Eyes of the Dragon Tutorials
Part 54
Character Stats

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

The main focus for this tutorial will be displaying the player's character stats. Also, it will double back on levelling up because a few topics were missed last time. So, let's get started.

I will get started with levelling up since it is the easier of the two areas I want to address. We have a LevelUp method in the Entity class that right now increases the entity's Level attribute by one. I am going to replace the LevelUp method with a new method. Replace the LevelUp method of the Entity class with the following code.

```
public void LevelUp()
{
    Level++;

    string[] parts = HealthCalculation.Split('|');

    int amount = int.Parse(parts[0]) + Mechanics.RollDie((DieType)int.Parse(parts[2]));

    Health.SetMaximum(amount + Health.MaximumValue);
    Health.Heal((ushort)amount);

    if (Mana.MaximumValue > 0)
    {
        parts = ManaCalculation.Split('|');

        amount = int.Parse(parts[0]) + Mechanics.RollDie((DieType)int.Parse(parts[2]));

        Mana.SetMaximum(amount + Mana.MaximumValue);
        Mana.Heal((ushort)amount);
    }
    else
    {
        parts = StaminaCalculation.Split('|');

        amount = int.Parse(parts[0]) + Mechanics.RollDie((DieType)int.Parse(parts[2]));

        Stamina.SetMaximum(amount + Stamina.MaximumValue);
        Stamina.Heal((ushort)amount);
    }
}
```

First, it splits the HealthCalculation attribute into its parts on the pipe character. Then it assigns a local variable to the first part, parsed as an integer, plus a die roll of the last substring. This is different than I had originally intended for this to work, but it turns out to do well. So, a fighter levelling up will get 12 health added to their maximum and then random between 1 and 12. For now, the attribute the

stat is based on is ignored. I will possibly update that in a future tutorial. What we have now does work well enough. I will leave that as an exercise for you if that interests you. The amount is then added to the maximum and current values.

Next, I check to see if the entity's mana is greater than zero, meaning they are a spellcaster. If it is, I split the ManaCalculation property into its parts. I then calculated the amount the same way I figured the health. Thus, the maximum and current mana attributes are increased. Similarly, I do the same thing for stamina.

While testing the level-up feature, I found a bug. I didn't save the formulas for calculating attributes when creating Entity objects from EntityData objects. Update the constructor of the Entity class as follows.

```
public Entity(
    string name,
    EntityData entityData,
    EntityGender gender,
    EntityType entityType) : this()
{
    EntityName = name;
    Level = entityData.Level;
    EntityClass = entityData.EntityName;
    Gender = gender;
    EntityType = entityType;
    Strength = entityData.Strength;
    Dexterity = entityData.Dexterity;
    Cunning = entityData.Cunning;
    Willpower = entityData.Willpower;
    Magic = entityData.Magic;
    Constitution = entityData.Constitution;

    HealthCalculation = entityData.HealthFormula;
    StaminaCalculation = entityData.StaminaFormula;
    ManaCalculation = entityData.MagicFormula;

    if (!string.IsNullOrEmpty(entityData.HealthFormula))
    {
        health = new AttributePair(CalculateAttribute(entityData.HealthFormula));
    }
    else
    {
        health = new AttributePair(entityData.MaximumHealth);
    }

    if (!string.IsNullOrEmpty(entityData.StaminaFormula))
    {
        stamina = new AttributePair(CalculateAttribute(entityData.StaminaFormula));
    }
    else
    {
        stamina = new AttributePair(entityData.MaximumStamina);
    }

    if (!string.IsNullOrEmpty(entityData.MagicFormula))
    {
        mana = new AttributePair(CalculateAttribute(entityData.MagicFormula));
```

```
    }
    else
    {
        mana = new AttributePair(entityData.MaximumMana);
    }
}
```

Nothing out of the ordinary here. I just assign the calculation attributes to the attributes from the EntityData class. If you don't do this, the level up will blow up with a null reference exception.

Now, turning my attention to displaying character stats. I know that it works well enough. However, it doesn't look the greatest and doesn't fit the rest of the game. To make them align, you will first need an image to display the player character's stats. You can find the texture that I used at https://cynthiamcmahon.ca/blog/downloads/character-gui.png. I took the image from the inventory state. I just wiped the area where the equipment was displayed on the right. I will use that area for displaying the character's stats. You need to download it and it to the GUI folder in the Content folder of the main project. You've done that often, and I hope by now you can do that without instructions.

Now I am going to update the state for displaying characters. It will be similar to the inventory state. Replace the StatsScreen file in the EyesOfTheDragon project with the following code.

```
using MGRpgLibrary;
using MGRpgLibrary.Controls;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using RpgLibrary.CharacterClasses;
using System;
using System.Collections.Generic;
using System.Text;

namespace EyesOfTheDragon.GameScreens
{
    public class StatsScreen : BaseGameState
    {
        protected Texture2D _background;
        protected Label _str;
        protected Label _dex;
        protected Label _cun;
        protected Label _mag;
        protected Label _wil;
        protected Label _con;
        protected Label _lvl;
        protected Label _xp;
        protected Label _next;
        protected Label _hp;
        protected Label _res;
        protected Label _resName;
        protected Rectangle _closeArea = new Rectangle(1175, 45, 45, 30);
        protected Rectangle _headGear;
        protected Rectangle _bodyGear;
        protected Rectangle _handGear;
        protected Rectangle _footGear;
        protected Rectangle _neckGear;
```

```
protected Rectangle _mainGear;
protected Rectangle _offGear;
protected Rectangle _fingerGear;

public StatsScreen(Game game, GameStateManager manager) : base(game, manager)
{
}

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

    _background = GameRef.Content.Load<Texture2D>("GUI/character-gui");

    int yOffset = 110;
    int xOffset = 630;

    Label label = new Label()
    {
        Position = new Vector2(xOffset, yOffset),
        Text = "Strength: ",
        Color = Color.White,
        TabStop = false
    };

    ControlManager.Add(label);

    _str = new Label()
    {
        Position = new Vector2(xOffset + 160, yOffset),
        Text = $"{GamePlayScreen.Player.Character.Entity.Strength}",
        Color = Color.White,
        TabStop = false
    };

    ControlManager.Add(_str);

    label = new Label()
    {
        Position = new Vector2(xOffset, 40 + yOffset),
        Text = "Dexterity: ",
        Color = Color.White,
        TabStop = false
    };

    ControlManager.Add(label);

    _dex = new Label()
    {
        Position = new Vector2(xOffset + 160, 40 + yOffset),
        Text = $"{GamePlayScreen.Player.Character.Entity.Dexterity}",
        Color = Color.White,
        TabStop = false
    };

    ControlManager.Add(_dex);

    label = new Label()
    {
        Position = new Vector2(xOffset, 80 + yOffset),
```

```
    Text = "Cunning: ",
    Color = Color.White,
    TabStop = false
};

ControlManager.Add(label);

_cun = new Label()
{
    Position = new Vector2(xOffset + 160, 80 + yOffset),
    Text = $"{GamePlayScreen.Player.Character.Entity.Cunning}",
    Color = Color.White,
    TabStop = false
};

ControlManager.Add(_cun);

label = new Label()
{
    Position = new Vector2(xOffset, 120 + yOffset),
    Text = "Magic: ",
    Color = Color.White,
    TabStop = false
};

ControlManager.Add(label);

_mag = new Label()
{
    Position = new Vector2(xOffset + 120, 120 + yOffset),
    Text = $"{GamePlayScreen.Player.Character.Entity.Magic}",
    Color = Color.White,
    TabStop = false
};

ControlManager.Add(_mag);

label = new Label()
{
    Position = new Vector2(xOffset, 160 + yOffset),
    Text = "Will Power: ",
    Color = Color.White,
    TabStop = false
};

ControlManager.Add(label);

_wil = new Label()
{
    Position = new Vector2(xOffset + 160, 160 + yOffset),
    Text = $"{GamePlayScreen.Player.Character.Entity.Willpower}",
    Color = Color.White,
    TabStop = false
};

ControlManager.Add(_wil);
label = new Label()
{
    Position = new Vector2(xOffset, 200 + yOffset),
    Text = "Constitution: ",
```

```
    Color = Color.White,
    TabStop = false
};

ControlManager.Add(label);

_con = new Label()
{
    Position = new Vector2(xOffset + 160, 200 + yOffset),
    Text = $"{GamePlayScreen.Player.Character.Entity.Constitution}",
    Color = Color.White,
    TabStop = false
};

ControlManager.Add(_con);

label = new Label()
{
    Position = new Vector2(xOffset + 250, yOffset),
    Text = "Level:",
    Color = Color.White,
    TabStop = false
};

ControlManager.Add(label);

_lvl = new Label
{
    Position = new Vector2(xOffset + 250 + 80, yOffset),
    Text = $"{GamePlayScreen.Player.Character.Entity.Level}",
    Color = Color.Yellow,
    TabStop = false
};

ControlManager.Add(_lvl);

label = new Label()
{
    Position = new Vector2(xOffset + 250, yOffset + 40),
    Text = "XP:",
    Color = Color.White,
    TabStop = false
};

ControlManager.Add(label);

_xp = new Label
{
    Position = new Vector2(xOffset + 250 + 80, yOffset + 40),
    Text = $"{GamePlayScreen.Player.Character.Entity.Experience}",
    Color = Color.Yellow,
    TabStop = false
};

ControlManager.Add(_xp);

label = new Label()
{
    Position = new Vector2(xOffset + 250, yOffset + 80),
    Text = "HP:",
```

```
                Color = Color.White,
                TabStop = false
            };

            ControlManager.Add(label);

            _hp = new Label
            {
                Position = new Vector2(xOffset + 250 + 80, yOffset + 80),
                Text = $"{GamePlayScreen.Player.Character.Entity.Health.CurrentValue} /
{GamePlayScreen.Player.Character.Entity.Health.MaximumValue}",
                Color = Color.Yellow,
                TabStop = false
            };

            ControlManager.Add(_hp);

            _resName = new Label
            {
                Position = new Vector2(xOffset + 250, yOffset + 120),
                Color = Color.White,
                TabStop = false
            };

            _resName.Text = GamePlayScreen.Player.Character.Entity.Mana.MaximumValue > 0 ?
"MP:" : "SP:";

            ControlManager.Add(_resName);

            _res = new Label
            {
                Position = new Vector2(xOffset + 250 + 80, yOffset + 120),
                Color = Color.Yellow,
                TabStop = false
            };

            _res.Color = GamePlayScreen.Player.Character.Entity.Mana.MaximumValue > 0 ?
Color.Yellow : Color.Blue;

            _res.Text = GamePlayScreen.Player.Character.Entity.Mana.MaximumValue > 0 ?
                $"{GamePlayScreen.Player.Character.Entity.Mana.CurrentValue} / " +
                $"{GamePlayScreen.Player.Character.Entity.Mana.MaximumValue}" :
                $"{GamePlayScreen.Player.Character.Entity.Stamina.CurrentValue} / " +
                $"{GamePlayScreen.Player.Character.Entity.Stamina.MaximumValue}";

            _headGear = new Rectangle(55, 157, 101, 73);
            _bodyGear = new Rectangle(55, 247, 101, 73);
            _handGear = new Rectangle(55, 337, 101, 73);
            _footGear = new Rectangle(55, 427, 101, 73);
            _neckGear = new Rectangle(500, 157, 101, 73);
            _mainGear = new Rectangle(500, 247, 101, 73);
            _offGear = new Rectangle(500, 337, 101, 73);
            _fingerGear = new Rectangle(500, 427, 101, 73);
        }

        private void Close_Selected(object sender, EventArgs e)
        {
            OnCloseSelected();
        }
```

```
        protected virtual void OnCloseSelected()
        {
            StateManager.PopState();
        }
        public override void Update(GameTime gameTime)
        {
            ControlManager.Update(gameTime, PlayerIndex.One);

            if (InputHandler.KeyReleased(Keys.Escape))
            {
                StateManager.PopState();
            }

            if (InputHandler.CheckMouseReleased(MouseButton.Left))
            {
                if (_closeArea.Contains(InputHandler.MouseAsPoint))
                {
                    StateManager.PopState();
                }
            }
            base.Update(gameTime);
        }

        public override void Draw(GameTime gameTime)
        {
            base.Draw(gameTime);
            _str.Text = $"{GamePlayScreen.Player.Character.Entity.Strength}";
            _dex.Text = $"{GamePlayScreen.Player.Character.Entity.Dexterity}";
            _cun.Text = $"{GamePlayScreen.Player.Character.Entity.Cunning}";
            _mag.Text = $"{GamePlayScreen.Player.Character.Entity.Magic}";
            _wil.Text = $"{GamePlayScreen.Player.Character.Entity.Willpower}";
            _con.Text = $"{GamePlayScreen.Player.Character.Entity.Constitution}";
            _lvl.Text = $"{GamePlayScreen.Player.Character.Entity.Level}";
            _xp.Text = $"{GamePlayScreen.Player.Character.Entity.Experience}";
            _hp.Text = $"{GamePlayScreen.Player.Character.Entity.Health.CurrentValue} /
{GamePlayScreen.Player.Character.Entity.Health.MaximumValue}";
            _resName.Text = GamePlayScreen.Player.Character.Entity.Mana.MaximumValue > 0 ?
"MP:" : "SP:";
            _res.Color = GamePlayScreen.Player.Character.Entity.Mana.MaximumValue > 0 ?
Color.Yellow : Color.Blue;
            _res.Text = GamePlayScreen.Player.Character.Entity.Mana.MaximumValue > 0 ?
                $"{GamePlayScreen.Player.Character.Entity.Mana.CurrentValue} / " +
                $"{GamePlayScreen.Player.Character.Entity.Mana.MaximumValue}" :
                $"{GamePlayScreen.Player.Character.Entity.Stamina.CurrentValue} / " +
                $"{GamePlayScreen.Player.Character.Entity.Stamina.MaximumValue}";

            GameRef.SpriteBatch.Begin();

            GameRef.SpriteBatch.Draw(_background, new Rectangle(0, 0, 1280, 720), Color.White);

            Entity e = GamePlayScreen.Player.Character.Entity;

            if (e.MainHand != null)
            {
                e.MainHand.Draw(GameRef.SpriteBatch, _mainGear);
            }

            if (e.OffHand != null)
            {
                e.OffHand.Draw(GameRef.SpriteBatch, _offGear);
```

```
            }

            if (e.Head != null)
            {
                e.Head.Draw(GameRef.SpriteBatch, _headGear);
            }

            if (e.Body != null)
            {
                e.Body.Draw(GameRef.SpriteBatch, _bodyGear);
            }

            if (e.Hands != null)
            {
                e.Hands.Draw(GameRef.SpriteBatch, _handGear);
            }

            if (e.Feet != null)
            {
                e.Feet.Draw(GameRef.SpriteBatch, _footGear);
            }

            ControlManager.Draw(GameRef.SpriteBatch);

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

Much of this stayed the same. The first difference is I added the destination rectangles from the inventory stat for the player's equipment. In the LoadContent method, I load the image that I linked to earlier instead of creating a texture through code. Instead of centering the texture, I have it fill the screen. I position the labels based on the image. The position of the inventory items remains the same because that part of the texture did not change. Also, I removed the close link label. I will handle closing by detecting if the Escape can has been released or if the player has clicked over the X from the image.

In the Update method, I now check to see if the Escape key has been released. If it has, the state is popped off the stack of states. Similarly, if the left mouse button has been released and the mouse is over the X, I pop the current state off the stack.

The Draw method is pretty close to the original version. The difference is now I draw any equipped items in their position.

So, that is going to be it for this tutorial. I know that it is short and most of it you have seen before. Regardless, I accomplished what I intended, and I don't want to venture further into this tutorial. So, please continue to visit my blog, https://cynthiamcmahon.ca/blog/, for the latest news on my tutorials and other goodness.

Good luck with your Game Programming Adventures!

Cynthia