Shadow Monsters – MonoGame Tutorial Series
Chapter 9
Battling Shadow Monsters Part Two

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.

The last tutorial added in battles between shadow monsters but it was only one shadow monster versus another shadow monster. This tutorial will cover battling all six, or however many the player is carrying, shadow monsters.

To get started I want to add a state that allows the player to select one of the shadow monsters they are carrying. Right click the GameStates folder in the SolutionExplorer, select Add and then Class. Name this new class ShadowMonsterSelectionState. Here is the code.

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

namespace ShadowMonsters.GameStates
{
    public interface IShadowMonsterSelectionState
    {
        int Selected { get; }
    }

    public class ShadowMonsterSelectionState : BaseGameState, IShadowMonsterSelectionState
    {
        private bool mouseOver;
        private Texture2D shadowMonsterBorder;
        private Texture2D shadowMonsterHealth;
```

```
private int Selected1;

public int Selected
{
    get { return Selected1; }
}

public int Selected1 { get; set; }

public ShadowMonsterSelectionState(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))
    {
        Selected1--;

        if (Selected1 < 0)
            Selected1 = Game1.Player.BattleShadowMonsters.Length - 1;
    }

    if (Xin.CheckKeyReleased(Keys.Down) || Xin.CheckKeyReleased(Keys.S))
    {
        Selected1++;

        if (Selected1 >= Game1.Player.BattleShadowMonsters.Length)
            Selected1 = 0;
    }

    if (Xin.CheckKeyReleased(Keys.Space) ||
        Xin.CheckKeyReleased(Keys.Enter) ||
        (mouseOver && Xin.CheckMouseReleased(MouseButtons.Left)))
```

```
            {
                if (Game1.Player.BattleShadowMonsters[Selected1] != null &&
                    Game1.Player.BattleShadowMonsters[Selected1].Alive)
                {
                    Game1.Player.SetCurrentShadowMonster(Selected1);
                    GameRef.BattleState.ChangePlayerShadowMonster(Game1.Player.Selected);
                    GameRef.ActionSelectionState.SetShadowMonsters(
                        Game1.Player.Selected,
                        ((BattleState)GameRef.BattleState).EnemyShadowMonster);
                    manager.PopState();
                }
            }

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

            base.Update(gameTime);
        }

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

            base.Draw(gameTime);

            Color tint = Color.White;
            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))
                    {
                        Selected1 = 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();
        }
    }
}
```

This class implements an interface with a read only property that is the index of the selected shadow monster. For fields there is a mouseOver field to tell if the mouse is over a shadow monster. There are also two Texture2D fields similar to those in the previous tutorial that are used to draw a border around a health bar. Finally there is a field to hold the selected shadow monster. There is just the single property to return the selected index. The LoadContent method creates the textures on the fly like we did in the last tutorial.

The Update method checks if the Up arrow was pressed. If it was the selected index is decremented. If it is less than zero it is bumped to the last index minus one. Similarly for the down arrow the selection is incremented and if it is greater than or equal to the last index it is set to zero. Next there is a check to see if the space bar or enter key have been released or if the mouse is over a shadow monster and the left button has been released. It that is true there is a check to see that the selected spot is not null and that the shadow monster is alive. In this case I call the SetCurrentShadowMonster method of the player class passing in the selected index. Then I call the ChangePlayerShadowMonster method of the battle state to switch the selected shadow monster. I then call the SetShadowMonster method of the ActionSelectionState that I will implement next and pop the state off of the stack. There is also a check to see if the escape key has been released that pops the state off of the stack.

The Draw method has several local variables. The tint variable determines what color text is drawn in. The destination variable is for the destination of a thumbnail of the shadow monster. The playerBorderRect and playerHealthRect are similar to the fields from the last tutorial. They control where the border and health bar are drawn. The playerName variable is where the name of the shadow monster will be drawn, The healthSourceRect is an artifact from my game where I had a texture that I created for the health bar. Finally cursor is the position of the mouse.

I loop over all of the shadow monsters that the player could possibly be carrying, I set the tint

variable to white, Then if i is equal to the selected index I set the tint to red. If the shadow monster at that index is not null I grab the shadow monster and draw it. If the mouse is over either the thumbnail or border I set selected to the index and mouseOver to true. I then draw the rest of the shadow monster. Finally I increment the destination rectangle's Y coordinate.

Now I'm going to implement the ActionSelectionState. It works like in Pokemon where you are given the choice to fight, switch pokemon, use an item or run away. Right click the GameStates folder in the Solution Explorer, select Add and then Class. Name this new class ActionSelectionState. Here is the code.

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

namespace ShadowMonsters.GameStates
{
    public interface IActionSelectionState
    {
        void SetShadowMonsters(ShadowMonster player, ShadowMonster enemy);
    }

    public class ActionSelectionState : BaseGameState, IActionSelectionState
    {
        private ShadowMonster player;
        private ShadowMonster enemy;
        private GameScene scene;
        private Texture2D background;
        private Rectangle playerRect;
        private Rectangle enemyRect;
        private Rectangle playerBorderRect;
        private Rectangle enemyBorderRect;
        private Rectangle playerMiniRect;
        private Rectangle enemyMiniRect;
        private Rectangle playerHealthRect;
        private Rectangle enemyHealthRect;
        private Rectangle healthSourceRect;
        private Vector2 playerName;
        private Vector2 enemyName;
        private float playerHealth;
        private float enemyHealth;
        private Texture2D ShadowMonsterBorder;
        private Texture2D ShadowMonsterHealth;
        private int frameCount = 0;

        public ActionSelectionState(Game game) : base(game)
        {
            playerRect = new Rectangle(10, 90, 300, 300);
            enemyRect = new Rectangle(game.Window.ClientBounds.Width - 310, 10, 300, 300);

            playerBorderRect = new Rectangle(10, 10, 300, 75);
            enemyBorderRect = new Rectangle(game.Window.ClientBounds.Width - 310, 320, 300,
```

```
75);

            healthSourceRect = new Rectangle(10, 50, 290, 20);
            playerHealthRect = new Rectangle(playerBorderRect.X + 12, playerBorderRect.Y + 52,
286, 16);
            enemyHealthRect = new Rectangle(enemyBorderRect.X + 12, enemyBorderRect.Y + 52,
286, 16);

            playerMiniRect = new Rectangle(playerBorderRect.X + 11, playerBorderRect.Y + 11,
28, 28);
            enemyMiniRect = new Rectangle(enemyBorderRect.X + 11, enemyBorderRect.Y + 11, 28,
28);

            playerName = new Vector2(playerBorderRect.X + 55, playerBorderRect.Y + 5);
            enemyName = new Vector2(enemyBorderRect.X + 55, enemyBorderRect.Y + 5);
        }

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

            background = new Texture2D(GraphicsDevice, 1280, 720);
            Color[] buffer = new Color[1280 * 720];

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

            background.SetData(buffer);

            ShadowMonsterBorder = new Texture2D(GraphicsDevice, 300, 75);
            ShadowMonsterHealth = new Texture2D(GraphicsDevice, 300, 25);

            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);

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

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

            option = new SceneOption("Item", "Item", new SceneAction());
```

```
            scene.Options.Add(option);

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

        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)
                {
                }

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

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

            GameRef.SpriteBatch.Begin();

            scene.Draw(GameRef.SpriteBatch, background);

            GameRef.SpriteBatch.Draw(player.Texture, playerRect, Color.White);
            GameRef.SpriteBatch.Draw(enemy.Texture, enemyRect, Color.White);

            GameRef.SpriteBatch.Draw(ShadowMonsterBorder, playerBorderRect, Color.White);

            playerHealth = (float)player.CurrentHealth / (float)player.GetHealth();
            MathHelper.Clamp(playerHealth, 0f, 1f);
            playerHealthRect.Width = (int)(playerHealth * 286);

            GameRef.SpriteBatch.Draw(ShadowMonsterHealth, playerHealthRect, healthSourceRect,
Color.White);
```

```
            GameRef.SpriteBatch.Draw(ShadowMonsterBorder, enemyBorderRect, Color.White);

            enemyHealth = (float)enemy.CurrentHealth / (float)enemy.GetHealth();
            MathHelper.Clamp(enemyHealth, 0f, 1f);
            enemyHealthRect.Width = (int)(enemyHealth * 286);

            GameRef.SpriteBatch.Draw(
                ShadowMonsterHealth,
                enemyHealthRect,
                healthSourceRect,
                Color.White);
            GameRef.SpriteBatch.DrawString(
                FontManager.GetFont("testfont"),
                player.DisplayName,
                playerName,
                Color.White);
            GameRef.SpriteBatch.DrawString(
                FontManager.GetFont("testfont"),
                enemy.DisplayName,
                enemyName,
                Color.White);

            GameRef.SpriteBatch.Draw(
                player.Texture,
                playerMiniRect,
                player.Source,
                Color.White);
            GameRef.SpriteBatch.Draw(
                enemy.Texture,
                enemyMiniRect,
                enemy.Source,
                Color.White);

            GameRef.SpriteBatch.End();
        }

        public void SetShadowMonsters(ShadowMonster player, ShadowMonster enemy)
        {
            this.player = player;
            this.enemy = enemy;

            player.StartCombat();
            enemy.StartCombat();
        }
    }
}
```

There is another interface here that we will be implementing that has a single method SeetShadowMonsters like in the BattleState. The class inherits from BaseGameState and implements the interface. There are the same fields as the BattleState and the constructor initializes them. There is also a frameCount field to prevent bleeding through of selections. The LoadContent method creates the textures and the GameScene.

The Update method increments the frameCount field and updates the scene. It then checks to see if the space bar or enter key have been released. If they have it resets frameCount to zero and pops the state off of the stack. If the selected index is 0 we currently don't have to do anything. If it is one we push the ShadowMonsterSelectionState on the stack. When we have

implemented items selected index two will push the item selection state onto the stack. If it is three we change states to the GamePlayState.

The Draw method works like the Draw method of the BattleState. It just positions the items and draws them.

There is one last state to implement. This is the StartBattleState and it controls the flow of combat with multiple shadow monsters. Right click the GameStates folder, select Add and then Class. Name this new class StartBattleState, Here is the code.

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

namespace ShadowMonsters.GameStates
{
    public interface IStartBattleState
    {
        void SetCombatants(Player player, Character character);
    }

    public class StartBattleState : BaseGameState, IStartBattleState
    {
        Player player;
        Character character;

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

        public override void Update(GameTime gameTime)
        {
            if (character.BattleMonster.CurrentHealth <= 0)
            {
                if (!character.NextMonster())
                {
                    manager.ChangeState(GameRef.GamePlayState);
                }
                else
                {
                    GameRef.BattleState.SetShadowMonsters(player.Selected,
character.BattleMonster);
                    GameRef.ActionSelectionState.SetShadowMonsters(player.Selected,
character.BattleMonster);
                    manager.PushState((BattleState)GameRef.BattleState);
                    manager.PushState((ActionSelectionState)GameRef.ActionSelectionState);
                }
            }
            else if (player.Selected.CurrentHealth <= 0)
            {
                if (!player.Alive())
                {
                    manager.ChangeState(GameRef.GamePlayState);
                }
```

```
                else
                {

manager.PushState((ShadowMonsterSelectionState)GameRef.ShadowMonsterSelectionState);
                }
            }
            else
            {
                GameRef.BattleState.SetShadowMonsters(player.Selected,
character.BattleMonster);
                GameRef.ActionSelectionState.SetShadowMonsters(player.Selected,
character.BattleMonster);
                manager.PushState((BattleState)GameRef.BattleState);
                manager.PushState((ActionSelectionState)GameRef.ActionSelectionState);
            }

            base.Update(gameTime);
        }

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

        public void SetCombatants(Player player, Character character)
        {
            this.player = player;
            this.character = character;

            if (!player.Alive())
            {
                manager.PopState();
                return;
            }

            if (!character.Alive())
            {
                manager.PopState();
                return;
            }

            GameRef.BattleState.SetShadowMonsters(player.Selected, character.BattleMonster);
            GameRef.ActionSelectionState.SetShadowMonsters(player.Selected,
character.BattleMonster);
            manager.PushState((BattleState)GameRef.BattleState);
            manager.PushState((ActionSelectionState)GameRef.ActionSelectionState);
        }
    }
}
```

This class also has an interface. This interface has a single method SetCombatants that takes a Player object and Character object as parameters. The class inherits from BaseGameState and implements the interface.

The Update method checks to see if the character's current shadow monster has health less than or equal to zero, If it does it calls NextMonster that will return false if there is no next monster. If there is no monster I change state to the game play state. If there is a shadow monster I call the SetShadowMonster methos of the BattleState and ActionSelectionState

then push the battle state on the stack and the action selection state. If the player's current shadow monster has fainted I check to see if the player has any shadow monsters. If they do not I change state to the game play state. If they do I just the ShadowMonsterSelectionState onto the stack. Otherwise I call the SetShadowMonsters method of the battle state and action selection state and push them onto the stack in that order.

The SetCombatants method sets the player and character fields. It checks to see if the player is not alive. If it is not I pop the state off of the stack, I then do the same check on the character and pop the state off the stack if it is not alive. The next step is to call the SetShadowMonster methods of the battle state and action selection state and push them onto the stack in that order.

The next step will be to implement the new states. First they need to be added to the Game1 class. It is just adding fields, properties and initializing them in the constructor. Here is the code for the Game1 class.

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using ShadowMonsters.ConversationComponents;
using ShadowMonsters.GameStates;
using ShadowMonsters.TileEngine;
using System;
using System.Collections.Generic;

namespace ShadowMonsters
{
    /// <summary>
    /// This is the main type for your game.
    /// </summary>
    public class Game1 : Game
    {
        public static Player Player;
        public static Random Random = new Random();
        private static Dictionary<AnimationKey, Animation> animations = new
Dictionary<AnimationKey, Animation>();
        private readonly GraphicsDeviceManager graphics;
        private SpriteBatch spriteBatch;
        private readonly GamePlayState gamePlayState;
        private readonly ConversationState conversationState;
        private readonly LevelUpState levelUpState;
        private readonly BattleOverState battleOverState;
        private readonly DamageState damageState;
        private readonly BattleState battleState;
        private readonly ActionSelectionState actionSelectionState;
        private readonly ShadowMonsterSelectionState shadowMonsterSelectionState;
        private readonly StartBattleState startBattleState;
        private readonly GameStateManager stateManager;

        public SpriteBatch SpriteBatch => spriteBatch;
        public GamePlayState GamePlayState => gamePlayState;
        public ConversationState ConversationState => conversationState;
        public LevelUpState LevelUpState => levelUpState;
        public BattleOverState BattleOverState => battleOverState;
        public DamageState DamageState => damageState;
        public BattleState BattleState => battleState;
        public ActionSelectionState ActionSelectionState => actionSelectionState;
```

```
        public ShadowMonsterSelectionState ShadowMonsterSelectionState =>
shadowMonsterSelectionState;
        public StartBattleState StartBattleState => startBattleState;

        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);

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

        /// <summary>
        /// Allows the game to perform any initialization it needs to before starting to run.
        /// This is where it can query for any required services and load any non-graphic
        /// related content.  Calling base.Initialize will enumerate through any components
        /// and initialize them as well.
        /// </summary>
        protected override void Initialize()
        {
            // TODO: Add your initialization logic here

            Animation animation = new Animation(3, 32, 36, 0, 0);
            animations.Add(AnimationKey.WalkUp, animation);

            animation = new Animation(3, 32, 36, 0, 36);
            animations.Add(AnimationKey.WalkRight, animation);

            animation = new Animation(3, 32, 36, 0, 72);
            animations.Add(AnimationKey.WalkDown, animation);

            animation = new Animation(3, 32, 36, 0, 108);
            animations.Add(AnimationKey.WalkLeft, animation);

            Components.Add(new Xin(this));
            Components.Add(new FontManager(this));
```

```
            Game1.Player = new Player(this, "Bonnie", true, @"Sprites\mage_f");
            base.Initialize();
        }

        /// <summary>
        /// LoadContent will be called once per game and is the place to load
        /// all of your content.
        /// </summary>
        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);

            // TODO: use this.Content to load your game content here
        }

        /// <summary>
        /// UnloadContent will be called once per game and is the place to unload
        /// game-specific content.
        /// </summary>
        protected override void UnloadContent()
        {
            // TODO: Unload any non ContentManager content here
        }

        /// <summary>
        /// Allows the game to run logic such as updating the world,
        /// checking for collisions, gathering input, and playing audio.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Update(GameTime gameTime)
        {
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(Keys.Escape))
                Exit();

            // TODO: Add your update logic here
            base.Update(gameTime);
        }

        /// <summary>
        /// This is called when the game should draw itself.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.CornflowerBlue);

            // TODO: Add your drawing code here

            base.Draw(gameTime);
        }
    }
}
```

Now to implement this in the game. I will do it in the GamePlayState class. First I want to update Paul so that he has two shadow monsters and will give the player a shadow monster. Update 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);

    engine.SetMap(map);
    base.LoadContent();
}
```

Now to update the Update method to push the StartBattleState onto the stack instead of the BattleState when the player triggers combat. There is also a check to see if the character is alive before pushing the state on the stack. Change the Update method 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.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);
}
```

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!