Shadow Monsters – MonoGame Tutorial Series
Chapter 10
Items

This tutorial series is about creating a Pokemon style game with the MonoGame Framework called Shadow Monsters. The tutorials will make more sense if you read them in order as each tutorial builds on the previous tutorials. You can find the list of tutorials on my blog: Shadow Monsters. The source code for each tutorial will be available as well. I will be using Visual Studio 2019 Community for the series. The code should compile on the 2013, 2015 and 2017 versions as well.

I want to mention though that the series is released as Creative Commons 3.0 Attribution. It means that you are free to use any of the code or graphics in your own game, even for commercial use, with attribution. Just give credit to Cynthia McMahon and add a link to my site, https://mygameprogrammingadventures.blogspot,com. Screenshots of your project and/or a video of game play would be appreciated.

I also want to mention that I assume you have a basic understanding of C# and MonoGame. If you don't I recommend that you learn basic C# and work with MonoGame a little. Enough to know the basics of fields, properties, methods, classes and the MonoGame framework.

This tutorial is going to focus on adding items to the game. First I need to address an issue that came up in tutorial nine that didn't make it into the tutorial. I had to add a check to the AddState method of the StateManager to make sure a state was not on the stack before adding it. Update that method to the following.

```
private void AddState(GameState state)
{
    gameStates.Push(state);
    if (!Game.Components.Contains(state))
        Game.Components.Add(state);
    StateChanged += state.StateChanged;
}
```

To get started on items right click the ShadowMonsters project in the Solution Explorer, select Add and then New Folder. Name this new folder Items. Right click the Items folder, select Add and then New Item. From the list that pops up choose Interface. Name this new interface IItem. Here is the code,

```
using Microsoft.Xna.Framework;
using ShadowMonsters.ShadowMonsters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ShadowMonsters.Items
{
    public interface IItem
    {
        string Name { get; }
        int Price { get; }
        bool Usable { get; }
        void Apply(ShadowMonster monster);
    }
}
```

To implement the interface you must have a read property for the name of the item, the price of the item, if it is usable and a source rectangle. You must also have a method that will apply the item to a shadow monster.

Let's create a couple items. Right click the Items folder, select Add and then Class. Name this new class Potion. Here is the code.

```
using Microsoft.Xna.Framework;
using ShadowMonsters.ShadowMonsters;

namespace ShadowMonsters.Items
{
    public class Potion : IItem
    {
        public string Name { get { return "Potion"; } }
        public int Price { get { return 200; } }
        public bool Usable => true;

        public void Apply(ShadowMonster monster)
        {
            monster.Heal(50);
        }
    }
}
```

The class implements the IItem interface. The properties return the values for the item. The Apply method calls the Heal method of the shadow monster passing in 50 for the amount to heal.

Let's add another item. Right click the Items folder, select Add and then Class. Name this new class Antidote. Here is the code.

```
using ShadowMonsters.ShadowMonsters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ShadowMonsters.Items
{
    public class Antidote : IItem
    {
        public string Name => "Antidote";

        public int Price => 1000;

        public bool Usable => true;
        public void Apply(ShadowMonster monster)
        {
            monster.IsPoisoned = false;
        }
    }
}
```

Similar to the Potion class. The difference is that instead of calling the Heal method I set the IsPoisoned property to false. Next we need a container to hold the player's items. Right click the Items folder, select Add and then Class. Name this new class Backpack. Here is the code.

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

namespace ShadowMonsters.Items
{
    public class Item
    {
        public string Name;
        public int Count;
    }

    public class Backpack
    {
        private readonly List<Item> items = new List<Item>();

        public List<Item> Items { get => items; }


        public Backpack()
        {
        }

        public void AddItem(IItem item, int count)
        {
            if (items.Any(x => x.Name == item.Name))
            {
                for (int i = 0; i < items.Count; i++)
                {
                    if (items[i].Name == item.Name)
                    {
                        items[i].Count = items[i].Count + count;
                    }
                }
            }
            else
            {
                Item i = new Item()
                {
                    Name = item.Name,
                    Count = count
                };

                items.Add(i);
            }
        }

        public void AddItem(string item, int count)
        {
            if (items.Any(x => x.Name == item))
            {
                for (int i = 0; i < items.Count; i++)
                {
                    if (items[i].Name == item)
                    {
                        items[i].Count = items[i].Count + count;
                    }
                }
            }
            else
            {
                Item i = new Item()
                {
                    Name = item,
                    Count = count
                };

                items.Add(i);
            }
        }

        public IItem GetItem(string item)
        {
            if (!items.Any(x => x.Name == item))
            {
                return null;
            }

            for (int i = 0; i < items.Count; i++)
            {
                if (items[i].Name == item)
                {
                    items[i].Count--;

                    if (items[i].Count < 1)
                    {
                        items.RemoveAt(i);
                    }

                    break;
                }
            }

            switch (item)
            {
                case "Potion":
                    return new Potion();
                case "Antidote":
                    return new Antidote();
                default:
                    return null;
            }
        }

        public IItem PeekItem(string item)
        {
            if (!items.Any(x => x.Name == item))
            {
                return null;
            }

            switch (item)
            {
                case "Potion":
                    return new Potion();
                case "Antidote":
                    return new Antidote();
                default:
                    return null;
            }
        }
    }
}
```

There is a helper class, Item, with two public fields: Name and Count. They hold the name of the item and the number of that item in the backpack, There is just one field in the backpack and that is a List<Item> that holds the items in the backpack. There is a read only property to expose its value.

The first AddItem method takes an IItem parameter and an integer parameter that represent the item to be added and the number of that item. It first checks to see if there are any items with that name in the collection. If there are it loops over the items in the collection. When the item with that name is found it increments the count of that item. If it is not found a new item is created and added to the collection. The overload of the AddItem method works the same way but takes a string instead of an IItem as a parameter.

The GetItem method is used to get an item out of the backpack. It first checks to see if there is an item with that name in the backpack and if not returns null. It then searches the backpack for the item and decreases the count. If the count is less than one the item is removed from the backpack. Next there is a switch on the name of the item that returns an item of that type. If no item with that name is found null is returned.

The PeekItem method works the same way as the GetItem method does. The difference is it does not remove the item from the backpack. It is used for displaying the backpack's contents.

Next I'm going to add in a merchant for the player to buy items from. Right click the Characters folder in the Solution Explorer, select Add and then Class. Name this new class Merchant. Here is the code.

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using ShadowMonsters.Items;
using ShadowMonsters.ShadowMonsters;
using ShadowMonsters.TileEngine;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ShadowMonsters.Characters
{
    public class Merchant : Character
    {
        private Backpack backpack;
        public Backpack Backpack => backpack;

        private Merchant()
            : base()
        {
        }
        public new static Merchant FromString(Game game, string characterString)
        {
            if (gameRef == null)
            {
                gameRef = game;
            }

            Merchant character = new Merchant();
            character.backpack = new Backpack();

            string[] parts = characterString.Split(',');

            character.name = parts[0];
            character.textureName = parts[1];
            character.sprite = new AnimatedSprite(
                game.Content.Load<Texture2D>(@"CharacterSprites\" + parts[1]),
                Game1.Animations)
            {
                CurrentAnimation = (AnimationKey)Enum.Parse(typeof(AnimationKey), parts[2])
            };
            character.conversation = parts[3];
            character.currentMonster = int.Parse(parts[4]);

            for (int i = 5; i < 11 && i < parts.Length - 1; i++)
            {
                ShadowMonster monster =
ShadowMonsterManager.GetShadowMonster(parts[i].ToLowerInvariant());
                character.monsters[i - 5] = monster;
            }

            character.givingMonster = ShadowMonsterManager.GetShadowMonster(parts[parts.Length
- 1].ToLowerInvariant());
            return character;
        }
    }
}
```

The class inherits from Character and has access to all of its protected, public and internal members. There is a parameterless constructor that will be used in loading and saving merchants. There is a new FromString method that is a copy of the other except that it creates a backpack.

Let's create a merchant to buy good from. Open the GamePlayState and change the LoadContent method to the following.

```
protected override void LoadContent()
{
    MoveManager.FillMoves();
    ShadowMonsterManager.FromFile(@".\Content\ShadowMonsters.txt", content);
    Game1.Player.AddShadowMonster(ShadowMonsterManager.GetShadowMonster("water1"));
    Game1.Player.SetCurrentShadowMonster(0);
    Game1.Player.BattleShadowMonsters[0] = Game1.Player.GetShadowMonster(0);
    TileSet set = new TileSet();
    set.TextureNames.Add("tileset1");
    set.Textures.Add(content.Load<Texture2D>(@"Tiles\tileset16-outdoors"));

    TileLayer groundLayer = new TileLayer(100, 100, 0, 1);
    TileLayer edgeLayer = new TileLayer(100, 100);
    TileLayer buildingLayer = new TileLayer(100, 100);
    TileLayer decorationLayer = new TileLayer(100, 100);

    for (int i = 0; i < 1000; i++)
    {
        decorationLayer.SetTile(random.Next(0, 100), random.Next(0, 100), 0, random.Next(2,
4));
    }

    map = new TileMap(set, groundLayer, edgeLayer, buildingLayer, decorationLayer, "level1");

    Character c = Character.FromString(GameRef,
"Paul,ninja_m,WalkDown,PaulHello,0,fire1,fire1,,,,,,dark1");
    c.Sprite.Position = new Vector2(2 * Engine.TileWidth, 2 * Engine.TileHeight);

    map.CharacterLayer.Characters.Add(new Point(2, 2), c);

    Merchant m = Merchant.FromString(GameRef,
"Bonnie,ninja_f,WalkLeft,BonnieHello,0,earth1,earth1,,,,,,");
    m.Sprite.Position = new Vector2(4 * Engine.TileWidth, 4 * Engine.TileHeight);
    m.Backpack.AddItem("Potion", 99);
    m.Backpack.AddItem("Antidote", 10);

    map.CharacterLayer.Characters.Add(new Point(4, 4), m);
    engine.SetMap(map);
    base.LoadContent();
}
```

The changes are I create a merchant, Bonnie, using the FromString method like I did for Paul, I gave her a couple shadow monsters just because I could. I set the position of the character to tile (4, 4). I then add some potions to her backpack and some antidotes. Then I add her to the character layer.

Now to give her a conversation. Update the CreateConversatioins method of the ConversationManager to the following.

```
public void CreateConversations(Game gameRef)
{
    ConversationList.Clear();
    Conversation c = new Conversation("PaulHello", "Hello")
    {
        BackgroundName = "scenebackground",
    };

    List<SceneOption> options = new List<SceneOption>();
```

```
SceneOption teach = new SceneOption(
    "Teach",
    "Teach",
    new SceneAction() { Action = ActionType.Teach, Parameter = "none" });
options.Add(teach);

SceneOption option = new SceneOption(
    "Good bye.",
    "",
    new SceneAction() { Action = ActionType.End, Parameter = "none" });
options.Add(option);

GameScene scene = new GameScene(
    gameRef,
    "Hello, my name is Paul. I'm still learning about training shadow monsters.",
    options);

c.AddScene("Hello", scene);

options = new List<SceneOption>();

scene = new GameScene(
    gameRef,
    "I have given you Brownie!",
    options);

option = new SceneOption(
    "Goodbye",
    "",
    new SceneAction() { Action = ActionType.End, Parameter = "none" });

options.Add(option);

c.AddScene("Teach", scene);
ConversationList.Add("PaulHello", c);

c = new Conversation("BonnieHello", "Hello");

options = new List<SceneOption>();

option = new SceneOption(
    "Shop",
    "",
    new SceneAction() { Action = ActionType.Shop, Parameter = "none" });

options.Add(option);

option = new SceneOption(
    "Goodbye",
    "",
    new SceneAction() { Action = ActionType.End, Parameter = "none" });

options.Add(option);

scene = new GameScene(
    gameRef,
    "Hi! I'm Bonnie. Feel free to browse my wares.",
    options);

c.AddScene("Hello", scene);
```

```
    ConversationList.Add("BonnieHello", c);
}
```

The new code creates a new conversation,, BonnieHello, with a first scene Hello, It then creates a new list of scene options. I add an option Shop to open the shopping interface. I then create a End option to exit out of the conversation. I create a scene for the conversation and add it to the conversation. The conversation is then added to the conversation list.

The player needs a backpack. Add the following field and property to the player class and update the constructors to the following.

```
private Backpack backpack;
public Backpack Backpack { get => backpack; }

private Player(Game game)
    : base(game)
{
    backpack = new Backpack();
}

public Player(Game game, string name, bool gender, string textureName)
    : base(game)
{
    gameRef = (Game1)game;
    this.name = name;
    this.gender = gender;

    this.textureName = textureName;
    sprite = new AnimatedSprite(
        game.Content.Load<Texture2D>(textureName),
        Game1.Animations)
    {
        CurrentAnimation = AnimationKey.WalkDown
    };
    Gold = 1000;
    backpack = new Backpack();
}
```

Now we have everything we need to create the shop interface. It is a game state so right click the GameStates folder, select Add and then Class. Name this new class ShopState. Here is the code.

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Input;
using ShadowMonsters.Characters;
using ShadowMonsters.ConversationComponents;
using ShadowMonsters.Items;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ShadowMonsters.GameStates
{
    public enum ShopStateType { Buy, Sell, Talk }
    public interface IShopState
    {
        ShopStateType State { get; set; }
        void SetMerchant(Merchant merchant);
    }

    public class ShopState : BaseGameState, IShopState
    {
        public Dictionary<string, int> Inventory = new Dictionary<string, int>();
        private GameScene scene;
        private Merchant merchant;
        private int selected;
        private bool mouseOver;
        private bool isFirst;

        public ShopStateType State { get; set; }

        public ShopState(Game game)
            : base(game)
        {
            State = ShopStateType.Talk;
        }

        protected override void LoadContent()
        {
            base.LoadContent();

            scene = new GameScene(GameRef, "", new List<SceneOption>());
            SceneOption option = new SceneOption("Buy", "Buy", new SceneAction());
            scene.Options.Add(option);

            option = new SceneOption("Sell", "Sell", new SceneAction());
            scene.Options.Add(option);

            option = new SceneOption("Leave", "Leave", new SceneAction());
            scene.Options.Add(option);
        }

        public override void Update(GameTime gameTime)
        {
            base.Update(gameTime);

            scene.Update();

            switch (State)
            {
                case ShopStateType.Buy:
                    if (isFirst)
                    {
                        isFirst = false;
                        break;
                    }

                    if (Xin.CheckKeyReleased(Keys.Down) ||
                        Xin.CheckKeyReleased(Keys.S))
                    {
                        selected++;
                        if (selected >= merchant.Backpack.Items.Count)
                        {
                            selected = 0;
                        }
                    }

                    if (Xin.CheckKeyReleased(Keys.Up) ||
                        Xin.CheckKeyReleased(Keys.W))
                    {
                        selected--;
                        if (selected < 0)
                        {
                            selected = merchant.Backpack.Items.Count - 1;
                        }
                    }


                    if (Xin.CheckKeyReleased(Keys.Space) ||
                        Xin.CheckKeyReleased(Keys.Enter) ||
                        (Xin.CheckMouseReleased(MouseButtons.Left) && mouseOver))
                    {
                        if (selected >= 0 &&
                            Game1.Player.Gold >=
                            merchant.Backpack.PeekItem(
                                merchant.Backpack.Items[selected].Name).Price)
                        {
                            Game1.Player.Backpack.AddItem(
                                merchant.Backpack.GetItem(
                                    merchant.Backpack.Items[selected].Name),
                                1);
                            Game1.Player.Gold -=
                                merchant.Backpack.PeekItem(
                                    merchant.Backpack.Items[selected].Name).Price;
                        }
                    }
                    break;
                case ShopStateType.Sell:
                    if (isFirst)
                    {
                        isFirst = false;
                        break;
                    }
                    if (Xin.CheckKeyReleased(Keys.Down) ||
                        Xin.CheckKeyReleased(Keys.S))
                    {
                        selected++;

                        if (selected >= Game1.Player.Backpack.Items.Count)
                        {
                            selected = 0;
                        }
                    }

                    if (Xin.CheckKeyReleased(Keys.Up) ||
                        Xin.CheckKeyReleased(Keys.W))
                    {
                        selected--;
                        if (selected < 0)
                        {
                            selected = Game1.Player.Backpack.Items.Count - 1;
                        }
                    }
```

```
                    if ((Xin.CheckKeyReleased(Keys.Space) ||
                        Xin.CheckKeyReleased(Keys.Enter) ||
                        (Xin.CheckMouseReleased(MouseButtons.Left) && mouseOver)))
                    {
                        if (selected >= 0)
                        {
                            IItem item = Game1.Player.Backpack.GetItem(
                                Game1.Player.Backpack.Items[selected].Name);
                            Game1.Player.Gold += item.Price * 3 / 4;
                        }
                    }
                    break;
                case ShopStateType.Talk:
                    if (Xin.CheckKeyReleased(Keys.Space) ||
                        Xin.CheckKeyReleased(Keys.Enter))
                    {
                        Xin.FlushInput();

                        if (scene.SelectedIndex == 0)
                        {
                            isFirst = true;
                            State = ShopStateType.Buy;
                            //selected = -1;
                            return;
                        }

                        if (scene.SelectedIndex == 1)
                        {
                            isFirst = true;
                            State = ShopStateType.Sell;
                            //selected = -1;
                            return;
                        }

                        if (scene.SelectedIndex == 2 && State == ShopStateType.Talk)
                        {
                            manager.PopState();
                        }
                    }
                    break;
            }

            if (Xin.CheckMouseReleased(MouseButtons.Right) ||
Xin.CheckKeyReleased(Keys.Escape))
            {
                switch (State)
                {
                    case ShopStateType.Buy:
                    case ShopStateType.Sell:
                        State = ShopStateType.Talk;
                        break;
                }
            }
        }

        public override void Draw(GameTime gameTime)
        {
            base.Draw(gameTime);
```

```
GameRef.SpriteBatch.Begin();

int i = 0;
Color tint;

switch (State)
{
    case ShopStateType.Buy:
        mouseOver = false;
        if (isFirst)
        {
            break;
        }
        GameRef.SpriteBatch.DrawString(
            FontManager.GetFont("testfont"),
            "Item",
            new Vector2(120, 5),
            Color.Red);

        GameRef.SpriteBatch.DrawString(
            FontManager.GetFont("testfont"),
            "Quantity",
            new Vector2(800, 5),
            Color.Red);

        GameRef.SpriteBatch.DrawString(
            FontManager.GetFont("testfont"),
            "Price",
            new Vector2(1100, 5),
            Color.Red);

        foreach (var v in merchant.Backpack.Items)
        {
            tint = Color.White;
            if (i == selected)
            {
                tint = Color.Red;
            }

            IItem item = merchant.Backpack.PeekItem(v.Name);

            if (item != null)
            {
                Rectangle r = new Rectangle(40, 74 * i + 24, 1200, 64);

                if (r.Contains(Xin.MouseAsPoint))
                {
                    selected = i;
                    mouseOver = true;
                }

                GameRef.SpriteBatch.DrawString(
                    FontManager.GetFont("testfont"),
                    v.Name,
                    new Vector2(120, 74 * i + 45),
                    tint);

                GameRef.SpriteBatch.DrawString(
                    FontManager.GetFont("testfont"),
                    v.Count.ToString(),
```

```csharp
                            new Vector2(800, 74 * i + 45),
                            tint);

                        GameRef.SpriteBatch.DrawString(
                            FontManager.GetFont("testfont"),
                            item.Price.ToString(),
                            new Vector2(1100, 74 * i + 45),
                            tint);

                        i++;
                    }
                }
                break;
            case ShopStateType.Sell:
                mouseOver = false;
                if (isFirst)
                {
                    break;
                }
                GameRef.SpriteBatch.DrawString(
                    FontManager.GetFont("testfont"),
                    "Item",
                    new Vector2(120, 5),
                    Color.Red);

                GameRef.SpriteBatch.DrawString(
                    FontManager.GetFont("testfont"),
                    "Quantity",
                    new Vector2(800, 5),
                    Color.Red);

                GameRef.SpriteBatch.DrawString(
                    FontManager.GetFont("testfont"),
                    "Price",
                    new Vector2(1100, 5),
                    Color.Red);

                foreach (var v in Game1.Player.Backpack.Items)
                {
                    tint = Color.White;

                    if (i == selected)
                    {
                        tint = Color.Red;
                    }

                    IItem item = Game1.Player.Backpack.PeekItem(v.Name);

                    if (item != null)
                    {
                        Rectangle r = new Rectangle(40, 74 * i + 24, 1200, 64);

                        if (r.Contains(Xin.MouseAsPoint))
                        {
                            selected = i;
                            mouseOver = true;
                        }

                        GameRef.SpriteBatch.DrawString(
                            FontManager.GetFont("testfont"),
                                v.Name,
                                new Vector2(120, 74 * i + 45),
                                tint);

                            GameRef.SpriteBatch.DrawString(
                                FontManager.GetFont("testfont"),
                                v.Count.ToString(),
                                new Vector2(800, 74 * i + 45),
                                tint);

                            GameRef.SpriteBatch.DrawString(
                                FontManager.GetFont("testfont"),
                                item.Price.ToString(),
                                new Vector2(1100, 74 * i + 45),
                                tint);

                            i++;
                        }
                    }
                    break;
                case ShopStateType.Talk:
                    scene.Draw(GameRef.SpriteBatch, null);
                    break;
            }

            GameRef.SpriteBatch.End();
        }

        public void SetMerchant(Merchant merchant)
        {
            this.merchant = merchant;
        }
    }
}
```

Apologies for the wall of code. Let's break it down. There is an enumeration that tells us what state the shop is in. The player is either buying, selling or talking to the merchant. There is an interface that the class will be implementing. It has a property that gets or sets the state of the shop and a method that sets the merchant. The class inherits from BaseGameState and implements the interface.

There are five fields in the class. The first, scene, is displayed when the player is talking to the merchant. The merchant field is the merchant we are talking to. The selected field is the currently selected item for buying or selling. The isFirst field is similar to the frameCount field and is used to prevent bleeding of key/mouse presses. There is also the property from the interface. The constructor sets the state to talk. The LoadContent method creates the scene. I do it a little differently. I create the options and add them to the scene directly rather than creating a list of options then the scene.

The Update method calls the Update method of the scene to update it. Next there is a switch on the State property. If the state is buy I check the isFirst field is true. If it is I set it to false and break out of the case. Next I check if the down or S keys have been pressed. If they have I move to the next item in the the merchant's backpack wrapping to the first if we are at the end of the list. I move in the opposite direction if the up or W keys have been pressed. If the space bar or enter keys have been pressed or the mouse is over and the left button has been pressed I check to see if there is a selected item and that the player has enough money to buy the item. If they do I add the item to the player's backpack and deduct the price from their gold. The Sell case works similarly but on the player's backpack instead of the merchant's backpack. When selling I give the player 75% of the item's value. The talk case checks to see if the space bar or enter keys have been pressed. If they have I call FlushInput to reset the input. If the selected state is 0 or Buy I set isFirst to true and State to Buy then exit the method. I do something similar for the Sell option. Finally if we are in the Talk state I pop the state off of the stack. If the right mouse button or escape key have been pressed I go to the Talk state.

The Draw method renders the backpacks and the scene. There are two local variables. The first is a counter and the second determines what color an item in the backpack is drawn in. There is a switch on the state. In the Buy case mouseOver is set to false. If this is the first time the state is being drawn break out of the case. Next I position the title for screen. I then loop over all of the items in the merchant's backpack. The tint is set to white then if the counter is equal to the selected item it is set to red. I then peek the item. If the item is not null I create a rectangle around the item. If it contains the mouse selected is set to the counter and mouse over to true. Finally I draw the item and increment the counter. The Sell option works the same way but on the player's backpack. The Talk state just draws the scene.

The SetMerchant method just sets the merchant field. Now we need to implement the new state. The first step is to create a field in the Game1 class and expose it with a property then create an instance. Add the following field and property and update the constructor of the Game1 class.

```
private readonly ShopState shopState;
public ShopState ShopState => shopState;

public Game1()
{
    graphics = new GraphicsDeviceManager(this)
    {
        PreferredBackBufferWidth = 1280,
        PreferredBackBufferHeight = 720
    };

    graphics.ApplyChanges();

    Content.RootDirectory = "Content";

    stateManager = new GameStateManager(this);
    Components.Add(stateManager);

    gamePlayState = new GamePlayState(this);
    conversationState = new ConversationState(this);
    levelUpState = new LevelUpState(this);
    damageState = new DamageState(this);
    battleOverState = new BattleOverState(this);
    battleState = new BattleState(this);
    actionSelectionState = new ActionSelectionState(this);
    shadowMonsterSelectionState = new ShadowMonsterSelectionState(this);
    startBattleState = new StartBattleState(this);
    shopState = new ShopState(this);
    stateManager.PushState(gamePlayState);
    ConversationManager.Instance.CreateConversations(this);
    IsMouseVisible = true;
}
```

The next thing I'm going to tackle in this tutorial is triggering the shop state. That will be done in the ConversationState. What we want to do is if the Shop action is selected is set the merchant to the spaeker, pop the conversation state off of the stack and push the shop state onto the stack. Update the Update method of the ConversationState class to the following.

```
public override void Update(GameTime gameTime)
{
    if (conversation.CurrentScene == null)
    {
        return;
    }
    frameCount++;
    if ((Xin.CheckKeyReleased(Keys.Space) ||
        Xin.CheckKeyReleased(Keys.Enter) ||
        (Xin.CheckMouseReleased(MouseButtons.Left) &&
        conversation.CurrentScene.IsMouseOver)) && frameCount > 5)
    {
        frameCount = 0;
        switch (conversation.CurrentScene.OptionAction.Action)
        {
            case ActionType.Teach:
                BuyShadowMonster();
                break;
            case ActionType.Change:
                speaker.SetConversation(conversation.CurrentScene.OptionScene);
                manager.PopState();
                break;
            case ActionType.End:
                manager.PopState();
                break;
            case ActionType.GiveItems:
                break;
            case ActionType.GiveKey:
                if (conversation.CurrentScene.OptionAction.Parameter != null)
                {
                    bool success = int.TryParse(
                        conversation.CurrentScene.OptionAction.Parameter,
                        out int key);

                    if (success)
                    {
                    }

                    conversation.ChangeScene(conversation.CurrentScene.OptionScene);
                }
                break;
            case ActionType.Quest:
                CheckQuest();
                break;
            case ActionType.Rest:
                conversation.ChangeScene(conversation.CurrentScene.OptionScene);
                break;
            case ActionType.Shop:
                GameRef.ShopState.SetMerchant((Merchant)speaker);
                manager.PopState();
                manager.PushState(GameRef.ShopState);
                break;
            case ActionType.Talk:
                conversation.ChangeScene(conversation.CurrentScene.OptionScene);
                break;
        }
    }

    conversation.Update();
    base.Update(gameTime);
}
```

Now I want to implement a state for applying an item to a selected shadow monster. Right click the GameStates folder, select Add and then Class. Name this new class UseItemState. Here is the code.

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using ShadowMonsters.Items;
using ShadowMonsters.ShadowMonsters;

namespace ShadowMonsters.GameStates
{
    public interface IUseItemState
    {
        void SetItem(IItem item);
    }

    public class UseItemState : BaseGameState, IUseItemState
    {
        private IItem item;
        private bool mouseOver;
        private Texture2D shadowMonsterBorder;
        private Texture2D shadowMonsterHealth;
        private int selected;

        public UseItemState(Game game)
            : base(game)
        {
        }
        protected override void LoadContent()
        {
            base.LoadContent();

            shadowMonsterBorder = new Texture2D(GraphicsDevice, 300, 75);
            shadowMonsterHealth = new Texture2D(GraphicsDevice, 300, 25);
            Color[] buffer = new Color[300 * 75];

            for (int i = 0; i < buffer.Length; i++)
            {
                buffer[i] = Color.Green;
            }

            shadowMonsterBorder.SetData(buffer);

            buffer = new Color[300 * 25];
            for (int i = 0; i < buffer.Length; i++)
            {
                buffer[i] = Color.Red;
            }

            shadowMonsterHealth.SetData(buffer);
        }

        public override void Update(GameTime gameTime)
        {
            if (Xin.CheckKeyReleased(Keys.Up) || Xin.CheckKeyReleased(Keys.W))
            {
                selected--;

                if (selected < 0)
                {
                    selected = Game1.Player.BattleShadowMonsters.Length - 1;
                }
            }

            if (Xin.CheckKeyReleased(Keys.Down) || Xin.CheckKeyReleased(Keys.S))
            {
                selected++;

                if (selected >= Game1.Player.BattleShadowMonsters.Length)
                {
                    selected = 0;
                }
            }

            if (Xin.CheckKeyReleased(Keys.Space) ||
                Xin.CheckKeyReleased(Keys.Enter) ||
                (mouseOver && Xin.CheckMouseReleased(MouseButtons.Left)))
            {
                if (Game1.Player.BattleShadowMonsters[selected] != null)
                {
                    item.Apply(Game1.Player.BattleShadowMonsters[selected]);
                    manager.PopState();
                    manager.PopState();
                }
            }

            if (Xin.CheckMouseReleased(MouseButtons.Right) ||
Xin.CheckKeyReleased(Keys.Escape))
            {
                manager.PopState();
            }

            base.Update(gameTime);
        }

        public override void Draw(GameTime gameTime)
        {
            GameRef.SpriteBatch.Begin();

            base.Draw(gameTime);

            Color tint;
            Rectangle destination = new Rectangle(50, 20, 100, 100);
            Rectangle playerBorderRect = new Rectangle(250, 20, 400, 100);
            Rectangle playerHealthRect = new Rectangle(
                playerBorderRect.X + 16,
                playerBorderRect.Y + 73, 350, 19);
            Vector2 playerName = new Vector2(325, 25);
            Rectangle healthSourceRect = new Rectangle(10, 50, 290, 20);
            Point cursor = Xin.MouseAsPoint;

            for (int i = 0; i < Game1.Player.BattleShadowMonsters.Length; i++)
            {
                tint = Color.White;

                if (i == selected)
                {
                    tint = Color.Red;
                }
                if (Game1.Player.BattleShadowMonsters[i] != null)
                {
                    ShadowMonster a = Game1.Player.BattleShadowMonsters[i];
                    GameRef.SpriteBatch.Draw(a.Texture, destination, Color.White);

                    if (destination.Contains(cursor) || playerBorderRect.Contains(cursor))
                    {
                        selected = i;
                        mouseOver = true;
                    }

                    GameRef.SpriteBatch.Draw(shadowMonsterBorder, playerBorderRect,
Color.White);
                    GameRef.SpriteBatch.DrawString(
                        FontManager.GetFont("testfont"),
                        a.DisplayName,
                        playerName,
                        tint);
                    float playerHealth = (float)a.CurrentHealth / (float)a.GetHealth();
                    MathHelper.Clamp(playerHealth, 0f, 1f);
                    playerHealthRect.Width = (int)(playerHealth * 384);
                    GameRef.SpriteBatch.Draw(
                        shadowMonsterHealth,
                        playerHealthRect,
                        healthSourceRect,
                        Color.White);
                    playerBorderRect.Y += 120;
                    playerName.Y += 120;
                    playerHealthRect.Y += 120;
                }

                destination.Y += 120;
            }

            GameRef.SpriteBatch.End();
        }

        public void SetItem(IItem item)
        {
            this.item = item;
        }
    }
}
```

There is an interface that has a single method SetItem that passes an item to the state. The class inherits from BaseGameState and implements the interface. For fields there is the item that will be applied to the shadow monster, a mouse over field to tell if the mouse is over a shadow monster, two Texture2D for drawing the shadow monster and a selected field for what shadow monster is currently selected.

The constructor takes a Game parameter because the base class requires a Game parameter. The LoadContent method creates two Texture2Ds on the fly. It creates the fields then sets their color using the SetData method.

The Update method checks if the Up or W key have been pressed and if they have move the selection up one wrapping to the last shadow monster in the list. It does the same for the Down or S key. If space, enter, or the left mouse button have been pressed and the mouse is over a shadow monster I apply the item to the shadow monster if it is not null an pop the two states off the stack. If the right mouse button or escape keys have been pressed I pop the state off the stack.

The Draw method works the same as the ShadowMonsterSelectionState. It sets mouseOver to false. It loops over the shadow monsters. I set the tint color to white. If the current shadow monster is the selected shadow monster I set the tint color to red. If the mouse is over the current shadow monster I set the selected index to the loop counter and mouseOver to true. I then draw the shadow monster.

There is one more state to add and that is an item selection state that allows the player to select the item to use. Right click the GameStates folder, select Add and then Class. Name this new class ItemSelectionState. Here is the code.

```
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework;
using ShadowMonsters.Items;

namespace ShadowMonsters.GameStates
{
    public interface IItemSelectionState
    {
        int SelectedIndex { get; }
    }

    public class ItemSelectionState : BaseGameState, IItemSelectionState
    {
        private int selected;
        private bool mouseOver;

        public int SelectedIndex
        {
            get { return selected; }
        }

        public ItemSelectionState(Game game)
            : base(game)
        {
        }

        protected override void LoadContent()
        {
            base.LoadContent();
        }

        public override void Update(GameTime gameTime)
        {
            base.Update(gameTime);

            if (Xin.CheckKeyReleased(Keys.Down) ||
                Xin.CheckKeyReleased(Keys.S))
            {
                selected++;

                if (selected >= Game1.Player.Backpack.Items.Count)
                {
                    selected = 0;
                }
            }

            if (Xin.CheckKeyReleased(Keys.Up) ||
                Xin.CheckKeyReleased(Keys.W))
            {
                selected--;

                if (selected < 0)
                {
                    selected = Game1.Player.Backpack.Items.Count - 1;
                }
            }

            if ((Xin.CheckKeyReleased(Keys.Space) ||
                Xin.CheckKeyReleased(Keys.Enter) ||
                (Xin.CheckMouseReleased(MouseButtons.Left) &&
                mouseOver)) &&
                selected >= 0 &&
                Game1.Player.Backpack.Items.Count > 0 &&
                Game1.Player.Backpack.PeekItem(
                    Game1.Player.Backpack.Items[selected].Name).Usable)
            {
                GameRef.UseItemState.SetItem(
                    Game1.Player.Backpack.GetItem(
                        Game1.Player.Backpack.Items[selected].Name));
                manager.PushState((UseItemState)GameRef.UseItemState);
            }

            if (Xin.CheckMouseReleased(MouseButtons.Right) ||
Xin.CheckKeyReleased(Keys.Escape))
            {
                manager.PopState();
            }
        }

        public override void Draw(GameTime gameTime)
        {
            Color tint;
            int i = 0;

            base.Draw(gameTime);

            GameRef.SpriteBatch.Begin();
            GameRef.SpriteBatch.DrawString(
                FontManager.GetFont("testfont"),
                "Item",
                new Vector2(120, 5),
                Color.Red);

            GameRef.SpriteBatch.DrawString(
                FontManager.GetFont("testfont"),
                "Quantity",
                new Vector2(800, 5),
                Color.Red);

            GameRef.SpriteBatch.DrawString(
                FontManager.GetFont("testfont"),
                "Price",
                new Vector2(1100, 5),
                Color.Red);

            foreach (var v in Game1.Player.Backpack.Items)
            {
                tint = Color.White;

                if (i == selected)
                {
                    tint = Color.Red;
                }

                IItem item = Game1.Player.Backpack.PeekItem(v.Name);

                if (item != null)
                {
                    Rectangle r = new Rectangle(40, 74 * i + 24, 1200, 64);

                    if (r.Contains(Xin.MouseAsPoint))
                    {
                        selected = i;
                        mouseOver = true;
                    }

                    GameRef.SpriteBatch.DrawString(
                        FontManager.GetFont("testfont"),
                        v.Name,
                        new Vector2(120, 74 * i + 45),
                        tint);

                    GameRef.SpriteBatch.DrawString(
                        FontManager.GetFont("testfont"),
                        v.Count.ToString(),
                        new Vector2(800, 74 * i + 45),
                        tint);

                    GameRef.SpriteBatch.DrawString(
                        FontManager.GetFont("testfont"),
                        item.Price.ToString(),
                        new Vector2(1100, 74 * i + 45),
                        tint);

                    i++;
                }
```

```
            }

            GameRef.SpriteBatch.End();
        }
    }
}
```

There is an interface that includes a single property, SelectedIndex, that is the index of the chosen item. The class inherits from BaseGameState and implements the interface. There are two fields: selected and mouseOver. The selected field is the currently selected item and mouseOver is if the mouse is over an item. The constructor takes a Game parameter because the base class takes a game parameter.

The Update method works pretty much the same as the ShopState. If Up or W have been pressed move the selection up one wrapping to the last item. Similarly moving down if the Down or S key have been released. If the space or enter or left mouse button have been released I push the UseItemState on the stack and set the item to the item in the backpack. If the right mouse button or escape key have been released I pop the state off the stack.

The Draw method works the same as drawing the sell state of the shop state. If you are confused I recommend going back and rereading it.

We need to update the Game1 class with the new states. Add the following two fields, properties and change the constructor.

```
private readonly ItemSelectionState itemSelectionState;
private readonly UseItemState useItemState;

public UseItemState UseItemState => useItemState;
public ItemSelectionState ItemSelectionState => itemSelectionState;

public static Dictionary<AnimationKey, Animation> Animations => animations;

public Game1()
{
    graphics = new GraphicsDeviceManager(this)
    {
        PreferredBackBufferWidth = 1280,
        PreferredBackBufferHeight = 720
    };

    graphics.ApplyChanges();

    Content.RootDirectory = "Content";

    stateManager = new GameStateManager(this);
    Components.Add(stateManager);

    gamePlayState = new GamePlayState(this);
    conversationState = new ConversationState(this);
    levelUpState = new LevelUpState(this);
    damageState = new DamageState(this);
    battleOverState = new BattleOverState(this);
    battleState = new BattleState(this);
    actionSelectionState = new ActionSelectionState(this);
    shadowMonsterSelectionState = new ShadowMonsterSelectionState(this);
    startBattleState = new StartBattleState(this);
    shopState = new ShopState(this);
    itemSelectionState = new ItemSelectionState(this);
    useItemState = new UseItemState(this);

    stateManager.PushState(gamePlayState);
    ConversationManager.Instance.CreateConversations(this);
    IsMouseVisible = true;
}
```

The new states will be implemented in two places. The first is the ActionSelectionState and the second is the GamePlayState. In the ActionSelectionState the Update method needs to push the ItemSelectionState onto the stack if the Item option is selected. Update that method to the following.

```
public override void Update(GameTime gameTime)
{
    base.Update(gameTime);
    frameCount++;
    scene.Update();

    if (Xin.CheckKeyReleased(Keys.Space) || Xin.CheckKeyReleased(Keys.Enter) && frameCount >=
5)

    {
        frameCount = 0;
        manager.PopState();

        if (scene.SelectedIndex == 0)
        {
            // do nothing
        }

        if (scene.SelectedIndex == 1)
        {

manager.PushState((ShadowMonsterSelectionState)GameRef.ShadowMonsterSelectionState);
        }

        if (scene.SelectedIndex == 2)
        {
            manager.PushState((ItemSelectionState)GameRef.ItemSelectionState);
        }

        if (scene.SelectedIndex == 3)
        {
            manager.ChangeState(GameRef.GamePlayState);
        }
    }
}
```

In the GamePlayState I check to see if the I key has been released in the Update method. If it has I push the ItemSelectionState onto the stack. Change the Update method of the GamePlayState to the following.

```
public override void Update(GameTime gameTime)
{
    engine.Update(gameTime);
```

```
frameCount++;
if (Xin.KeyboardState.IsKeyDown(Keys.W) && !inMotion)
{
    motion.Y = -1;
    Game1.Player.Sprite.CurrentAnimation = AnimationKey.WalkUp;
    Game1.Player.Sprite.IsAnimating = true;
    inMotion = true;
    collision = new Rectangle(
        (int)Game1.Player.Sprite.Position.X,
        (int)Game1.Player.Sprite.Position.Y - Engine.TileHeight * 2,
        Engine.TileWidth,
        Engine.TileHeight);
}
else if (Xin.KeyboardState.IsKeyDown(Keys.S) && !inMotion)
{
    motion.Y = 1;
    Game1.Player.Sprite.CurrentAnimation = AnimationKey.WalkDown;
    Game1.Player.Sprite.IsAnimating = true;
    inMotion = true;
    collision = new Rectangle(
        (int)Game1.Player.Sprite.Position.X,
        (int)Game1.Player.Sprite.Position.Y + Engine.TileHeight * 2,
        Engine.TileWidth,
        Engine.TileHeight);
}
else if (Xin.KeyboardState.IsKeyDown(Keys.A) && !inMotion)
{
    motion.X = -1;
    Game1.Player.Sprite.CurrentAnimation = AnimationKey.WalkLeft;
    Game1.Player.Sprite.IsAnimating = true;
    inMotion = true;
    collision = new Rectangle(
        (int)Game1.Player.Sprite.Position.X - Engine.TileWidth * 2,
        (int)Game1.Player.Sprite.Position.Y,
        Engine.TileWidth,
        Engine.TileHeight);
}
else if (Xin.KeyboardState.IsKeyDown(Keys.D) && !inMotion)
{
    motion.X = 1;
    Game1.Player.Sprite.CurrentAnimation = AnimationKey.WalkRight;
    Game1.Player.Sprite.IsAnimating = true;
    inMotion = true;
    collision = new Rectangle(
        (int)Game1.Player.Sprite.Position.X + Engine.TileWidth * 2,
        (int)Game1.Player.Sprite.Position.Y,
        Engine.TileWidth,
        Engine.TileHeight);
}

if (motion != Vector2.Zero)
{
    motion.Normalize();
    motion *= (Game1.Player.Sprite.Speed * (float)gameTime.ElapsedGameTime.TotalSeconds);
    Rectangle pRect = new Rectangle(
            (int)(Game1.Player.Sprite.Position.X + motion.X),
            (int)(Game1.Player.Sprite.Position.Y + motion.Y),
            Engine.TileWidth,
            Engine.TileHeight);
    if (pRect.Intersects(collision))
    {
        Game1.Player.Sprite.IsAnimating = false;
        inMotion = false;
        motion = Vector2.Zero;
    }

    foreach (Point p in engine.Map.CharacterLayer.Characters.Keys)
    {
        Rectangle r = new Rectangle(
            p.X * Engine.TileWidth,
            p.Y * Engine.TileHeight,
            Engine.TileWidth,
            Engine.TileHeight);

        if (r.Intersects(pRect))
        {
            motion = Vector2.Zero;
            Game1.Player.Sprite.IsAnimating = false;
            inMotion = false;
        }
    }

    Vector2 newPosition = Game1.Player.Sprite.Position + motion;
    newPosition.X = (int)newPosition.X;
    newPosition.Y = (int)newPosition.Y;

    Game1.Player.Sprite.Position = newPosition;
    motion = Game1.Player.Sprite.LockToMap(
        new Point(
            map.WidthInPixels,
            map.HeightInPixels),
        motion);

    if (motion == Vector2.Zero)
    {
        Vector2 origin = new Vector2(
                Game1.Player.Sprite.Position.X + Game1.Player.Sprite.Origin.X,
                Game1.Player.Sprite.Position.Y + Game1.Player.Sprite.Origin.Y);
        Game1.Player.Sprite.Position = Engine.VectorFromOrigin(origin);
        inMotion = false;
        Game1.Player.Sprite.IsAnimating = false;
    }
}

if ((Xin.CheckKeyReleased(Keys.Space) ||
    Xin.CheckKeyReleased(Keys.Enter)) && frameCount >= 5)
{
    frameCount = 0;
    foreach (Point s in engine.Map.CharacterLayer.Characters.Keys)
    {
        Character c = engine.Map.CharacterLayer.Characters[s];

        AnimationKey animation = Game1.Player.Sprite.CurrentAnimation;

        if (animation == AnimationKey.WalkLeft &&
            ((int)c.Sprite.Position.X > (int)Game1.Player.Sprite.Position.X ||
                (int)c.Sprite.Position.Y != (int)Game1.Player.Sprite.Position.Y))
        {
            continue;
        }

        if (animation == AnimationKey.WalkUp &&
            ((int)c.Sprite.Position.X != (int)Game1.Player.Sprite.Position.X ||
                (int)c.Sprite.Position.Y > (int)Game1.Player.Sprite.Position.Y))
        {
            continue;
        }

        if (animation == AnimationKey.WalkRight &&
            ((int)c.Sprite.Position.X < (int)Game1.Player.Sprite.Position.X ||
                (int)c.Sprite.Position.Y != (int)Game1.Player.Sprite.Position.Y))
        {
            continue;
        }

        if (animation == AnimationKey.WalkDown &&
            ((int)c.Sprite.Position.X != (int)Game1.Player.Sprite.Position.X ||
                (int)c.Sprite.Position.Y < (int)Game1.Player.Sprite.Position.Y))
        {
            continue;
        }

        float distance = Vector2.Distance(
            Game1.Player.Sprite.Origin + Game1.Player.Sprite.Position,
            c.Sprite.Origin + c.Sprite.Position);

        if (Math.Abs(distance) < Engine.TileWidth + Engine.TileWidth / 2)
        {
            manager.PushState(
                (ConversationState)GameRef.ConversationState);

            GameRef.ConversationState.SetConversation(c);
            GameRef.ConversationState.StartConversation();
            break;
        }
    }
}

if (Xin.CheckKeyReleased(Keys.I))
{
    manager.PushState(GameRef.ItemSelectionState);
}

if ((Xin.CheckKeyReleased(Keys.B)) && frameCount >= 5)
{
    frameCount = 0;
    foreach (Point s in engine.Map.CharacterLayer.Characters.Keys)
    {
        Character c = engine.Map.CharacterLayer.Characters[s];

        AnimationKey animation = Game1.Player.Sprite.CurrentAnimation;

        if (animation == AnimationKey.WalkLeft &&
            ((int)c.Sprite.Position.X > (int)Game1.Player.Sprite.Position.X ||
                (int)c.Sprite.Position.Y != (int)Game1.Player.Sprite.Position.Y))
        {
            continue;
        }
```

```
            if (animation == AnimationKey.WalkUp &&
                ((int)c.Sprite.Position.X != (int)Game1.Player.Sprite.Position.X ||
                    (int)c.Sprite.Position.Y > (int)Game1.Player.Sprite.Position.Y))
            {
                continue;
            }

            if (animation == AnimationKey.WalkRight &&
                ((int)c.Sprite.Position.X < (int)Game1.Player.Sprite.Position.X ||
                    (int)c.Sprite.Position.Y != (int)Game1.Player.Sprite.Position.Y))
            {
                continue;
            }

            if (animation == AnimationKey.WalkDown &&
                ((int)c.Sprite.Position.X != (int)Game1.Player.Sprite.Position.X ||
                    (int)c.Sprite.Position.Y < (int)Game1.Player.Sprite.Position.Y))
            {
                continue;
            }

            float distance = Vector2.Distance(
                Game1.Player.Sprite.Origin + Game1.Player.Sprite.Position,
                c.Sprite.Origin + c.Sprite.Position);

            if (Math.Abs(distance) < Engine.TileWidth + Engine.TileWidth / 2 && c.Alive())
            {
                GameRef.StartBattleState.SetCombatants(Game1.Player, c);
                manager.PushState(GameRef.StartBattleState);
                break;
            }
        }
    }

    Engine.Camera.LockToSprite(map, Game1.Player.Sprite, new Rectangle(0, 0, 1280, 720));
    Game1.Player.Update(gameTime);

    base.Update(gameTime);
}
```

That is it for adding basic items to the game. This system can be extended to include more items and different types of item that aren't applied to shadow monsters.

I'm going to wrap this tutorial up here. I will be starting work on the next tutorial shortly. Keep checking back on the blog for news on that tutorial. I hope to have it up in the next week or so.

I wish you the best in your MonoGame Programming Adventures!