Eyes of the Dragon Tutorials
Part 52
Gaining Gold and Experience

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 point of this tutorial is gaining gold and experience so the character can level up and buy more powerful items. Also, I will be adding in healing the character and more mobs for the player to fight, though they will still just be bandits because I don't want to search for sprites. So, let's get started.

The first step will be to update the Mechanics class to expose the Random object. Change the Property region of that class to the following.

```
#region Property Region

public static Random Random => random;

#endregion
```

Nothing exciting here, there is just a property that returns the random field. The next step will be to update the Mob class. I added a using statement for the RpgLibrary to bring the Mechanics class into scope. I also added two field _minGold and _maxGold which are the minimum amount of gold dropped and the maximum amount of gold dropped respectively. There is also a property that returns the gold that was dropped. I use the overload of the Next method of the Random class that takes a minimum and maximum value, minus one. Add this using statement to the Mob class and add the following region.

```
using RpgLibrary;

#region Money Drop Region

protected int _minGold;
protected int _maxGold;

public int GoldDrop
{
    get { return Mechanics.Random.Next(_minGold, _maxGold); }
}

#endregion
```

I changed the Bandit class to drop gold. Since it is a short class, I will give you the code for the entire class.

```
using MGRpgLibrary.SpriteClasses;
using RpgLibrary;
using RpgLibrary.CharacterClasses;
using System;
using System.Collections.Generic;
using System.Text;
```

```
namespace MGRpgLibrary.Mobs
{
    public class Bandit : Mob
    {
        public Bandit(Entity entity, AnimatedSprite sprite)
            : base(entity, sprite)
        {
            _minGold = 20;
            _maxGold = 50;
        }

        public override void Attack(Entity source)
        {
            if (Mechanics.RollDie(DieType.D20) >= 10 + Mechanics.GetModifier(entity.Dexterity))
            {
                entity.ApplyDamage(source.MainHand);
            }
        }

        public override void DoAttack(Entity target)
        {
            if (Mechanics.RollDie(DieType.D20) >= 10 + Mechanics.GetModifier(target.Dexterity))
            {
                target.ApplyDamage(entity.MainHand);
            }
        }
    }
}
```

The next thing I'm going to tackle is having multiple mobs on the map. I can't just add the same one over and over because they are reference types. When you defeat one of them, you would defeat them all. What I did was wrap the creation in a loop and add news instances. Change the LoadWorld method of the CharacterGeneratorScreen to the following. Also, add this using statement to bring the RpgLibrary classes into scope.

```
using RpgLibrary;

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

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

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

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

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

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

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

                character = new NonPlayerCharacter(entity, sprite);

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

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

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

    Level level = new Level(map);

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

    Chest chest = new Chest(chestData);

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

    ItemSprite itemSprite = new ItemSprite(
        chest,
        chestSprite);

    level.Chests.Add(itemSprite);

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

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

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

    EntityData ed = new EntityData("Eliza", 1, 10, 10, 10, 10, 10, 10, "20|CON|12", "16|WIL|
16",
        "0|0|0");
    Entity e = new Entity("Eliza", ed, EntityGender.Female, EntityType.NPC);

    NonPlayerCharacter npc = new NonPlayerCharacter(e, s);

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

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

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

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

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

    for (int i = 0; i < 25; i++)
    {
        ed = new EntityData("Bandit", 1, 10, 12, 12, 10, 10, 10, "20|CON|10", "12|WIL|12", "0|
0|0");

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

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

        Mob mob = new Bandit(e, s);

        Rectangle r = new Rectangle(Mechanics.Random.Next(10, 20) * 32,
Mechanics.Random.Next(10, 20) * 32, 32, 32);

        if (!mobLayer.Mobs.ContainsKey(r))
        {
            mobLayer.Mobs.Add(r, mob);
        }

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

As I just mentioned, I wrapped the code for creating a bandit in a for loop that loops 25 times. After creating the bandit, I create a rectangle that is between tiles 10 and 20 for the X and Y coordinates. If the dictionary of mobs does not contain a rectangle the I add the mob at that tile to the dictionary.

If you run and build now you will find up to 25 bandits clumped on the map. I say up to 25 because it is possible for the loop to place a bandit on an existing tile.

The next thing to do is add the gold to the character when a mob dies. What I did is update the CombatScreen to push the mob's gold to the LootScreen. Then on close, add the gold to the character's gold. First, update the LootScreen.

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

namespace EyesOfTheDragon.GameScreens
{
    public class LootScreen : BaseGameState
    {
        public readonly List<GameItem> Items;
        private bool _over;
        private int _index;
        public int Gold { get; set; }

        public LootScreen(Game game, GameStateManager manager) : base(game, manager)
        {
            Items = new List<GameItem>();
        }

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

            LinkLabel linkLabel = new LinkLabel()
            {
                Text = "Close",
                TabStop = true,
                Color = Color.White,
                SelectedColor = Color.Red,
                Position = new Vector2(
                    (Game1.ScreenWidth - 
FontManager.GetFont("testfont").MeasureString("Close").X) / 2,
                    Game1.ScreenHeight - FontManager.GetFont("testfont").LineSpacing * 2)
            };

            linkLabel.Selected += LinkLabel_Selected;
            ControlManager.Add(linkLabel);
        }

        private void LinkLabel_Selected(object sender, EventArgs e)
        {
            GamePlayScreen.Player.Gold += Gold;
            StateManager.PopState();
        }

        public override void Update(GameTime gameTime)
        {
            if (_over)
            {
                if (InputHandler.CheckMouseReleased(MouseButton.Left) && _index > -1 && _index
< Items.Count)
                {
                    GamePlayScreen.Player.Backpack.Items.Add(Items[_index]);
                    Items.RemoveAt(_index);
                }
            }

            if (InputHandler.KeyReleased(Keys.Escape))
            {
                GamePlayScreen.Player.Gold += Gold;
                StateManager.PopState();
            }

            ControlManager.Update(gameTime, PlayerIndex.One);
        }

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

            GameRef.SpriteBatch.Begin();


            GameRef.SpriteBatch.DrawString(FontManager.GetFont("testfont"), $"Found {Gold}
gold!", Vector2.One * 10, Color.Yellow);

            ControlManager.Draw(GameRef.SpriteBatch);

            Rectangle destination = new Rectangle(50, 50, 32, 32);
            Point mouse = InputHandler.MouseAsPoint;

            int i = 0;

            _over = false;
            _index = -1;

            foreach (GameItem item in Items)
            {
                item.Draw(GameRef.SpriteBatch, destination);

                if (destination.Contains(mouse))
                {
                    _over = true;
                    _index = i;
                }

                destination.X += 50;

                if (destination.X > GameRef.ScreenRectangle.Width - 50 - 32)
                {
                    destination.X = 50;
                    destination.Y += 50;
                }

                i++;
            }

            GameRef.SpriteBatch.End();
        }

        protected override void Show()
        {
            Items.Clear();

            base.Show();
        }
    }
}
```

What I did was a a property, Gold, to the screen that will hold the gold found by the character. Then, whenever I pop the state off the stack I add the Gold property to the character's Gold property. Also, in the Draw method I draw the gold that was found.

Next step, is to update the CombatScreen to push the gold to the LootState. That needs to be done in two places, when we push the state on the stack of states. Modify the DoAction and Update methods as follows.

```
public override void Update(GameTime gameTime)
{
_queueTimer += gameTime.ElapsedGameTime.TotalSeconds;

if (_queueTimer > 2)
{
    if (GameState._descriptions.Count > 0)
    {
        GameState._descriptions.Dequeue();
    }

    _queueTimer = 0;
}

if (!_displayActionTexture)
{
    _scene.Update(gameTime, PlayerIndex.One);
}

if (GamePlayScreen.Player.Character.Entity.Health.CurrentValue <= 0)
{
    StateManager.ChangeState(GameRef.GameOverScreen);
}

if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Space) && !
_displayActionTexture)
{
    switch (_scene.SelectedIndex)
    {
        case 0:
            _queueTimer = 0;
            if (GamePlayScreen.Player.Character.Entity.Dexterity >= _mob.Entity.Dexterity)
            {
                _descriptions.Enqueue($"You attack the {_mob.Entity.EntityClass}.");
                _mob.Attack(GamePlayScreen.Player.Character.Entity);
```

```
                    if (_mob.Entity.Health.CurrentValue <= 0)
                    {
                        StateManager.PopState();
                        StateManager.PushState(GameRef.LootScreen);
                        GameRef.LootScreen.Gold = _mob.GoldDrop;

                        foreach (var i in _mob.Drops)
                        {
                            GameRef.LootScreen.Items.Add(i);
                        }

                        return;
                    }

                    _descriptions.Enqueue($"The {_mob.Entity.EntityClass} attacks you.");
                    _mob.DoAttack(GamePlayScreen.Player.Character.Entity);
                }
                else
                {
                    _descriptions.Enqueue($"The {_mob.Entity.EntityClass} attacks you.");
                    _mob.DoAttack(GamePlayScreen.Player.Character.Entity);
                    _descriptions.Enqueue($"You attack the {_mob.Entity.EntityClass}.");
                    _mob.Attack(GamePlayScreen.Player.Character.Entity);

                    if (_mob.Entity.Health.CurrentValue <= 0)
                    {
                        StateManager.PopState();
                        StateManager.PushState(GameRef.LootScreen);

                        foreach (var i in _mob.Drops)
                        {
                            GameRef.LootScreen.Items.Add(i);
                        }

                        return;
                    }
                }
                break;
            case 1:
                _displayActionTexture = true;

                _actions.Clear();
                Entity entity = GamePlayScreen.Player.Character.Entity;

                if (entity.Mana.MaximumValue > 0)
                {
                    foreach (SpellData s in DataManager.SpellData.SpellData.Values)
                    {
                        foreach (string c in s.AllowedClasses)
                        {
                            if (c == entity.EntityClass && entity.Level >= s.LevelRequirement)
                            {
                                _actions.Add(s.Name);
                            }
                        }
                    }
                }
                else
                {
```

```
                foreach (TalentData s in DataManager.TalentData.TalentData.Values)
                {
                    foreach (string c in s.AllowedClasses)
                    {
                        if (c == entity.EntityClass && entity.Level >= s.LevelRequirement)
                        {
                            _actions.Add(s.Name);
                        }
                    }
                }
            }
            return;
        case 2:
            Vector2 center = GamePlayScreen.Player.Sprite.Center;
            Vector2 mobCenter = _mob.Sprite.Center;
            _action = 0;

            if (center.Y < mobCenter.Y)
            {
                GamePlayScreen.Player.Sprite.Position.Y -= 8;
            }
            else if (center.Y > mobCenter.Y)
            {
                GamePlayScreen.Player.Sprite.Position.Y += 8;
            }
            else if (center.X < mobCenter.X)
            {
                GamePlayScreen.Player.Sprite.Position.X -= 8;
            }
            else
            {
                GamePlayScreen.Player.Sprite.Position.X += 8;
            }

            StateManager.PopState();
            break;
    }
}

if (_displayActionTexture)
{
    if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Escape))
    {
        _displayActionTexture = false;
        return;
    }
    else if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Space))
    {
        DoAction();
    }
    else if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Down))
    {
        _action++;

        if (_action >= _actions.Count)
        {
            _action = 0;
        }
    }
    else if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Up))
```

```
        {
            _action--;

            if (_action < 0)
            {
                _action = _actions.Count - 1;
            }
        }
    }

    private void DoAction()
    {
        Entity entity = GamePlayScreen.Player.Character.Entity;

        if (entity.Mana.MaximumValue > 0)
        {
            SpellData spell = DataManager.SpellData.SpellData[_actions[_action]];

            if (spell.ActivationCost > entity.Mana.CurrentValue)
            {
                return;
            }

            entity.Mana.Damage((ushort)spell.ActivationCost);

            Spell s = Spell.FromSpellData(spell);

            if (s.SpellType == SpellType.Activated)
            {
                foreach (BaseEffect e in s.Effects)
                {
                    switch (e.TargetType)
                    {
                        case TargetType.Enemy:
                            e.Apply(_mob.Entity);
                            break;
                        case TargetType.Self:
                            e.Apply(entity);
                            break;
                    }
                }
            }
        }
        else
        {
            TalentData talent = DataManager.TalentData.TalentData[_actions[_action]];

            if (talent.ActivationCost > entity.Stamina.CurrentValue)
            {
                return;
            }

            entity.Stamina.Damage((ushort)talent.ActivationCost);

            Talent s = Talent.FromTalentData(talent);

            if (s.TalentType == TalentType.Activated)
            {
                foreach (BaseEffect e in s.Effects)
                {
```

```
                switch (e.TargetType)
                {
                    case TargetType.Enemy:
                        e.Apply(_mob.Entity);
                        break;
                    case TargetType.Self:
                        e.Apply(entity);
                        break;
                }
            }
        }
    }

    _displayActionTexture = false;

    if (_mob.Entity.Health.CurrentValue > 0)
    {
        _mob.DoAttack(entity);
    }

    if (_mob.Entity.Health.CurrentValue <= 0)
    {
        GameRef.LootScreen.Gold = _mob.GoldDrop;
        StateManager.PopState();
        StateManager.PushState(GameRef.LootScreen);

        foreach (var i in _mob.Drops)
        {
            GameRef.LootScreen.Items.Add(i);
        }
    }
}
```

Now, I'm going to add the classes for potions. First, I added a data class that would be used in the editors. Right click the ItemClasses folder in the MGRpgLibrary project, select Add and then class. Name this new class PotionData. Here is the code.

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

namespace MGRpgLibrary.ItemClasses
{
    public class PotionData
    {
        public string Name;
        public string Type;
        public string Target;
        public int Price;
        public float Weight;
        public int Minimum;
        public int Maximum;
        public string[] AllowableClasses;

        public override string ToString()
        {
            string s = Name + ", ";
            s += Type + ", ";
            s += Target + ", ";
            s += Price + ", ";
            s += Weight + " , ";
            s += Minimum + ", ";
            s += Maximum;

            return s;
        }
    }
}
```

This class has the standard fields for an item: Name, Type, Price, Weight and AllowableClasses. In addition, it has fields a Target that is what the potion affects, Minimum that is the minimum amount the potion affects, and Maximum that is the maximum amount the potion affects. There is an override of the ToString method to write the data out as a string.

Now, the class that is used in the game. Right click the ItemClasses folder, select Add and then Class. Name this new class Potion. Here is the code.

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

namespace MGRpgLibrary.ItemClasses
{
    public class Potion : BaseItem
    {
        public string Target { get; private set; }
        public int Minimum { get; private set; }
        public int Maximum { get; private set; }

        public Potion(
            string name,
            string type,
            int price,
            float weight,
            string target,
            int minimum,
            int maximum,
            params string[] allowableClasses)
            : base(name, type, price, weight, allowableClasses)
        {
            Target = target;
            Minimum = minimum;
            Maximum = maximum;
        }


        public Potion(PotionData potion)
            : base(potion.Name, potion.Type, potion.Price, potion.Weight,
potion.AllowableClasses)
        {
            Target = potion.Target;
            Minimum = potion.Minimum;
            Maximum = potion.Maximum;
        }
        public void Apply(Entity entity)
        {
            switch (Target)
            {
                case "Health":
                    entity.Health.Heal((ushort)Mechanics.Random.Next(Minimum, Maximum));
                    break;
                case "Stamina":
                    entity.Stamina.Heal((ushort)Mechanics.Random.Next(Minimum, Maximum));
                    break;
                case "Mana":
                    entity.Mana.Heal((ushort)Mechanics.Random.Next(Minimum, Maximum));
                    break;
            }
        }

        public override object Clone()
        {
            Potion p = new Potion(Name, Type, Price, Weight, Target, Minimum, Maximum,
AllowableClasses.ToArray());

            return p;
        }
    }
}
```

There are using statements RpgLibrary classes, RpgLibrary.CharacterClasses classes and RpgLibrary.ItemClasses classes into scope. The class inherits from the BaseItem class. I added fields for the target, minimum value and maximum value. Because this class inherits from BaseItem, I had to include a constructors that calls the base class. I added one that takes as parameters the name, type, price, weight, target, minimum, maximum and allowed classes. It sets the fields for this new class. The second constructor takes as parameters a PotionData object. It calls the base constructor with the appropriate fields and sets the fields for this class.

I added a method, Apply, that takes an Entity object as an argument. Inside of that there is a switch on the Target property of the class. If it is Health, I call the Heal method on the entity's Health property and return a random number between the minimum and maximum. If it is Stamina or Mana I do something similar on the appropriate property.

Finally, I included the required Clone method. It just calls in the constructor passing in the fields and returns the object.

The next steps will be to update the ItemDataManager and ItemManager to include potions. For the ItemDataManager I just added a field, potionData, and property to expose it, PotionData. Update the ItemDataManager to the following code.

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

namespace RpgLibrary.ItemClasses
{
    public class ItemDataManager
    {
        #region Field Region
```

```
readonly Dictionary<string, ArmorData> armorData = new
    Dictionary<string, ArmorData>();

readonly Dictionary<string, ShieldData> shieldData = new
    Dictionary<string, ShieldData>();

readonly Dictionary<string, WeaponData> weaponData = new
    Dictionary<string, WeaponData>();

readonly Dictionary<string, ReagentData> reagentData = new
    Dictionary<string, ReagentData>();

readonly Dictionary<string, KeyData> keyData = new
    Dictionary<string, KeyData>();

readonly Dictionary<string, ChestData> chestData = new
    Dictionary<string, ChestData>();

readonly Dictionary<string, PotionData> potionData = new
    Dictionary<string, PotionData>();

#endregion

#region Property Region

public Dictionary<string, ArmorData> ArmorData
{
    get { return armorData; }
}

public Dictionary<string, ShieldData> ShieldData
{
    get { return shieldData; }
}

public Dictionary<string, WeaponData> WeaponData
{
    get { return weaponData; }
}

public Dictionary<string, ReagentData> ReagentData
{
    get { return reagentData; }
}

public Dictionary<string, KeyData> KeyData
{
    get { return keyData; }
}

public Dictionary<string, ChestData> ChestData
{
    get { return chestData; }
}

public Dictionary<string, PotionData> PotionData
{
    get { return potionData; }
}
```

```
        #endregion

        #region Constructor Region
        #endregion

        #region Method Region
        #endregion
    }
}
```

For the ItemMannager I also included a field for potions. I then added a property that exposes the key collection for the potions field. Finally, I added AddPotion, GetPotion and ContainsPotion methods to add a potion, get a potion and see if a potions exists. Since the methods are the same as for armor, shields and weapons I won't break it down line by line. Here is the new code for the ItemManager class.

```
#region Potion Methods

public static void AddPotion(Potion Potion)
{
    if (!potions.ContainsKey(Potion.Name))
    {
        potions.Add(Potion.Name, Potion);
    }
}

public static Potion GetPotion(string name)
{
    if (potions.ContainsKey(name))
    {
        return (Potion)potions[name].Clone();
    }

    return null;
}

public static bool ContainsPotion(string name)
{
    return potions.ContainsKey(name);
}

#endregion
```

Bouncing back to mobs, I am going to add another field and property. They are for the experience point value the character gains when they defeat the mob. Replace the field and property regions with the following.

```
#region Field Region

protected Entity entity;
protected AnimatedSprite sprite;
protected int _xpValue;

#endregion

#region Property Region
```

```
public Entity Entity
{
    get { return entity; }
}

public AnimatedSprite Sprite
{
    get { return sprite; }
}

public int XPValue
{
    get { return _xpValue; }
}

#endregion
```

Pretty straightforward. Just a simple integer field and a read only accessor. Now, in the constructor for the Bandit class we need to initialize the field. Update the constructor for the Bandit class to the following.

```
public Bandit(Entity entity, AnimatedSprite sprite)
    : base(entity, sprite)
{
    _minGold = 20;
    _maxGold = 50;
    _xpValue = 100;
}
```

I just added an assignment to the _xpValue field, setting it to 100. This is a choice thinking that to get to the second level the player needs to accumulate 1000 experience, which would be 10 bandits. It sounds like a reasonable number. I haven't come up with the experience forumlae, yet. There are several approaches I can take, though. It will likely be some sort of power series, like x squared or cubed.

Sorry, for bouncing around a bit. I want to turn my attention to the Game1 class. I want to update the LoadContent method to create some potions. Change the LoadContent method to the following.

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

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

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

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

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

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

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

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

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

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

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

PotionData potionData = new PotionData
{
    Name = "Minor Healing Potion",
    Type = "Potion",
    Target = "Health",
    Price = 100,
    Weight = .25f,
    Minimum = 6,
    Maximum = 13,
    AllowableClasses = new[] { "Fighter", "Wizard", "Rogue", "Priest" }
};

Potion potion = new Potion(potionData);

ItemManager.AddPotion(potion);

potionData = new PotionData
{
    Name = "Minor Mana Potion",
    Type = "Potion",
    Target = "Mana",
    Price = 100,
    Weight = .25f,
    Minimum = 6,
    Maximum = 13,
    AllowableClasses = new[] { "Wizard", "Priest" }
};
```

```
    potion = new Potion(potionData);

    ItemManager.AddPotion(potion);

    potionData = new PotionData
    {
        Name = "Minor Stamina Potion",
        Type = "Potion",
        Target = "Stamina",
        Price = 100,
        Weight = .25f,
        Minimum = 6,
        Maximum = 13,
        AllowableClasses = new[] { "Fighter", "Rogue" }
    };

    potion = new Potion(potionData);

    ItemManager.AddPotion(potion);

    GameItemManager.AddItem("Long Sword", new GameItem(ItemManager.GetWeapon("Long Sword"), 
"FullSheet", new Rectangle(1696, 1408, 32, 32)));
    GameItemManager.AddItem("Short Sword", new GameItem(ItemManager.GetWeapon("Short Sword"), 
"FullSheet", new Rectangle(800, 1504, 32, 32)));
    GameItemManager.AddItem("Apprentice Staff", new GameItem(ItemManager.GetWeapon("Apprentice 
Staff"), "FullSheet", new Rectangle(224, 1408, 32, 32)));
    GameItemManager.AddItem("Acolyte Staff", new GameItem(ItemManager.GetWeapon("Acolyte 
Staff"), "FullSheet", new Rectangle(256, 1408, 32, 32)));
    GameItemManager.AddItem("Leather Armor", new GameItem(ItemManager.GetArmor("Leather 
Armor"), "FullSheet", new Rectangle(1248, 1216, 32, 32)));
    GameItemManager.AddItem("Chain Mail", new GameItem(ItemManager.GetArmor("Chain Mail"), 
"FullSheet", new Rectangle(1472, 1184, 32, 32)));
    GameItemManager.AddItem("Studded Leather Armor", new GameItem(ItemManager.GetArmor("Studded 
Leather Armor"), "FullSheet", new Rectangle(1984, 1120, 32, 32)));
    GameItemManager.AddItem("Light Robes", new GameItem(ItemManager.GetArmor("Light Robes"), 
"FullSheet", new Rectangle(992, 1216, 32, 32)));
    GameItemManager.AddItem("Medium Robes", new GameItem(ItemManager.GetArmor("Medium Robes"), 
"FullSheet", new Rectangle(1024, 1216, 32, 32)));
    GameItemManager.AddItem("Minor Healing Potion", new GameItem(ItemManager.GetPotion("Minor 
Healing Potion"), "FullSheet", new Rectangle(832, 1344, 32, 32)));
    GameItemManager.AddItem("Minor Mana Potion", new GameItem(ItemManager.GetPotion("Minor Mana 
Potion"), "FullSheet", new Rectangle(576, 1344, 32, 32)));
    GameItemManager.AddItem("Minor Stamina Potion", new GameItem(ItemManager.GetPotion("Minor 
Stamina Potion"), "FullSheet", new Rectangle(704, 1344, 32, 32)));
}
```

So, the first change is a local variable, potionData, that is a PotionData object. I initialize the fields to be a healing potion. The name is Minor Healing Potion, the Type is Potion, the Target is Health, the price is 100, the weight is .25, the minimum is 6 and the maximum is 13, effectively giving a range of 6 to 12. Also, the allowed class is all four clases. I then create a Potion object using the data object. The potion object is then added to the ItemManager. I repeat the process for mana potions and stamina potions. I then create GameItems and add them to the GameItemManager.

Now that we have potions we should set up bandits to drop a potion for the player, mostly for testing purposes. Update the LoadWorld method of the CharacterGeneratorScreen to the following.

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

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

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

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

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

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

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

                character = new NonPlayerCharacter(entity, sprite);

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

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

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

    Level level = new Level(map);

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

    Chest chest = new Chest(chestData);

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

    level.Chests.Add(itemSprite);

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

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

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

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

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

    NonPlayerCharacter npc = new NonPlayerCharacter(e, s);

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

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

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

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

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

    for (int i = 0; i < 25; i++)
    {
        ed = new EntityData("Bandit", 1, 10, 12, 12, 10, 10, 10, "20|CON|10", "12|WIL|12", "0|
0|0");

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

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

        Mob mob = new Bandit(e, s);

        Rectangle r = new Rectangle(Mechanics.Random.Next(10, 20) * 32,
Mechanics.Random.Next(10, 20) * 32, 32, 32);

        if (!mobLayer.Mobs.ContainsKey(r))
        {
            mobLayer.Mobs.Add(r, mob);
        }

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

The last thing I'm going to cover in this tutorial is incrementing experience when a mob dies. To do that, I added a method to the Entity class AddExperience. I did this because the Experience property has a private set and I wanted to keep it that way. Add the following method to the Entity class.

```
public void AddExperience(int experience)
{
    Experience += experience;
}
```

Nothing of difficulty here. I just increment the Experience property by the experience passed in. Now we get to calling this method. That is, of course, done in the CombatScreen class. When pushing the LootScreen we now update the Entity of the player by the experience value of the mob. That is in the Update method and the DoAction method. Replace those methods with the following code.

```
public override void Update(GameTime gameTime)
{
    _queueTimer += gameTime.ElapsedGameTime.TotalSeconds;

    if (_queueTimer > 2)
    {
        if (GameState._descriptions.Count > 0)
        {
            GameState._descriptions.Dequeue();
        }

        _queueTimer = 0;
    }

    if (!_displayActionTexture)
    {
        _scene.Update(gameTime, PlayerIndex.One);
    }
    if (GamePlayScreen.Player.Character.Entity.Health.CurrentValue <= 0)
    {
        StateManager.ChangeState(GameRef.GameOverScreen);
    }

    if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Space) && !
_displayActionTexture)
    {
        switch (_scene.SelectedIndex)
        {
            case 0:
                _queueTimer = 0;
                if (GamePlayScreen.Player.Character.Entity.Dexterity >= _mob.Entity.Dexterity)
                {
                    _descriptions.Enqueue($"You attack the {_mob.Entity.EntityClass}.");
                    _mob.Attack(GamePlayScreen.Player.Character.Entity);

                    if (_mob.Entity.Health.CurrentValue <= 0)
                    {
                        StateManager.PopState();
                        StateManager.PushState(GameRef.LootScreen);
                        GameRef.LootScreen.Gold = _mob.GoldDrop;

                        foreach (var i in _mob.Drops)
                        {
                            GameRef.LootScreen.Items.Add(i);
                        }

                        return;
                    }

                    _descriptions.Enqueue($"The {_mob.Entity.EntityClass} attacks you.");
                    _mob.DoAttack(GamePlayScreen.Player.Character.Entity);
                }
                else
                {
                        _descriptions.Enqueue($"The {_mob.Entity.EntityClass} attacks you.");
                        _mob.DoAttack(GamePlayScreen.Player.Character.Entity);
                        _descriptions.Enqueue($"You attack the {_mob.Entity.EntityClass}.");
                        _mob.Attack(GamePlayScreen.Player.Character.Entity);

                        if (_mob.Entity.Health.CurrentValue <= 0)
                        {
                            StateManager.PopState();
                            StateManager.PushState(GameRef.LootScreen);

                            GamePlayScreen.Player.Character.Entity.AddExperience(_mob.XPValue);
                            foreach (var i in _mob.Drops)
                            {
                                GameRef.LootScreen.Items.Add(i);
                            }

                            return;
                    }
                }
                break;
            case 1:
                _displayActionTexture = true;

                _actions.Clear();
                Entity entity = GamePlayScreen.Player.Character.Entity;

                if (entity.Mana.MaximumValue > 0)
                {
                    foreach (SpellData s in DataManager.SpellData.SpellData.Values)
                    {
                        foreach (string c in s.AllowedClasses)
                        {
                            if (c == entity.EntityClass && entity.Level >= s.LevelRequirement)
                            {
                                _actions.Add(s.Name);
                            }
                        }
                    }
                }
                else
                {
                    foreach (TalentData s in DataManager.TalentData.TalentData.Values)
                    {
                        foreach (string c in s.AllowedClasses)
                        {
                            if (c == entity.EntityClass && entity.Level >= s.LevelRequirement)
                            {
                                _actions.Add(s.Name);
                            }
                        }
                    }
                }
                return;
            case 2:
                Vector2 center = GamePlayScreen.Player.Sprite.Center;
                Vector2 mobCenter = _mob.Sprite.Center;
                _action = 0;

                if (center.Y < mobCenter.Y)
                {
                    GamePlayScreen.Player.Sprite.Position.Y -= 8;
                }
                else if (center.Y > mobCenter.Y)
                {
                    GamePlayScreen.Player.Sprite.Position.Y += 8;
                }
                else if (center.X < mobCenter.X)
                {
                    GamePlayScreen.Player.Sprite.Position.X -= 8;
                }
                else
                {
                    GamePlayScreen.Player.Sprite.Position.X += 8;
                }

                StateManager.PopState();
                break;
        }
    }

    if (_displayActionTexture)
    {
        if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Escape))
        {
            _displayActionTexture = false;
            return;
        }
        else if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Space))
        {
            DoAction();
        }
        else if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Down))
        {
            _action++;

            if (_action >= _actions.Count)
            {
                _action = 0;
            }
        }
        else if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Up))
        {
            _action--;

            if (_action < 0)
            {
                _action = _actions.Count - 1;
            }
        }
    }

    base.Update(gameTime);
}

private void DoAction()
{
    Entity entity = GamePlayScreen.Player.Character.Entity;

    if (entity.Mana.MaximumValue > 0)
    {
        SpellData spell = DataManager.SpellData.SpellData[_actions[_action]];

        if (spell.ActivationCost > entity.Mana.CurrentValue)
        {
            return;
        }

        entity.Mana.Damage((ushort)spell.ActivationCost);

        Spell s = Spell.FromSpellData(spell);

        if (s.SpellType == SpellType.Activated)
        {
            foreach (BaseEffect e in s.Effects)
            {
                switch (e.TargetType)
                {
                    case TargetType.Enemy:
                        e.Apply(_mob.Entity);
                        break;
                    case TargetType.Self:
                        e.Apply(entity);
                        break;
                }
            }
        }
    }
    else
    {
        TalentData talent = DataManager.TalentData.TalentData[_actions[_action]];

        if (talent.ActivationCost > entity.Stamina.CurrentValue)
        {
            return;
        }

        entity.Stamina.Damage((ushort)talent.ActivationCost);

        Talent s = Talent.FromTalentData(talent);

        if (s.TalentType == TalentType.Activated)
        {
            foreach (BaseEffect e in s.Effects)
            {
                switch (e.TargetType)
                {
                    case TargetType.Enemy:
                        e.Apply(_mob.Entity);
                        break;
                    case TargetType.Self:
                        e.Apply(entity);
                        break;
                }
            }
        }
    }

    _displayActionTexture = false;

    if (_mob.Entity.Health.CurrentValue > 0)
    {
        _mob.DoAttack(entity);
    }

    if (_mob.Entity.Health.CurrentValue <= 0)
    {
        GameRef.LootScreen.Gold = _mob.GoldDrop;
        StateManager.PopState();
        StateManager.PushState(GameRef.LootScreen);

        GamePlayScreen.Player.Character.Entity.AddExperience(_mob.XPValue);

        foreach (var i in _mob.Drops)
        {
            GameRef.LootScreen.Items.Add(i);
        }
    }
}
```

All that I do is after pushing the LootScreen on the stack call the new AddExperince method passing in the XPValue property of the mob. So, if you build and run now you can gain experience and gold. Also, you can collect potions but currently there is no way to use them.

I suppose I might as well fix that now rather than in a future tutorial. That will be done in the Entity class. Rather than create a separate method for using and equipping an item, I am just going to use the Equip method. We just need to make sure to remove the item from the backpack or the player will have an infinite amount of the item. Change the Equip method of the Entity class with the following code.

```
public void Equip(GameItem item)
{
    if (!item.Item.AllowableClasses.Contains(EntityClass) && EntityType != EntityType.Monster)
    {
        return;
    }

    if (item.Item is Potion potion)
    {
        potion.Apply(this);
    }

    if (item.Item is Weapon weapon)
    {
        if (mainHand == null)
        {
            mainHand = item;
            item.Item.IsEquiped = true;
        }
        else
        {
            if (mainHand != item)
            {
                mainHand.Item.IsEquiped = false;
                mainHand = item;
                mainHand.Item.IsEquiped = true;
            }
            else
            {
                mainHand = null;
                item.Item.IsEquiped = false;
            }
        }

        if (weapon.NumberHands == ItemClasses.Hands.Both && offHand != null)
        {
            offHand.Item.IsEquiped = false;
            offHand = null;
        }
    }

    if (item.Item is Shield shield)
    {
        if (offHand != null)
        {
            offHand.Item.IsEquiped = false;
        }

        offHand = item;
        offHand.Item.IsEquiped = true;
    }
    if (item.Item is Armor armor)
    {
        if (armor.Location == ArmorLocation.Body)
        {
            if (body != null)
            {
                body.Item.IsEquiped = false;
            }

            body = item;
            body.Item.IsEquiped = true;
        }

        if (armor.Location == ArmorLocation.Head)
        {
            if (head != null)
            {
                head.Item.IsEquiped = false;
            }

            head = item;
            head.Item.IsEquiped = true;
        }

        if (armor.Location == ArmorLocation.Hands)
        {
            if (hands != null)
            {
                hands.Item.IsEquiped = false;
            }

            hands = item;
            hands.Item.IsEquiped = true;
        }

        if (armor.Location == ArmorLocation.Head)
        {
            if (feet != null)
            {
                feet.Item.IsEquiped = false;
            }

            feet = item;
            feet.Item.IsEquiped = true;
        }
    }
}
```

Removing the item needs to happen in the InventoryScreen. This is because the backpack is part of the player and the library has no idea what the player is. Also, we should break out of the loop when using the potion. Modify the Update method of the InventoryScreen class to the following.

```
public override void Update(GameTime gameTime)
{
    if (InputHandler.CheckMouseReleased(MouseButton.Right) ||
InputHandler.KeyReleased(Keys.Escape))
    {
        StateManager.PopState();
    }
    if (InputHandler.CheckMouseReleased(MouseButton.Left))
    {
        if (new Rectangle(1175, 45, 42, 32).Contains(InputHandler.MouseAsPoint))
        {
            StateManager.PopState();
        }

        for (int i = 0; i < _backpackDestinations.Count && i <
GamePlayScreen.Player.Backpack.Items.Count; i++)
        {
            if (_backpackDestinations[i].Contains(InputHandler.MouseAsPoint))
            {

GamePlayScreen.Player.Character.Entity.Equip(GamePlayScreen.Player.Backpack.Items[i]);

                if (GamePlayScreen.Player.Backpack.Items[i].Item is Potion)
                {
                    GamePlayScreen.Player.Backpack.Items.RemoveAt(i);
                    break;
                }
                break;
            }
        }
    }
    base.Update(gameTime);
}
```

If you build and run now you can gain gold and experience. Also, mobs will drop potions that you can use to heal the player.

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

Good luck with your Game Programming Adventures!

Cynthia