Eyes of the Dragon Tutorials 4.0

Part 13

Mechanical Engineering, Well Mechanics – Part Two

I'm writing these tutorials for the MonoGame 3.8.1 framework using Visual Studio 2022. The tutorials will make more sense if they are read in order. You can find the list of tutorials on the Eyes of the Dragon 4.0 page of my web blog. I will be making the project available on GitHub here. It will be included on the page that links to the tutorials.

This is part thirteen of a series of tutorials I plan to write on creating a role-playing game with MonoGame. I've worked on similar tutorials using XNA and the past version of MonoGame. In the process of writing more tutorials, I discovered better ways of doing some things and had to go back to fix things. I'm hoping in this series to avoid making those same mistakes. Also, I am going to make the game cross-platform. In my previous tutorials, they were focused on Windows only. I want to open things up for macOS, Linux, Android, iOS, and any platform MonoGame supports.

This tutorial is picking up where the last one finished. First, I need to address a little errata. I have a typing mistake in the ICharacter interface. I have Stength instead of Strength. Easy fix as we haven't done much work with the interface. Just replace it with this version.

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

namespace RpgLibrary.Characters
{
    public struct AttributePair
    {
        public int Current;
        public int Maximum;

        public AttributePair()
        {
            Current = 10;
            Maximum = 10;
        }

        public AttributePair(int maximum)
        {
            Current = Maximum = maximum;
        }

        public void Adjust(int amount)
        {
            Current += amount;
```

```
            if (Current > Maximum)
            {
                Current = Maximum;
            }
        }
    }

    public interface ICharacter
    {
        string Name { get; }

        int Strength { get; set; }
        int Perception { get; set; }
        int Endurance { get; set; }
        int Charisma { get; set; }
        int Intellect { get; set; }
        int Agility { get; set; }
        int Luck { get; set; }

        AttributePair Health { get; set; }
        AttributePair Mana { get; set; }

        int Gold { get; set; }
        int Experience { get; set; }

        bool Enabled { get; set; }
        bool Visible { get; set; }
        Vector2 Position { get; set; }
        Point Tile { get; set; }
        void Update(GameTime gameTime);
        void Draw(SpriteBatch spriteBatch);
    }
}
```

We also need to update the Villager and Character classes. Since they are short, I will provide the new code. First, the Villager class.

```
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework;
using RpgLibrary.Characters;
using RpgLibrary.TileEngine;
using SharedProject.Sprites;
using System;
using System.Collections.Generic;
using System.Text;

namespace SharedProject.Characters
{
    public class Villager : ICharacter
    {
        public string Name { get; set; }
        public AnimatedSprite AnimatedSprite { get; private set; }
        public Point Tile { get; set; }
        public Rectangle Bounds { get; private set; }
        public string Conversation { get; set; }
        public string Scene { get; set; }
        public bool Enabled { get; set; }
        public bool Visible { get; set; }
```

```
        public Vector2 Position { get; set; }
        public int Strength { get; set; }
        public int Perception { get; set; }
        public int Endurance { get; set; }
        public int Charisma { get; set; }
        public int Intellect { get; set; }
        public int Agility { get; set; }
        public int Luck { get; set; }
        public AttributePair Health { get; set; }
        public AttributePair Mana { get; set; }
        public int Gold { get; set; }
        public int Experience { get; set; }

        public Villager(AnimatedSprite sprite, Point tile)
        {
            Enabled = true;
            Visible = true;
            Position = new();
            Tile = new();
            AnimatedSprite = sprite;
            Tile = tile;
            Bounds = new(Engine.PointToWorld(Tile), new(Engine.TileWidth, Engine.Tile-
Height));
        }

        public void Update(GameTime gameTime)
        {
            AnimatedSprite.Position = Position;
            AnimatedSprite.Update(gameTime);
        }

        public void Draw(SpriteBatch spriteBatch)
        {
            AnimatedSprite.Draw(spriteBatch);
        }
    }
}
```

The same change to the Character class.

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using RpgLibrary.Characters;
using SharedProject.Sprites;
using System;
using System.Collections.Generic;
using System.Text;

namespace SharedProject
{
    public class Character : ICharacter
    {
        private string _name;
        private AnimatedSprite _sprite;
        private string _spriteName;

        public string Name => _name;

        public bool Enabled { get; set; }
```

```
        public bool Visible { get; set; }
        public Vector2 Position { get; set; }
        public Point Tile { get; set; }
        public int Strength { get; set; }
        public int Perception { get; set; }
        public int Endurance { get; set; }
        public int Charisma { get; set; }
        public int Intellect { get; set; }
        public int Agility { get; set; }
        public int Luck { get; set; }
        public AttributePair Health { get; set; }
        public AttributePair Mana { get; set; }
        public int Gold { get; set; }
        public int Experience { get; set; }

        private Character()
        {
            Enabled = true;
            Visible = true;
            Position = new();
            Tile = new();
        }

        public Character(string name, AnimatedSprite animatedSprite, string spriteName)
        {
            _name = name;
            _sprite = animatedSprite;
            _spriteName = spriteName;
        }

        public void Draw(SpriteBatch spriteBatch)
        {
            _sprite.Draw(spriteBatch);
        }

        public void Update(GameTime gameTime)
        {
            _sprite.Position= Position;
            _sprite.Update(gameTime);
        }
    }
}
```

Now, back to the main tutorial. I want to add a few extension methods. These methods will be on the SpriteBatch class. What they will do is draw using a Point instead of a Vector2 for the destination. This will make it easier to render some things. So, add these three methods to the ExtensionMethods class.

```
public static void Draw(this SpriteBatch spriteBatch, Texture2D texture, Point
destination, Rectangle? source, Color color)
{
    spriteBatch.Draw(texture, new Vector2(destination.X, destination.Y), source, color);
}

public static void Draw(this SpriteBatch spriteBatch, Texture2D texture, Point
destination, Color color)
{
    spriteBatch.Draw(texture, new Vector2(destination.X, destination.Y), color);
}
```

```
public static void DrawString(this SpriteBatch spriteBatch, SpriteFont font, string text,
Point destination, Color tint)
{
    spriteBatch.DrawString(font, text, new Vector2(destination.X, destination.Y), tint);
}
```

This will allow us to render directly using points rather than having to do the casting inside the class that is doing the rendering. As you can see, they create a Vector2 behind the scene rather than having to do it ourselves.

Now, I will add the idea of a player to the gameplay state. We had a Player property in the GamePlayState, but it didn't do anything. This tutorial will address that. To start, we will want to update the IGamePlayState interface. What we want to do is add a new property that will implement a read/write property for the player interface. Then, in the GamePlayState class, change the property to have public get and set accessors. Finally, change the IGamePlayState interface to the following.

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

namespace SharedProject.GamesScreens
{
    public interface IGamePlayState
    {
        GameState Tag { get; }
        Player Player { get; set; }
    }
}
```

Now, I will have the IPlayer interface implement the ICharacter interface. "What? Are you crazy, girl?" I hear you asking. Yes, we can have an interface implement another interface. That interface can be referenced as the base interface or the interface being implemented, the same as polymorphism in classes. Update the IPlayer interface to the following.

```
    public interface IPlayer : ICharacter
    {
        AnimatedSprite Sprite { get; }
    }
```

What I did was have the interface implement the ICharacter interface. Because ICharacter has a Name property, I removed the Name property from the interface. What I also did is add an AnimatedSprite property because the player has a sprite associated with it.

Now, let's update the Player class. What we want to do is update it to require an AnimatedSprite and implement the new IPlayer interface that implemented ICharacter. Replace the Character class with the following code.

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using RpgLibrary.Characters;
using SharedProject.Sprites;
using System;
```

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

namespace SharedProject
{
    public interface IPlayer : ICharacter
    {
        AnimatedSprite Sprite { get; }
    }

    public sealed class Player : DrawableGameComponent, IPlayer
    {
        private string _name;
        private AnimatedSprite _sprite;

        public Player(Game game, AnimatedSprite animatedSprite) : base(game)
        {
            Game.Services.AddService<IPlayer>(this);
            _sprite = animatedSprite;
        }

        public string Name => _name;

        public int Stength { get; set; }
        public int Perception { get; set; }
        public int Endurance { get; set; }
        public int Charisma { get; set; }
        public int Intellect { get; set; }
        public int Agility { get; set; }
        public int Luck { get; set; }
        public AttributePair Health { get; set; }
        public AttributePair Mana { get; set; }
        public int Gold { get; set; }
        public int Experience { get; set; }
        public Vector2 Position { get; set; }
        public Point Tile { get; set; }

        public AnimatedSprite Sprite => _sprite;

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

            _sprite.Update(gameTime);
        }

        public void Draw(SpriteBatch spriteBatch)
        {
            _sprite.Draw(spriteBatch);
        }
    }
}
```

A very straightforward class. The IPlayer interface implements the ICharacter interface to give the IPlayer interface all of the methods and properties. This way, the player object can be retrieved and manipulated as a service. The interface has a property that will be the player's sprite.

As I mentioned in a previous tutorial, this class is sealed, so no class can derive from it. It has the

possibility of making methods sealed, which increases performance slightly. It has two fields. One for the name and another for the sprite. The constructor takes as an additional argument the sprite for the player. It registers itself as a service and sets the sprite field.

Next, there are the properties that must be implemented. They are read/write except for Name and Sprite, which are read-only. The Update method calls the base update method and then the update method of the sprite. The Draw method just renders the sprite.

I made a few updates to the LoadContent method of the NewGameState class. I changed the portrait selector to select the class of the player rather than portraits. Also, I changed the maximum width of the selector so it lines up with the gender selector. Replace the LoadContent method of the New GameState with the following

```
        protected override void LoadContent()
        {
            SpriteBatch = Game.Services.GetService<SpriteBatch>();
            renderTarget2D = new(GraphicsDevice, Settings.TargetWidth, Settings.Tar-
getHeight);

            ContentManager content = Game.Content;

            string[] items = new[] { "2", "3", "4", "5", "6", "7", "8", "9" };

            controls = new(content.Load<SpriteFont>(@"Fonts/MainFont"), 100);
            _genderSelector = new(
                content.Load<Texture2D>(@"GUI\g22987"),
                content.Load<Texture2D>(@"GUI\g21245"))
            {
                Position = new Vector2(207 - 70, 298)
            };

            _genderSelector.SelectionChanged += GenderSelector_SelectionChanged;
            _genderSelector.SetItems(new[] { "Female", "Male" }, 300);

            _portraitSelector = new(
                content.Load<Texture2D>(@"GUI\g22987"),
                content.Load<Texture2D>(@"GUI\g21245"))
            {
                Position = new Vector2(207 - 70, 458)
            };

            _portraitSelector.SelectionChanged += PortraitSelector_SelectionChanged;
            _portraitSelector.SetItems(new[] { "Fighter", "Wizard", "Rogue", "Priest" },
300);

            _pointsLabel = new()
            {
                Text = "Points to spend: ",
                Position = new(700, 20),
                Color = Color.White,
            };

            _remainingLabel = new()
            {
                Text = "22",
                Position = new(900, 20),
```

```
        Color = Color.Red,
    };

    _strengthLabel = new Label()
    {
        Color = Color.White,
        Text = "Strength",
        Position = new(700, 75)
    };

    _strengthSelector = new RightLeftSelector(
        content.Load<Texture2D>(@"GUI\g22987"),
        content.Load<Texture2D>(@"GUI\g21245"))
    {
        Position = new(900, 75)
    };

    _strengthSelector.SetItems(items, 75);
    _strengthSelector.SelectionChanged += Ability_SelectorChanged;

    _perceptionLabel = new Label()
    {
        Color = Color.White,
        Text = "Perception",
        Position = new(700, 150)
    };

    _perceptionSelector = new RightLeftSelector(
        content.Load<Texture2D>(@"GUI\g22987"),
        content.Load<Texture2D>(@"GUI\g21245"))
    {
        Position = new(900, 150)
    };

    _perceptionSelector.SetItems(items, 75);
    _perceptionSelector.SelectionChanged += Ability_SelectorChanged;

    _enduranceLabel = new Label()
    {
        Color = Color.White,
        Text = "Endurance",
        Position = new(700, 225)
    };

    _enduranceSelector = new RightLeftSelector(
        content.Load<Texture2D>(@"GUI\g22987"),
        content.Load<Texture2D>(@"GUI\g21245"))
    {
        Position = new(900, 225)
    };

    _enduranceSelector.SetItems(items, 75);
    _enduranceSelector.SelectionChanged += Ability_SelectorChanged;

    _charismaLabel = new Label()
    {
        Color = Color.White,
        Text = "Charisma",
        Position = new(700, 300)
    };
```

```
_charismaSelector = new RightLeftSelector(
    content.Load<Texture2D>(@"GUI\g22987"),
    content.Load<Texture2D>(@"GUI\g21245"))
{
    Position = new(900, 300)
};

_charismaSelector.SetItems(items, 75);
_charismaSelector.SelectionChanged += Ability_SelectorChanged;

_intellectLabel = new Label()
{
    Color = Color.White,
    Text = "Intellect",
    Position = new(700, 375)
};

_intellectSelector = new RightLeftSelector(
    content.Load<Texture2D>(@"GUI\g22987"),
    content.Load<Texture2D>(@"GUI\g21245"))
{
    Position = new(900, 375)
};

_intellectSelector.SetItems(items, 75);
_intellectSelector.SelectionChanged += Ability_SelectorChanged;

_agilityLabel = new Label()
{
    Color = Color.White,
    Text = "Agility",
    Position = new(700, 450)
};

_agilitySelector = new RightLeftSelector(
    content.Load<Texture2D>(@"GUI\g22987"),
    content.Load<Texture2D>(@"GUI\g21245"))
{
    Position = new(900, 450)
};

_agilitySelector.SetItems(items, 75);
_agilitySelector.SelectionChanged += Ability_SelectorChanged;

_luckLabel = new Label()
{
    Color = Color.White,
    Text = "Luck",
    Position = new(700, 525)
};

_luckSelector = new RightLeftSelector(
    content.Load<Texture2D>(@"GUI\g22987"),
    content.Load<Texture2D>(@"GUI\g21245"))
{
    Position = new(900, 525)
};

_luckSelector.SetItems(items, 75);
```

```
_luckSelector.SelectionChanged += Ability_SelectorChanged;

Texture2D background = new(GraphicsDevice, 100, 25);
Texture2D caret = new(GraphicsDevice, 2, 25);
Texture2D border = new(GraphicsDevice, 100, 25);

caret.Fill(Color.Black);
border.Fill(Color.Black);
background.Fill(Color.White);

_nameTextBox = new(background, caret, border)
{
    Position = new(207, 138),
    HasFocus = true,
    Enabled = true,
    Color = Color.Black,
    Text = "Bethany",
    Size = new(100, 25)
};

_femalePortraits.Add(
"Female 0",
    content.Load<Texture2D>(@"PlayerSprites/femalefighter"));
_femalePortraits.Add(
"Female 1",
    content.Load<Texture2D>(@"PlayerSprites/femalepriest"));
_femalePortraits.Add(
"Female 2",
    content.Load<Texture2D>(@"PlayerSprites/femalerogue"));
_femalePortraits.Add(
"Female 3",
    content.Load<Texture2D>(@"PlayerSprites/femalewizard"));

_malePortraits.Add(
"Male 0",
    content.Load<Texture2D>(@"PlayerSprites/malefighter"));
_malePortraits.Add(
"Male 1",
    content.Load<Texture2D>(@"PlayerSprites/malepriest"));
_malePortraits.Add(
"Male 2",
    content.Load<Texture2D>(@"PlayerSprites/malerogue"));
_malePortraits.Add(
"Male 3",
    content.Load<Texture2D>(@"PlayerSprites/malewizard"));


_portraitSelector.SetItems(_femalePortraits.Keys.ToArray(), 270);
_create = new(
    content.Load<Texture2D>(@"GUI\g9202"),
    ButtonRole.Accept)
{
    Text = "Create",
    Position = new(180, 640)
};

_create.Click += Create_Click;
_back = new(
    content.Load<Texture2D>(@"GUI\g9202"),
    ButtonRole.Cancel)
```

```
            {
                Text = "Back",
                Position = new(350, 640),
                Color = Color.White
            };

            _back.Click += Back_Click;

            ControlManager.Add(_pointsLabel);
            ControlManager.Add(_remainingLabel);
            ControlManager.Add(_nameTextBox);
            ControlManager.Add(_genderSelector);
            ControlManager.Add(_portraitSelector);
            ControlManager.Add(_create);
            ControlManager.Add(_back);
            ControlManager.Add(_strengthLabel);
            ControlManager.Add(_strengthSelector);
            ControlManager.Add(_perceptionLabel);
            ControlManager.Add(_perceptionSelector);
            ControlManager.Add(_enduranceLabel);
            ControlManager.Add(_enduranceSelector);
            ControlManager.Add(_charismaLabel);
            ControlManager.Add(_charismaSelector);
            ControlManager.Add(_intellectLabel);
            ControlManager.Add(_intellectSelector);
            ControlManager.Add(_agilityLabel);
            ControlManager.Add(_agilitySelector);
            ControlManager.Add(_luckLabel);
            ControlManager.Add(_luckSelector);
        }
```

Now, in the click event handler of the Create button, we want to create a Player object. First, we need to make the set accessor of the GamePlayState accessible. You could go internal. However, I went public. Replace the Player accessor of the GamePlayState with the following.

```
        public Player Player { get; set; }
```

Now we can turn our attention to the Create button. There are a few things that we need. First, we need animations in order to create the sprite. Next, we need the values from the controls, if the player has spent all their points. Replace the Create_Click method of the NewGameState with this new version.

```
private void Create_Click(object sender, EventArgs e)
{
    if (_points > 0) return;

    Dictionary<string, Animation> animations = new();

    Animation animation = new(3, 32, 32, 0, 0) { CurrentFrame = 0, FramesPerSecond = 8 };
    animations.Add("walkdown", animation);

    animation = new(3, 32, 32, 0, 32) { CurrentFrame = 0, FramesPerSecond = 8 };
    animations.Add("walkleft", animation);

    animation = new(3, 32, 32, 0, 64) { CurrentFrame = 0, FramesPerSecond = 8 };
    animations.Add("walkright", animation);
```

```
    animation = new(3, 32, 32, 0, 96) { CurrentFrame = 0, FramesPerSecond = 8 };
    animations.Add("walkup", animation);

    Player player = new(
        Game,
        new(_genderSelector.SelectedIndex == 0 ?
            _femalePortraits[_portraitSelector.SelectedItem] :
            _malePortraits[_portraitSelector.SelectedItem],
        animations))
    {
        Strength = 2 + _strengthSelector.SelectedIndex,
        Perception = 2 + _perceptionSelector.SelectedIndex,
        Endurance = 2 + _enduranceSelector.SelectedIndex,
        Charisma = 2 + _charismaSelector.SelectedIndex,
        Intellect = 2 + _intellectSelector.SelectedIndex,
        Agility = 2 + _agilitySelector.SelectedIndex,
        Luck = 2 + _luckSelector.SelectedIndex
    };

    player.Health = new(player.Strength * player.Endurance);
    player.Mana = new(player.Intellect * player.Endurance);

    player.Sprite.CurrentAnimation = "walkdown";

    IGamePlayState gamePlayState = Game.Services.GetService<IGamePlayState>();

    gamePlayState.Player = player;

    StateManager.PopState();
    StateManager.PushState(gamePlayState.Tag);

    gamePlayState.Tag.Enabled = true;
}
```

What has changed? Well, the first thing is there is a Dictionary<string, Animation> to hold the animations of the sprite. I create the animations the same way that I did in the GamePlayState. After creating the animations, I create the player. To get the texture for the player's sprite, I use the ternary operator on the selected index of the _genderSelector. If it is zero, I use the female portrait as the first option is female. If it is not, I use the male portraits. I also set the value of the attributes. They are two plus the selected index of the selector. After creating the player object, I set the health and mana attributes. As you may recall, health is strength times endurance, and mana is intellect by endurance. I set the initial animation of the sprite to walk down. I did not set it to IsAnimating to be true. You could do that if you had an idle animation. These sprites do not. I retrieve the instance of the GamePlayState and set the Player property to the player. The rest of the method flows as before.

That leaves a few changes to the GamePlayState. We no longer want to create a sprite for the player in the LoadContent method. Though, we still want to create Rio, so she can dance across the sand. The other thing we want to do is call the Update method of the player and the Draw method of the player. I also updated the code for moving the sprite, as it no longer uses the local sprite that we are using. So, while the changes were superficial, there were enough of them to be life-threatening. Therefore, I am going to give you the code for the entire class. I'm not going to go over it any further than I already have. I feel that the changes are straightforward and it is easy to see what is going on.

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

```
using Microsoft.Xna.Framework.Graphics;
using Rpglibrary.TileEngine;
using RpgLibrary.Characters;
using RpgLibrary.TileEngine;
using SharedProject.Characters;
using SharedProject.Controls;
using SharedProject.GamesScreens;
using SharedProject.Sprites;
using SharedProject.StateManagement;
using System;
using System.Collections.Generic;
using System.Linq;

namespace SharedProject.GameScreens
{
    public class GamePlayState : GameState, IGamePlayState
    {
        readonly Camera camera;
        TileMap map;
        readonly Engine engine;
        RenderTarget2D renderTarget;
        private Button upButton, downButton, leftButton, rightButton;
        private bool inMotion = false;
        private Rectangle collision = new();
        private float speed;
        private Vector2 motion;

        public Player Player { get; set; }

        private IConversationManager _conversationManager;
        private IConversationState _conversationState;

        public GamePlayState(Game game) : base(game)
        {
            Game.Services.AddService<IGamePlayState>(this);
            camera = new(Settings.BaseRectangle);
            engine = new(32, 32, Settings.BaseRectangle);
        }

        public GameState Tag => this;

        public override void Initialize()
        {
            speed = 96;

            base.Initialize();
        }

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

            _conversationState = Game.Services.GetService<IConversationState>();
            _conversationManager = Game.Services.GetService<IConversationManager>();

            renderTarget = new(GraphicsDevice, Settings.BaseWidth, Settings.BaseHeight);

            Texture2D texture = Game.Content.Load<Texture2D>(@"Tiles/tileset1");

            List<Tileset> tilesets = new()
```

```
            {
                new(texture, 8, 8, 32, 32),
            };

            TileLayer layer = new(100, 100);

            map = new("test", tilesets[0], layer);

            Dictionary<string, Animation> animations = new();

            Animation animation = new(3, 32, 32, 0, 0) { CurrentFrame = 0, FramesPerSec-
ond = 8 };
            animations.Add("walkdown", animation);

            animation = new(3, 32, 32, 0, 32) { CurrentFrame = 0, FramesPerSecond = 8 };
            animations.Add("walkleft", animation);

            animation = new(3, 32, 32, 0, 64) { CurrentFrame = 0, FramesPerSecond = 8 };
            animations.Add("walkright", animation);

            animation = new(3, 32, 32, 0, 96) { CurrentFrame = 0, FramesPerSecond = 8 };
            animations.Add("walkup", animation);

            texture = Game.Content.Load<Texture2D>(@"PlayerSprites/femalefighter");

            AnimatedSprite rio = new(texture, animations)
            {
                CurrentAnimation = "walkdown",
                IsAnimating = true,
            };

            CharacterLayer chars = new();

            chars.Characters.Add(
                new Villager(rio, new(10, 10))
                {
                    Position = new(320, 320),
                    Tile = new(10, 10),
                    Visible= true,
                    Enabled=true,
                    Conversation="Rio"
                });

            map.AddLayer(chars);

            rightButton = new(Game.Content.Load<Texture2D>("GUI/g21245"), But-
tonRole.Menu)
            {
                Position = new(80, Settings.BaseHeight - 80),
                Size = new(32, 32),
                Text = "",
                Color = Color.White,
            };

            rightButton.Down += RightButton_Down;
            ControlManager.Add(rightButton);

            upButton = new(Game.Content.Load<Texture2D>("GUI/g21263"), ButtonRole.Menu)
            {
                Position = new(48, Settings.BaseHeight - 48 - 64),
```

```
            Size = new(32, 32),
            Text = "",
            Color = Color.White,
        };

        upButton.Down += UpButton_Down;
        ControlManager.Add(upButton);

        downButton = new(Game.Content.Load<Texture2D>("GUI/g21272"), ButtonRole.Menu)
        {
            Position = new(48, Settings.BaseHeight - 48),
            Size = new(32, 32),
            Text = "",
            Color = Color.White,
        };

        downButton.Down += DownButton_Down;
        ControlManager.Add(downButton);

        leftButton = new(Game.Content.Load<Texture2D>("GUI/g22987"), ButtonRole.Menu)
        {
            Position = new(16, Settings.BaseHeight - 80),
            Size = new(32, 32),
            Text = "",
            Color = Color.White,
        };

        leftButton.Down += LeftButton_Down;

        ControlManager.Add(leftButton);
    }

    private void LeftButton_Down(object sender, EventArgs e)
    {
        if (!inMotion)
        {
            MoveLeft();
        }
    }

    private void MoveLeft()
    {
        motion = new(-1, 0);
        inMotion = true;
        Player.Sprite.CurrentAnimation = "walkleft";
        collision = new(
            (Player.Sprite.Tile.X - 2) * Engine.TileWidth,
            Player.Sprite.Tile.Y * Engine.TileHeight,
            Engine.TileWidth,
            Engine.TileHeight);
    }

    private void RightButton_Down(object sender, EventArgs e)
    {
        if (!inMotion)
        {
            MoveRight();
        }
    }
```

```
private void MoveRight()
{
    motion = new(1, 0);
    inMotion = true;
    Player.Sprite.CurrentAnimation = "walkright";
    collision = new(
        (Player.Sprite.Tile.X + 2) * Engine.TileWidth,
        Player.Sprite.Tile.Y * Engine.TileHeight,
        Engine.TileWidth,
        Engine.TileHeight);
}

private void DownButton_Down(object sender, EventArgs e)
{
    if (!inMotion)
    {
        MoveDown();
    }
}
private void MoveDown()
{
    motion = new(0, 1);
    Point newTile = Player.Sprite.Tile + new Point(0, 2);
    inMotion = true;
    Player.Sprite.CurrentAnimation = "walkdown";
    collision = new(
        newTile.X * Engine.TileWidth,
        newTile.Y * Engine.TileHeight,
        Engine.TileWidth,
        Engine.TileHeight);
}

private void UpButton_Down(object sender, EventArgs e)
{
    if (!inMotion)
    {
        MoveUp();
    }
}

private void MoveUp()
{
    motion = new(0, -1);
    inMotion = true;
    Player.Sprite.CurrentAnimation = "walkup";
    collision = new(
        Player.Sprite.Tile.X * Engine.TileWidth,
        (Player.Sprite.Tile.Y - 2) * Engine.TileHeight,
        Engine.TileWidth,
        Engine.TileHeight);
}

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

    Player.Update(gameTime);
    map.Update(gameTime);

    if (Xin.IsKeyDown(Microsoft.Xna.Framework.Input.Keys.A) && !inMotion)
```

```
            {
                MoveLeft();
            }
            else if (Xin.IsKeyDown(Microsoft.Xna.Framework.Input.Keys.D) && !inMotion)
            {
                MoveRight();
            }

            if (Xin.IsKeyDown(Microsoft.Xna.Framework.Input.Keys.W) && !inMotion)
            {
                MoveUp();
            }
            else if (Xin.IsKeyDown(Microsoft.Xna.Framework.Input.Keys.S) && !inMotion)
            {
                MoveDown();
            }

            if (Xin.WasKeyReleased(Microsoft.Xna.Framework.Input.Keys.F) && !inMotion)
            {
                HandleConversation();
            }

            if (motion != Vector2.Zero)
            {
                motion.Normalize();
                Player.Sprite.IsAnimating = true;
            }
            else
            {
                inMotion = false;
                Player.Sprite.IsAnimating = false;
                return;
            }

            if (!Player.Sprite.LockToMap(new(99 * Engine.TileWidth, 99 * Engine.Tile-
Height), ref motion))
            {
                inMotion = false;
                return;
            }

            Vector2 newPosition = Player.Sprite.Position + motion * speed * (float)ga-
meTime.ElapsedGameTime.TotalSeconds;

            Rectangle nextPotition = new(
                (int)newPosition.X,
                (int)newPosition.Y,
                Engine.TileWidth,
                Engine.TileHeight);

            if (nextPotition.Intersects(collision))
            {
                inMotion = false;
                motion = Vector2.Zero;
                Player.Sprite.Position = new((int)Player.Sprite.Position.X,
(int)Player.Sprite.Position.Y);
                return;
            }

            if (map.PlayerCollides(nextPotition))
```

```
            {
                inMotion = false;
                motion = Vector2.Zero;
                return;
            }

            Player.Sprite.Position = newPosition;
            Player.Sprite.Tile = Engine.VectorToCell(newPosition);

            camera.LockToSprite(Player.Sprite, map);

            base.Update(gameTime);
        }

        private void HandleConversation()
        {
            var layer = map.Layers.FirstOrDefault(x => x is CharacterLayer);

            if (layer is CharacterLayer characterLayer)
            {
                foreach (ICharacter c in characterLayer.Characters)
                {
                    if (c.Tile.X == Player.Sprite.Tile.X && Math.Abs(Player.Sprite.Tile.Y
- c.Tile.Y) == 1 ||
                        (c.Tile.Y == Player.Sprite.Tile.Y &&
Math.Abs(Player.Sprite.Tile.X - c.Tile.X) == 1))
                    {
                        if (c is Villager villager)
                        {
                            _conversationState.SetConversation(Player, villager.Conversa-
tion);
                            StateManager.PushState((ConversationState)_conversation-
State);
                            //Conversation conversation = (Conversation)_conversationMan-
ager.GetConversation(villager.Conversation);
                            //if (conversation != null)
                            //{
                            //}
                        }
                    }
                }
            }
        }

        public override void Draw(GameTime gameTime)
        {
            GraphicsDevice.SetRenderTarget(renderTarget);
            GraphicsDevice.Clear(Color.Black);

            SpriteBatch.Begin(SpriteSortMode.Immediate,
                              BlendState.AlphaBlend,
                              SamplerState.PointClamp,
                              null,
                              null,
                              null,
                              camera.Transformation);

            map.Draw(gameTime, SpriteBatch, camera);
            Player.Draw(SpriteBatch);
```

```
            SpriteBatch.End();

            SpriteBatch.Begin();

            base.Draw(gameTime);

            SpriteBatch.End();

            GraphicsDevice.SetRenderTarget(null);

            SpriteBatch.Begin(SpriteSortMode.Immediate,
                              BlendState.AlphaBlend,
                              SamplerState.PointClamp);

            SpriteBatch.Draw(renderTarget, Settings.TargetRectangle, Color.White);

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

Before, I wrap things up. I will address a defect pointed out by one of my readers, Weak Entity. It had to do with conversations. There were two problems. The first was in the constructor of the GameScene class. The second was in the Button_Click method. Part of the problem was I did not set the OptionAction when creating a new instance of ButtonGroup. The other part of the problem is I was passing sender as a parameter when I should have been passing button as a parameter. Replace the constructor with this version and the Button_Click method with its version.

```
public GameScene(Game game, string text, List<SceneOption> options)
{
    this.game = game;
    this.text = text;

    LoadContent();

    Size = MeasureText(out this.text, text);
    Size += new Vector2(0, (options.Count + 5) * font.LineSpacing);

    this.options = new List<SceneOption>();

    int index = 0;

    foreach (var option in options)
    {
        this.options.Add(option);
        ButtonGroup buttonGroup = new()
        {
            Button = new(button, ButtonRole.Menu)
            {
                Size = new(font.LineSpacing, font.LineSpacing),
                Text = "",
                Index = index++,
            },
            Text = option.OptionText,
            Action = option.OptionAction
        };

        buttonGroup.Button.Click += Button_Click;
```

```
            buttons.Add(buttonGroup);
        }

        highLight = Color.Red;
        normal = Color.Black;
    }

    private void Button_Click(object sender, EventArgs e)
    {
        Button btn = (Button)sender;

        if (btn.Index.HasValue)
        {
            ButtonGroup button = this.buttons[btn.Index.Value];
            SelectedIndexEventArgs item = new() { Index = btn.Index.Value };
            ItemSelected?.Invoke(button, item);
        }
    }
```

If you build and run now, you can run through the process of creating a character, and your changes actually have an influence on the game, at least superficially. You can also have a conversation with Rio without the game crashing. If you're curious, Rio is the name of a song by Duran Duran. In the lyrics, she dances across the desert sand like a river.

I'm going to park this tutorial here. There is more that I could pack in, but I will save it for the following tutorials. I don't want you to have too much to digest at once. I encourage you to visit the news page of my site, https://cynthiamcmahon.ca/blog/, for the latest news on my tutorials. Also, I'm thinking of reviving my newsletter so you will be informed of new stuff rather than having to keep looking for new things.

Good luck with your game programming adventures!
Cynthia