregion

        #region Property Region
        #endregion

        #region Constructor Region

        public FormWeapon()
        {
            InitializeComponent();

            btnAdd.Click += new EventHandler(btnAdd_Click);
            btnEdit.Click += new EventHandler(btnEdit_Click);
            btnDelete.Click += new EventHandler(btnDelete_Click);
        }
        #endregion

        #region Button Event Handler Region

        void btnAdd_Click(object sender, EventArgs e)
        {
            using (FormWeaponDetails frmWeaponDetails = new FormWeaponDetails())
            {
                frmWeaponDetails.ShowDialog();

                if (frmWeaponDetails.Weapon != null)
                {
                    AddWeapon(frmWeaponDetails.Weapon);
                }
            }
        }

        void btnEdit_Click(object sender, EventArgs e)
        {
            if (lbDetails.SelectedItem != null)
            {
                string detail = lbDetails.SelectedItem.ToString();
                string[] parts = detail.Split(',');
                string entity = parts[0].Trim();

                WeaponData data = itemManager.WeaponData[entity];
                WeaponData newData = null;

                using (FormWeaponDetails frmWeaponData = new FormWeaponDetails())
                {
                    frmWeaponData.Weapon = data;
                    frmWeaponData.ShowDialog();

                    if (frmWeaponData.Weapon == null)
                        return;

                    if (frmWeaponData.Weapon.Name == entity)
                    {
                        itemManager.WeaponData[entity] = frmWeaponData.Weapon;
                        FillListBox();
                        return;
                    }

                    newData = frmWeaponData.Weapon;
                }

                DialogResult result = MessageBox.Show(
                    "Name has changed. Do you want to add a new entry?",
                    "New Entry",
                    MessageBoxButtons.YesNo);

                if (result == DialogResult.No)
                    return;

                if (itemManager.WeaponData.ContainsKey(newData.Name))
                {
                    MessageBox.Show("Entry already exists. Use Edit to modify the entry.");
                    return;
                }

                lbDetails.Items.Add(newData);
                itemManager.WeaponData.Add(newData.Name, newData);
            }
        }

        void btnDelete_Click(object sender, EventArgs e)
        {
            if (lbDetails.SelectedItem != null)
            {
                string detail = (string)lbDetails.SelectedItem;
                string[] parts = detail.Split(',');
                string entity = parts[0].Trim();

                DialogResult result = MessageBox.Show(
                    "Are you sure you want to delete " + entity + "?",
                    "Delete",
                    MessageBoxButtons.YesNo);

                if (result == DialogResult.Yes)
                {
                    lbDetails.Items.RemoveAt(lbDetails.SelectedIndex);
                    itemManager.WeaponData.Remove(entity);

                    if (File.Exists(FormMain.ItemPath + @"\Weapon\" + entity + ".xml"))
                        File.Delete(FormMain.ItemPath + @"\Weapon\" + entity + ".xml");
                }
            }
        }

        #endregion

        #region Method Region

        public void FillListBox()
        {
            lbDetails.Items.Clear();

            foreach (string s in FormDetails.ItemManager.WeaponData.Keys)
                lbDetails.Items.Add(FormDetails.ItemManager.WeaponData[s]);
        }

        private void AddWeapon(WeaponData weaponData)
        {
            if (FormDetails.ItemManager.WeaponData.ContainsKey(weaponData.Name))
            {
                DialogResult result = MessageBox.Show(
                    weaponData.Name + " already exists. Overwrite it?",
                    "Existing weapon",
                    MessageBoxButtons.YesNo);

                if (result == DialogResult.No)
                    return;

                itemManager.WeaponData[weaponData.Name] = weaponData;
                FillListBox();
                return;
            }

            itemManager.WeaponData.Add(weaponData.Name, weaponData);
            lbDetails.Items.Add(weaponData);
        }

        #endregion
    }
}
```

It might be a pain in the ass to have to do this every time but I'm going to ask the user if they are sure they want to exit before closing the editor. If they choose the No option then closing the form will be cancelled. I will wire a handler for the **FormClosing** event in the constructor of **FormMain** and handle the event. Right click **FormMain** and select **View Code**. Change the constructor to the following and add in the following handler.

```
public FormMain()
{
    InitializeComponent();

    this.FormClosing += new FormClosingEventHandler(FormMain_FormClosing);
    newGameToolStripMenuItem.Click += new EventHandler(newGameToolStripMenuItem_Click);
    openGameToolStripMenuItem.Click += new EventHandler(openGameToolStripMenuItem_Click);
    saveGameToolStripMenuItem.Click += new EventHandler(saveGameToolStripMenuItem_Click);
    exitToolStripMenuItem.Click += new EventHandler(exitToolStripMenuItem_Click);

    classesToolStripMenuItem.Click += new EventHandler(classesToolStripMenuItem_Click);
    armorToolStripMenuItem.Click += new EventHandler(armorToolStripMenuItem_Click);
    shieldToolStripMenuItem.Click += new EventHandler(shieldToolStripMenuItem_Click);
    weaponToolStripMenuItem.Click += new EventHandler(weaponToolStripMenuItem_Click);
}

void FormMain_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult result = MessageBox.Show(
        "Unsaved changes will be lost. Are you sure you want to exit?",
        "Exit?",
        MessageBoxButtons.YesNo,
        MessageBoxIcon.Warning);

    if (result == DialogResult.No)
        e.Cancel = true;
}
```

That gets the editors up and going. With these you have different classes for the player character, and non-player characters, and some basic items. We still don't have any data to work with though and there is something that I need to explain, the formula fields in the **EntityData** class. I don't plan on using complicated formulae for health, mana, or stamina. What I plan to do is have a three part formula, with each part separated with a |. The first part is the base for the attribute, the second is the basic attribute to add to the base, and the third is a random amount that will be added with each level. So, if you had 20|CON|12, the formula would be 20 + CON + 1-12 for first level and 1-12 more points each level gained after the first. I will be, eventually, be adding in modifiers to health, mana, and stamina. The reason I say modifiers rather than bonuses is that bonuses, to me, signifies positives and there could be penalties from injuries or cursed items. If a class can not have the attribute, fighters don't have mana, you use 0|0|0 as the formula.

I was torn about whether or not to go over adding in how I came up with the data values I used or not. In the end I decided that it was worth it. I will be using a 100 point system for the basic character attributes with 10 being an “average” person. As you've seen I decided to go with Fighters, Rogues, Priests, and Wizards as character classes for the player character. There will be other character classes that are for non-player characters. Fighters are strong, hardy characters, and are not gifted magically. Rogues are fair fighters but very perceptive and dexterous. Priests are not bad fighters and have access to healing magic and a few offensive spells. Wizards, while not the strongest physically, command powerful magics that make them dreaded foes. This is the data I used for each of the character classes.

Character Classes

Class Name	STR	DEX	CUN	WIL	MAG	CON	Health Formula	Stamina Formula	Magic Formula
Fighter	14	12	10	12	10	12	20|CON|12	12|WIL|12	0|0|0
Rogue	10	14	14	12	10	10	10|CON|10	10|WIL|10	0|0|0|
Priest	12	10	12	12	12	12	12|CON|12	0|0|0	10|WIL|10
Wizard	10	10	12	14	14	10	10|CON|10	0|0|0|	20|WIL|12

What I suggest is you run the editor and create a game and data, to see the editor in action and we really need data to work with. Launch the editor and then create a new game. Select a directory other than the **EyesOfTheDragonContent** folder. I named my game: **Eyes of the Dragon**. For the description I put something like: **Tutorial game for creating a RPG with XNA 4.0**. Click **Classes** to bring up the **Classes** form. Add each of the classes above. My data is in the project file for this tutorial, or if you just want the data: http://xnagpa.net/xna4/downloads/gamedata14.zip.

I'm now going to go into creating a few items in this tutorial. I'm going to set up a few tables with the values I used. If you're not interested in doing them on your own they are all in the file I linked to in the last paragraph. These are just suggest values.

Armor

Name	Type	Price	Weight	Location	Defense Value	Defense Modifier	Allowed Classes
Leather Gloves	Gloves	10	1	Hands	5	0	Fighter Rogue Priest
Leather Boots	Boots	10	1	Feet	5	0	Fighter Rogue Priest Wizard
Leather Armor	Armor	20	8	Body	10	0	Fighter Rogue Priest
Studded Leather Gloves	Gloves	15	2	Hands	7	0	Fighter Rogue Priest
Studded Leather Boots	Boots	15	2	Feet	7	0	Fighter Rogue Priest
Studded Leather Armor	Armor	30	10	Body	14	0	Fighter Rogue Priest
Leather Helm	Helm	10	2	Head	5	0	Fighter Rogue Priest
Studded Leather Helm	Helm	15	3	Head	7	0	Fighter Rogue Priest
Chain Mail Boots	Boots	30	4	Feet	10	0	Fighter Priest
Chain Mail Gloves	Gloves	30	4	Hands	10	0	Fighter Priest
Chain Mail	Armor	80	25	Body	20	0	Fighter Priest
Chain Mail Helm	Helm	40	8	Head	10	0	Fighter Priest
Light Robes	Robes	10	5	Body	2	0	Wizard
Medium Robes	Robes	20	8	Body	5	0	Wizard

Shields

Name	Type	Price	Weight	Defense Value	Defense Modifier	Allowed Classes
Small Wooden Shield	Small	5	4	5	0	Fighter Rogue Priest
Medium Wooden Shield	Medium	10	8	8	0	Fighter Priest
Large Wooden Shield	Large	20	12	15	0	Fighter
Small Metal Shield	Small	10	8	8	0	Fighter Rogue Priest
Medium Metal Shield	Medium	40	12	12	0	Fighter Priest
Large Metal Shield	Large	80	16	20	0	Fighter
Large Kite Shield	Large	100	18	25	0	Fighter
Heavy Tower Shield	Large	125	20	30	0	Fighter

Weapons

Name	Type	Price	Weight	Hands	Attack Value	Attack Modifier	Damage Value	Damage Modifier	Allowed Classes
Club	Crushing	8	10	One	4	0	6	0	Fighter Rogue Priest
Mace	Crushing	16	12	One	6	0	8	0	Fighter Rogue Priest
Flail	Crushing	20	14	One	8	0	10	0	Fighter Priest
Apprentice Staff	Magic	20	5	Two	6	0	6	0	Wizard
Acolyte Staff	Magic	40	8	Two	8	0	8	0	Wizard
Dagger	Piercing	10	3	One	4	0	6	0	Fighter Rogue
Short Sword	Piercing	20	10	One	6	0	8	0	Fighter Rogue
Long Sword	Slashing	40	15	One	10	0	12	0	Fighter Rogue
Broad Sword	Slashing	60	18	One	12	0	14	0	Fighter Rogue
Great Sword	Slashing	80	25	Two	12	0	16	0	Fighter
Halberd	Slashing	100	30	Two	16	0	20	0	Fighter
War Axe	Slashing	20	15	One	10	0	10	0	Fighter Rogue
Battle Axe	Slashing	50	25	Two	12	0	16	0	Fighter

I'm going to end the second part of the tutorial here. I'd like to try and keep the tutorials to a reasonable length. I encourage you to visit the news page of my site, XNA Game Programming Adventures, for the latest news on my tutorials.

Good luck in your game programming adventures!

Jamie McMahon