Creating a Role Playing Game with XNA Game Studio
Part 51
Inventory - Part 1

To follow along with this tutorial you will have to have read the previous tutorials to understand much of what it going on. You can find a list of tutorials here: XNA Role Playing Game Tutorials You will also find the latest version of the project on the web site on that page. If you want to follow along and type in the code from this PDF as you go, you can find the previous project at this link: http://www.jtmbooks.com/rpgtutorials/New2DRPG50.zip You can download the graphics from this link: Graphics.zip

This is short tutorial will pave the way for future tutorials on inventory management. This can be a rather complex topic and is important for a role playing game. There player character shouldn't be able to carry hundreds of pounds of equipment and still be able to act like they are carrying nothing. Other things to consider are buying and selling items. The possibility of upgrading items exists as well and other scenarios too.

Before I can continue on with picking up items I need to start work on the player's inventory. The player will be carrying items, equipping items, dropping items, selling items, and using items. There has to be a system in place form managing items for the player. For this tutorial I will just be laying the foundation for inventory.

The handling of the player character's inventory will be done in the player character classes: **PlayerCharacter**, **FighterCharacter**, **ThiefCharacter**, **PriestCharacter**, and **WizardCharacter**. What I did was in the **PlayerCharacter** class is add protected fields that should be available to any of the inherited classes and properties to expose them. I also added fields and properties to some of the inherited classes so they can use items other classes cannot. I decided that only fighters and priests can use shields so there is nothing related to shields in the **ThiefCharacter** and **WizardCharacter** classes.

To get started there is one thing that I need to do. Items are stored as classes so they are reference types. If you change one item all items that reference that item will be affected. I plan on the player being able to modify items with blessings to improve them and things happening to items to weaken them. This means that the items shouldn't be added to the player's inventory directly from the dictionaries of items, they should be given copies. The best way to do that was to add the **ICloneable** interface to the **BaseItem** class abstractly so that all inherited classes must implement it. This way when you add an item to the player's inventory you can add a call to **Clone** to make it a copy.

The first thing to do is to implement the **ICloneable** interface abstractly for **BaseItem** so that the inherited classes must implement the **Clone** method. To do that you add the **ICloneable** interface to the definition of **BaseItem** and then make the **Clone** method abstract. This is the new code for the **BaseItem** class.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace New2DRPG.ItemClasses
{
    public enum ItemSize { Tiny, Small, Medium, Large };

    public abstract class BaseItem : ICloneable
    {
        string name;
        int price;
        int weight;
        ItemSize itemSize;

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

        public int Price
        {
            get { return price; }
            protected set { price = value; }
        }

        public int Weight
        {
            get { return weight; }
            protected set { weight = value; }
        }

        public ItemSize Size
        {
            get { return itemSize; }
            protected set { itemSize = value; }
        }

        public BaseItem(string name, int price, int weight, ItemSize size)
        {
            Name = name;
            Price = price;
            Weight = weight;
            Size = size;
        }

        public abstract object Clone();
    }
}
```

The next thing to do is to implement the **Clone** method in all of the classes that inherit from **BaseItem**. Since the **Clone** method was defined as abstract you must override the **Clone** method in the inherited classes. All of the fields in the classes are value types, not reference types, so it is safe to just create a new instance of the appropriate class using the fields. Any changes to those fields will not affect the base items. What I did was create a new instance of the specific class and cast it as an **object** because that is what the **Clone** method returns. The code for the **Weapon**, **Shield**, and **Armor** classes follow.

Weapon Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace New2DRPG.ItemClasses
{
    public enum Hands { One, Two }

    public class Weapon : BaseItem
    {
        Hands hands;
        int attackValue;
        int attackBonus;

        public Hands NumberHands
        {
            get { return hands; }
            protected set { hands = value; }
        }

        public int AttackValue
        {
            get { return attackValue; }
            protected set { attackValue = value; }
        }

        public int AttackBonus
        {
            get { return attackBonus; }
            protected set { attackBonus = value; }
        }

        public Weapon(
                string weaponName,
                int price,
                int weight,
                ItemSize size,
                Hands hands,
                int attackValue,
                int attackBonus)
            : base(weaponName, price, weight, size)
        {
            NumberHands = hands;
            AttackValue = attackValue;
            AttackBonus = attackBonus;
        }

        public override object Clone()
        {
            Weapon weapon = new Weapon(
                Name,
                Price,
                Weight,
                Size,
                NumberHands,
                AttackValue,
                AttackBonus);
            return (object)weapon;
        }
    }
}
```

Armor Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace New2DRPG.ItemClasses
{
    public class Armor : BaseItem
    {
        int defenseValue;
        int defenseBonus;

        public int DefenseValue
        {
            get { return defenseValue; }
            protected set { defenseValue = value; }
        }

        public int DefenseBonus
        {
            get { return defenseBonus; }
            protected set { defenseBonus = value; }
        }

        public Armor(
                string armorName,
                int price,
                int weight,
                ItemSize size,
                int defenseValue,
                int defenseBonus)
            : base(armorName, price, weight, size)
        {
            DefenseValue = defenseValue;
            DefenseBonus = defenseBonus;
        }

        public override object Clone()
        {
            Armor armor = new Armor(
                Name,
                Price,
                Weight,
                Size,
                DefenseValue,
                DefenseBonus);

            return (object)armor;
        }
    }
}
```

Weapon Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace New2DRPG.ItemClasses
{
    public class Shield : BaseItem
    {
        int defenseValue;
        int defenseBonus;

        public int DefenseValue
        {
            get { return defenseValue; }
            protected set { defenseValue = value; }
        }

        public int DefenseBonus
        {
            get { return defenseBonus; }
            protected set { defenseBonus = value; }
        }

        public Shield(
                string shieldName,
                int price,
                int weight,
                ItemSize size,
                int defenseValue,
                int defenseBonus)
            : base(shieldName, price, weight, size)
        {
            DefenseValue = defenseValue;
            DefenseBonus = defenseBonus;
        }

        public override object Clone()
        {
            Shield shield = new Shield(
                Name,
                Price,
                Weight,
                Size,
                DefenseValue,
                DefenseBonus);

            return (object)shield;
        }
    }
}
```

I decided, at least for the moment, all player characters will have a backpack to carry their items in, a weapon they can use to fight with, and armor for protection. I will be making the game so that a wizard can not wear heavy armor like a fighter can and a fighter can not use a wand like a wizard can. Since all classes will have a backpack, armor, and weapon it made sense to make those protected fields of the base class so they will be available to all of the classes that inherit form **PlayerCharacter**. For this tutorial there is no way to add items to a backpack, that will be something that is done down the road. To get and set a player's armor or weapon I created public virtual properties so that they can be overridden by inherited classes. I used a **List<BaseItem>** for the backpack so there isn't a fixed number of items in the backpack. Add using statement at the top of the **PlayerCharacter** class for the **New2DRPG.ItemClasses** name space and near the constructor add the following fields and properties.

```
using New2DRPG.ItemClasses;

protected List<BaseItem> backpack = new List<BaseItem>();

protected Weapon weapon;
protected Armor armor;

public virtual Weapon Weapon
{
    get { return weapon; }
    set { weapon = value; }
}

public virtual Armor Armor
{
    get { return armor; }
    set { armor = value; }
}
```

For the **FighterCharacter** class I added in a field for a **Shield** object and a property to expose the shield. I also went a head and created overrides of the **Weapon** and **Armor** properties. I did the same thing for the **PriestCharacter** class. For the **WizardCharacter** and **ThiefCharacter** classes I just created the overrides of the **Weapon** and **Armor** properties. I did the overrides now so I wouldn't have to go back and add them in later. This is the new code for the **PriestCharacter**, **ThiefCharacter**, **WizardCharacter**, and **FighterCharacter** classes. You will also need a using statement for the **New2DRPG.ItemClasses** name space.

FighterCharacter

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;
using New2DRPG.ItemClasses;

namespace New2DRPG
{
    class FighterCharacter : PlayerCharacter
    {
        const int startingHitPoints = 20;
        const int startingSpellPoints = 5;
        const int startingStrength = 16;
        const int startingStamina = 14;
        const int startingAgility = 12;
        const int startingSpeed = 10;
        const int startingIntellect = 10;
        const int startingLuck = 10;

        protected Shield shield;

        public override Armor Armor
        {
            get
            {
                return base.Armor;
            }
            set
            {
                base.Armor = value;
            }
        }

        public override Weapon Weapon
        {
            get
            {
                return base.Weapon;
            }
            set
            {
                base.Weapon = value;
            }
        }

        public Shield Shield
        {
            get { return shield; }
            set { shield = value; }
        }

        public FighterCharacter(string name,
                bool gender,
                Level difficulty,
                Game game)
            : base(game)
        {
            this.difficulty = difficulty;
            this.className = "Fighter";
            this.name = name;
            this.gender = gender;

            this.hitPoints[0] = startingHitPoints;
            this.hitPoints[1] = startingHitPoints;
            this.spellPoints[0] = startingSpellPoints;
            this.spellPoints[1] = startingSpellPoints;

            abilities.Strength = startingStrength;
            abilities.Stamina = startingStamina;
            abilities.Agility = startingAgility;
```

```
            abilities.Speed = startingSpeed;
            abilities.Intellect = startingIntellect;
            abilities.Luck = startingLuck;
        }
    }
}
```

PriestCharacter

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;
using New2DRPG.ItemClasses;

namespace New2DRPG
{
    class PriestCharacter : PlayerCharacter
    {
        const int startingHitPoints = 15;
        const int startingSpellPoints = 15;

        const int startingStrength = 12;
        const int startingStamina = 12;
        const int startingAgility = 10;
        const int startingSpeed = 10;
        const int startingIntellect = 12;
        const int startingLuck = 12;

        protected Shield shield;

        public override Armor Armor
        {
            get
            {
                return base.Armor;
            }
            set
            {
                base.Armor = value;
            }
        }

        public override Weapon Weapon
        {
            get
            {
                return base.Weapon;
            }
            set
            {
                base.Weapon = value;
            }
        }

        public Shield Shield
        {
            get { return shield; }
            set { shield = value; }
        }

        public PriestCharacter(string name,
                bool gender,
                Level difficulty,
                Game game)
            : base(game)
        {
            this.difficulty = difficulty;
            this.className = "Priest";
            this.name = name;
            this.gender = gender;
            this.hitPoints[0] = startingHitPoints;
            this.hitPoints[1] = startingHitPoints;
            this.spellPoints[0] = startingSpellPoints;
            this.spellPoints[1] = startingSpellPoints;

            abilities.Strength = startingStrength;
            abilities.Stamina = startingStamina;
            abilities.Agility = startingAgility;
            abilities.Speed = startingSpeed;
            abilities.Intellect = startingIntellect;
            abilities.Luck = startingLuck;
        }
    }
}
```

ThiefCharacter

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;
using New2DRPG.ItemClasses;

namespace New2DRPG
{
    class ThiefCharacter : PlayerCharacter
    {
        const int startingHitPoints = 15;
        const int startingSpellPoints = 5;
        const int startingStrength = 12;
        const int startingStamina = 10;
        const int startingAgility = 14;
        const int startingSpeed = 12;
        const int startingIntellect = 10;
        const int startingLuck = 12;

        public override Armor Armor
        {
            get
            {
                return base.Armor;
            }
            set
            {
                base.Armor = value;
            }
        }

        public override Weapon Weapon
        {
            get
            {
                return base.Weapon;
            }
            set
            {
                base.Weapon = value;
            }
        }

        public ThiefCharacter(string name,
                bool gender,
                Level difficulty,
                Game game)
            : base(game)
        {
            this.difficulty = difficulty;
            this.className = "Thief";
            this.name = name;
            this.gender = gender;
            this.hitPoints[0] = startingHitPoints;
            this.hitPoints[1] = startingHitPoints;
            this.spellPoints[0] = startingSpellPoints;
            this.spellPoints[1] = startingSpellPoints;

            abilities.Strength = startingStrength;
            abilities.Stamina = startingStamina;
            abilities.Agility = startingAgility;
            abilities.Speed = startingSpeed;
            abilities.Intellect = startingIntellect;
            abilities.Luck = startingLuck;
        }
    }
}
```

WizardCharacter

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;
using New2DRPG.ItemClasses;

namespace New2DRPG
{
    class WizardCharacter : PlayerCharacter
    {
        const int startingHitPoints = 10;
        const int startingSpellPoints = 20;

        const int startingStrength = 10;
        const int startingStamina = 12;
        const int startingAgility = 12;
        const int startingSpeed = 12;
        const int startingIntellect = 16;
        const int startingLuck = 10;

        public override Armor Armor
        {
            get
            {
                return base.Armor;
            }
            set
            {
                base.Armor = value;
            }
        }

        public override Weapon Weapon
        {
            get
            {
                return base.Weapon;
            }
            set
            {
                base.Weapon = value;
            }
        }

        public WizardCharacter(string name,
                bool gender,
                Level difficulty,
                Game game)
            : base(game)
        {
            this.difficulty = difficulty;
            this.className = "Wizard";
            this.name = name;
            this.gender = gender;
            this.hitPoints[0] = startingHitPoints;
            this.hitPoints[1] = startingHitPoints;
            this.spellPoints[0] = startingSpellPoints;
            this.spellPoints[1] = startingSpellPoints;

            abilities.Strength = startingStrength;
            abilities.Stamina = startingStamina;
            abilities.Agility = startingAgility;
            abilities.Speed = startingSpeed;
            abilities.Intellect = startingIntellect;
            abilities.Luck = startingLuck;
        }
    }
}
```

That is it for this tutorial. Like I said, it was more of a preparation tutorial than anything else. There is still a lot to do when it comes to inventory management and picking up items. I will be adding in more and more functionality to the game that you would expect to find in a role playing game. I encourage you to keep either visiting my site http://xnagpa.net or my blog, XNA Game Programming Adventures Blog for the latest news on my tutorials.