Eyes of the Dragon Tutorials
Part 51
Refactoring Game Items

I'm writing these tutorials for the MonoGame 3.8 framework using Visual Studio 2019. The tutorials will make more sense if they are read in order. You can find the list of tutorials on the Eyes of the Dragon page of my web blog. I will be making each version of the project available on GitHub here. It will be included on the page that links to the tutorials.

In this tutorial I will be updating game items. I will update the GameItemManager to be static. I will also have GameItem implement the ICloneable interface. Eventually, I will want to update the editors for items to include a source rectangle property. For now, I will create GameItems manually in the Game1 class. I will also update the CharacterGeneratorScreen to use the updated GameItemManager.

The first step is to update the GameItem class to implement the ICloneable interface and include a Clone method. Replace the GameItem class with the following.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using RpgLibrary.ItemClasses;

namespace MGRpgLibrary.ItemClasses
{
    public class GameItem : ICloneable
    {
        #region Field Region

        public Vector2 Position;
        private string image;
        private Rectangle? sourceRectangle;
        private BaseItem baseItem;
        private Type type;

        #endregion

        #region Property Region

        public string Image
        {
            get { return image; }
        }

        public Rectangle? SourceRectangle
        {
            get { return sourceRectangle; }
            set { sourceRectangle = value; }
        }

        public BaseItem Item
        {
            get { return baseItem; }
        }

        public Type Type
        {
            get { return type; }
        }

        #endregion

        #region Constructor Region

        private GameItem()
        {

        }

        public GameItem(BaseItem item, string texture, Rectangle? source)
        {
            baseItem = item;
            image = texture;
            sourceRectangle = source;
            type = item.GetType();
        }

        #endregion

        #region Method Region

        public void Draw(SpriteBatch spriteBatch)
        {
            spriteBatch.Draw(TextureManager.GetTexture(image), Position, sourceRectangle,
Color.White);
        }

        public void Draw(SpriteBatch spriteBatch, Rectangle destination)
        {
            spriteBatch.Draw(TextureManager.GetTexture(Image), destination, sourceRectangle,
Color.White);
        }

        public object Clone()
        {
            GameItem item = new GameItem()
            {
                baseItem = this.baseItem,
                Position = this.Position,
                image = this.image,
                sourceRectangle = this.sourceRectangle,
                type = this.type
            };

            return item;
        }

        #endregion

        #region Virtual Method region
        #endregion
```

```
    }
}
```

The changes were to add the ICloneable interface to the class declaration. I then changed the baseItem field from readonly to normal. I added a private constructor for creating a GameItem that will be used in the implementation of the Clone method. The Clone method uses the new constructor to generate an instance of the class and return it.

The next step is to update the GameItemManager class. What I want to do is add methods to add an item to the dictionary of items and one to retrieve an item. Also, I made the field and property static because I only ever want one instance of the dictionary. Replace the GameItemManager class with the following code.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using RpgLibrary.ItemClasses;

namespace MGRpgLibrary.ItemClasses
{
    public class GameItemManager
    {
        #region Field Region

        static readonly Dictionary<string, GameItem> gameItems = new Dictionary<string,
GameItem>();
        static SpriteFont spriteFont;

        #endregion

        #region Property Region

        public static Dictionary<string, GameItem> GameItems
        {
            get { return gameItems; }
        }

        public static SpriteFont SpriteFont
        {
            get { return spriteFont; }
            private set { spriteFont = value; }
        }

        #endregion

        #region Constructor Region

        public GameItemManager(SpriteFont spriteFont)
        {
            SpriteFont = spriteFont;
        }

        #endregion

        #region Method Region
        public static void AddItem(string name, GameItem item)
        {
            if (!GameItems.ContainsKey(name))
            {
                GameItems.Add(name, item);
            }
        }

        public static GameItem GetItem(string name)
        {
            if (GameItems.ContainsKey(name))
            {
                return (GameItem)GameItems[name].Clone();
            }

            return null;
        }

        #endregion

        #region Virtual Method region
        #endregion
    }
}
```

I've already covered the changes so I won't go over them again. I don't want to go into the editor for creating items in this tutorial. So, I am going to add items manually to the GameItemManager. Also, I will update the CharacterGeneratorScreen so that it uses the GameItemManager instead of creating GameItems on the fly. First, change the LoadContent method of the Game1 class to generate the GameItems.

```
protected override void LoadContent()
{
    _spriteBatch = new SpriteBatch(GraphicsDevice);

    DataManager.ReadEntityData(Content);
    DataManager.ReadArmorData(Content);
    DataManager.ReadShieldData(Content);
    DataManager.ReadWeaponData(Content);
    DataManager.ReadChestData(Content);
    DataManager.ReadKeyData(Content);
    DataManager.ReadSkillData(Content);

    FontManager.AddFont("testfont", Content.Load<SpriteFont>("Fonts/scenefont"));
    TextureManager.AddTexture("FullSheet", Content.Load<Texture2D>("GUI/ProjectUtumno_full"));
    string[] fileNames = Directory.GetFiles(
        Path.Combine("Content/Game/Items", "Armor"),
        "*.xnb");

    foreach (string a in fileNames)
    {
        string path = "Game/Items/Armor/" + Path.GetFileNameWithoutExtension(a);

        ArmorData armorData = Content.Load<ArmorData>(path);
        ItemManager.AddArmor(new Armor(armorData));
    }

    fileNames = Directory.GetFiles(
        Path.Combine("Content/Game/Items", "Shield"),
        "*.xnb");
    foreach (string a in fileNames)
    {
        string path = "Game/Items/Shield/" + Path.GetFileNameWithoutExtension(a);

        ShieldData shieldData = Content.Load<ShieldData>(path);
        ItemManager.AddShield(new Shield(shieldData));
    }

    fileNames = Directory.GetFiles(
        Path.Combine("Content/Game/Items", "Weapon"),
        "*.xnb");

    foreach (string a in fileNames)
    {
        string path = "Game/Items/Weapon/" + Path.GetFileNameWithoutExtension(a);

        WeaponData weaponData = Content.Load<WeaponData>(path);
        ItemManager.AddWeapon(new Weapon(weaponData));
    }

    GameItemManager.AddItem("Long Sword", new GameItem(ItemManager.GetWeapon("Long Sword"),
"FullSheet", new Rectangle(1696, 1408, 32, 32)));
    GameItemManager.AddItem("Short Sword", new GameItem(ItemManager.GetWeapon("Short Sword"),
"FullSheet", new Rectangle(800, 1504, 32, 32)));
    GameItemManager.AddItem("Apprentice Staff", new GameItem(ItemManager.GetWeapon("Apprentice
Staff"), "FullSheet", new Rectangle(224, 1408, 32, 32)));
    GameItemManager.AddItem("Acolye Staff", new GameItem(ItemManager.GetWeapon("Acolyte
Staff"), "FullSheet", new Rectangle(256, 1408, 32, 32)));
    GameItemManager.AddItem("Leather Armor", new GameItem(ItemManager.GetArmor("Leather
Armor"), "FullSheet", new Rectangle(1248, 1216, 32, 32)));
    GameItemManager.AddItem("Chain Mail", new GameItem(ItemManager.GetArmor("Chain Mail"),
"FullSheet", new Rectangle(1472, 1184, 32, 32)));
    GameItemManager.AddItem("Studded Leather Armor", new GameItem(ItemManager.GetArmor("Studded
Leather Armor"), "FullSheet", new Rectangle(1984, 1120, 32, 32)));
    GameItemManager.AddItem("Light Robes", new GameItem(ItemManager.GetArmor("Light Robes"),
"FullSheet", new Rectangle(992, 1216, 32, 32)));
    GameItemManager.AddItem("Medium Robes", new GameItem(ItemManager.GetArmor("Medium Robes"),
"FullSheet", new Rectangle(1024, 1216, 32, 32)));
}
```

All I do is call the AddItem method of the GameItemManager to add the item to the dictionary of items. I copied and pasted the code from the CharacterGeneratorScreen for creating GameItems for the merchant. Also, I cut the code that loads the items from the LoadWorld method and pasted it into the LoadContent method of the Game1 class.

What I did next is update the LoadWorld method of the CharacterGeneratorScreen to use the GameItemManager instead of generating items on the fly. Here is the code LoadWorld method of the CharacterGeneratorScreen.

```
private void LoadWorld()
{
    RpgLibrary.WorldClasses.LevelData levelData =
        Game.Content.Load<RpgLibrary.WorldClasses.LevelData>(@"Game\Levels\Starting Level");

    RpgLibrary.WorldClasses.MapData mapData =
        Game.Content.Load<RpgLibrary.WorldClasses.MapData>(@"Game\Levels\Maps\" +
levelData.MapName);
```

```
    CharacterLayerData charData =
        Game.Content.Load<CharacterLayerData>(@"Game\Levels\Chars\Starting Level");
    CharacterLayer characterLayer = new CharacterLayer();
    MobLayer mobLayer = new MobLayer();

    TileMap map = TileMap.FromMapData(mapData, Game.Content);

    foreach (var c in charData.Characters)
    {
        Character character;

        if (c.Value is NonPlayerCharacterData data)
        {
            Entity entity = new Entity(c.Value.Name, c.Value.EntityData, c.Value.Gender,
EntityType.NPC);

            using (Stream stream = new FileStream(c.Value.TextureName, FileMode.Open,
FileAccess.Read))
            {
                Texture2D texture = Texture2D.FromStream(GraphicsDevice, stream);
                AnimatedSprite sprite = new AnimatedSprite(texture,
AnimationManager.Instance.Animations)
                {
                    Position = new Vector2(c.Key.X * Engine.TileWidth, c.Key.Y *
Engine.TileHeight)
                };

                character = new NonPlayerCharacter(entity, sprite);

                ((NonPlayerCharacter)character).SetConversation(
                    data.CurrentConversation);
            }

            characterLayer.Characters.Add(c.Key, character);
        }
    }

    map.AddLayer(characterLayer);
    map.AddLayer(mobLayer);

    Level level = new Level(map);

    ChestData chestData = Game.Content.Load<ChestData>(@"Game\Chests\Plain Chest");

    Chest chest = new Chest(chestData);

    BaseSprite chestSprite = new BaseSprite(
        containers,
        new Rectangle(0, 0, 32, 32),
        new Point(10, 10));

    ItemSprite itemSprite = new ItemSprite(
        chest,
        chestSprite);

    level.Chests.Add(itemSprite);

    World world = new World(GameRef, GameRef.ScreenRectangle);

    world.Levels.Add(level);
    world.CurrentLevel = 0;

    AnimatedSprite s = new AnimatedSprite(
        GameRef.Content.Load<Texture2D>(@"SpriteSheets\Eliza"),
        AnimationManager.Instance.Animations)
    {
        Position = new Vector2(0 * Engine.TileWidth, 5 * Engine.TileHeight)
    };

    EntityData ed = new EntityData("Eliza", 1, 10, 10, 10, 10, 10, 10, "20|CON|12", "16|WIL|
16",
        "0|0|0");

    Entity e = new Entity("Eliza", ed, EntityGender.Female, EntityType.NPC);

    NonPlayerCharacter npc = new NonPlayerCharacter(e, s);

    npc.SetConversation("eliza1");
    //world.Levels[world.CurrentLevel].Characters.Add(npc);

    s = new AnimatedSprite(
        GameRef.Content.Load<Texture2D>(@"SpriteSheets\Eliza"),
        AnimationManager.Instance.Animations)
    {
        Position = new Vector2(10 * Engine.TileWidth, 0)
    };

    ed = new EntityData("Barbra", 2, 10, 10, 10, 10, 10, 10, "20|CON|12", "16|WIL|16", "0|0|
0");

    e = new Entity("Barbra", ed, EntityGender.Female, EntityType.Merchant);

    Merchant m = new Merchant(e, s);
    Texture2D items = Game.Content.Load<Texture2D>("ObjectSprites/roguelikeitems");
    m.Backpack.AddItem(GameItemManager.GetItem("Long Sword"));
    m.Backpack.AddItem(GameItemManager.GetItem("Short Sword"));
    m.Backpack.AddItem(GameItemManager.GetItem("Apprentice Staff"));
    m.Backpack.AddItem(GameItemManager.GetItem("Acolyte Staff"));
    m.Backpack.AddItem(GameItemManager.GetItem("Leather Armor"));
    m.Backpack.AddItem(GameItemManager.GetItem("Chain Mail"));
    m.Backpack.AddItem(GameItemManager.GetItem("Studded Leather Armor"));
    m.Backpack.AddItem(GameItemManager.GetItem("Light Robes"));
    m.Backpack.AddItem(GameItemManager.GetItem("Medium Robes"));
    world.Levels[world.CurrentLevel].Characters.Add(m);
    ((CharacterLayer)world.Levels[world.CurrentLevel].Map.Layers.Find(x => x is
CharacterLayer)).Characters.Add(new Point(10, 0), m);
    GamePlayScreen.World = world;

    ed = new EntityData("Bandit", 1, 10, 12, 12, 10, 10, 10, "20|CON|10", "12|WIL|12", "0|0|
0");

    e = new Entity("Bandit", ed, EntityGender.Male, EntityType.Monster);

    s = new AnimatedSprite(
        GameRef.Content.Load<Texture2D>(@"PlayerSprites/malerogue"),
        AnimationManager.Instance.Animations);

    Mob mob = new Bandit(e, s);

    ((MobLayer)world.Levels[world.CurrentLevel].Map.Layers.Find(x => x is
MobLayer)).Mobs.Add(new Rectangle(0, 512, 32, 32), mob);

    mob.Entity.Equip(GameItemManager.GetItem("Short Sword"));
    mob.Drops.Add(GameItemManager.GetItem("Short Sword"));
}
```

All I did was instead of creating an item use the GameItemManager's GetItem method to get the game items. I also update the Backpack class to use clones of GameItems rather than the original. Replace the Backpack with the following.

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

namespace MGRpgLibrary.ItemClasses
{
    public class Backpack
    {
        #region Field Region

        readonly List<GameItem> items;

        #endregion

        #region Property Region

        public List<GameItem> Items
        {
            get { return items; }
        }

        public int Capacity
        {
            get { return items.Count; }
        }

        #endregion

        #region Constructor Region

        public Backpack()
        {
            items = new List<GameItem>();
        }

        #endregion

        #region Method Region

        public GameItem GetItem(string name)
        {
            GameItem item = items.Find(x => x.Item.Name == name);

            if (item != null)
            {
                items.Remove(item);
            }
            return item;
        }

        public BaseItem PeekItem(string name)
        {
            GameItem item = (GameItem)items.Find(x => x.Item.Name == name).Clone();

            return item.Item;
        }

        public void AddItem(GameItem gameItem)
        {
            items.Add((GameItem)gameItem.Clone());
        }

        public void RemoveItem(GameItem gameItem)
        {
            items.Remove(gameItem);
        }

        #endregion

        #region Virtual Method region
        #endregion
    }
}
```

All I did was in the PeekItem, and AddItem methods is cast the object to a GameItem and call the Clone method. I didn't in the GetItem method because we want to use the actual object, and not a clone of it, so that the actual item is removed from the backpack.

If you build and run at this point the game will function as before. You will not notice a difference at all, nothing visual has changed. This was an important step, though. It will make life easier down the road.

So, that is going to be it for this tutorial. I accomplished what I intended and I don't want to venture further in this tutorial. So, please continue to visit my blog, https://cynthiamcmahon.ca/blog/, for the latest news on my tutorials and other goodness.

Good luck with your Game Programming Adventures!

Cynthia