Eyes of the Dragon Tutorials
Part 49
Combat Engine – Part Four

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.

In this tutorial I will be continuing on with the combat engine. In particular, I will be adding loot drops when the enemy is slain. So, let's get started.

We have a property in the Mob class that holds the items a mob can drop. However, it is not populated right now. Therefore, the first step will be to add something to drop. I did that in the LoadWorld method of the CharacterGeneratorScreen. Updat that method as follows.

```
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);


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

    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(new GameItem(ItemManager.GetWeapon("Long Sword"), "FullSheet", new
Rectangle(1696, 1408, 32, 32)));
    m.Backpack.AddItem(new GameItem(ItemManager.GetWeapon("Short Sword"), "FullSheet", new
Rectangle(800, 1504, 32, 32)));
    m.Backpack.AddItem(new GameItem(ItemManager.GetWeapon("Apprentice Staff"), "FullSheet", new
Rectangle(224, 1408, 32, 32)));
    m.Backpack.AddItem(new GameItem(ItemManager.GetWeapon("Acolyte Staff"), "FullSheet", new
Rectangle(256, 1408, 32, 32)));
    m.Backpack.AddItem(new GameItem(ItemManager.GetArmor("Leather Armor"), "FullSheet", new
Rectangle(1248, 1216, 32, 32)));
    m.Backpack.AddItem(new GameItem(ItemManager.GetArmor("Chain Mail"), "FullSheet", new
Rectangle(1472, 1184, 32, 32)));
    m.Backpack.AddItem(new GameItem(ItemManager.GetArmor("Studded Leather Armor"), "FullSheet",
new Rectangle(1984, 1120, 32, 32)));
    m.Backpack.AddItem(new GameItem(ItemManager.GetArmor("Light Robes"), "FullSheet", new
Rectangle(992, 1216, 32, 32)));
    m.Backpack.AddItem(new GameItem(ItemManager.GetArmor("Medium Robes"), "FullSheet", new
Rectangle(1024, 1216, 32, 32)));
```

```
    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;

    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);

    ((MobLayer)world.Levels[world.CurrentLevel].Map.Layers.Find(x => x is
MobLayer)).Mobs.Add(new Rectangle(0, 512, 32, 32), mob);

    mob.Entity.Equip(new GameItem(ItemManager.GetWeapon("Short Sword"), "FullSheet", new
Rectangle(800, 1504, 32, 32)));
    mob.Drops.Add(new GameItem(ItemManager.GetWeapon("Short Sword"), "FullSheet", new
Rectangle(800, 1504, 32, 32)));

    //
((NonPlayerCharacter)world.Levels[world.CurrentLevel].Characters[0]).SetConversation("eliza1");
}
```

All I do is add a GameItem that represents a short sword to the Drops property of the Mob class. The next step will be to add a new state that holds the loot that is dropped when an enemy is defeated. I would normally use a texture for this. However, I don't think that downloading and adding one to the project is a good idea. Instead, I will just render the items on the screen with a link label to close the screen. Right click the GameScreens folder in the EyesOfTheDragon project, select Add and then Class. Name this new class LootScreen. Here is the initial code for the screen. We will update it later on.

```
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 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)
            };

            ControlManager.Add(linkLabel);
        }

        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);
                }
            }
        }

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

            GameRef.SpriteBatch.Begin();

            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();
        }
    }
}
```

There are using statements to bring MGRpgLibrary classes and MonoGame classes into scope. The class inherits from BaseGameState, like all of our game states. It has a readonly field, Items, that holds the items that were dropped when the mob was defeated. The _over field tells if the mouse is over an items that is being rendered. The _index field is the index of the items the mouse is over, or -1. The constructor initializes the Items field.

In the LoadContent method I create a LinkLabel that will be used to close the screen. I initialize the Text, TabStop, Color, SelectedColor, and Position properties. The interesting initialization is the one for the position. It is set to be centered on the screen horizontally by taking the width of the screen, the width of the text and dividing it by 2. I position the LinkLabel to be the bottom of the screen minus twice the line spacing of the font. The control is then added to the ControlManager field of the class.

In the Update method I check to see if the _over field is true, meaning it is over an item. If it is and the left mouse button has been release as well as the _index field being greater than -1 and less than the number of items, I add the item to the player's backpack. I then remove the item from the list of items.

In the Draw method I call Begin on the sprite batch to begin rendering. I then create a destination rectangle to control where the item is rendered, the same way I did in the InventoryScreen. I then grab the position of the mouse as a point. There is a local variable the is the index of the item currently being rendered. I set the _over field to false and the _index field to -1, meaning the mouse is not over anything. I then loop over all of the items in the Items field. I call the Draw method of the item passing in the sprite batch object and the destination rectangle. If the destination rectangle contains the mouse I set the _over property to true and the index equal to the counter. I then increment the X property of the destination rectangle by 50, the width of the items plus some padding. If that is greater than the width of the screen minus 50 and 32 I set the X property back to 50 and increment the Y property by 50, the height of an item plus a little padding. The next step is to increment that counter by one. Finally, I call the End method of the sprite batch.

I added an override of the Show method. It removes the items from the Items collection. It then calls the base Show method to display the screen.

What needs to happen next is adding a property to the Game1 class for this new screen. Add the following property to the Game1 class for this new state and update the constructor to initialize it.

```
public LootScreen LootScreen { get; private set; }

public Game1()
{
    _graphics = new GraphicsDeviceManager(this);
    ScreenRectangle = new Rectangle(
        0,
        0,
        ScreenWidth,
        ScreenHeight);
    IsMouseVisible = true;
```

```
    Content.RootDirectory = "Content";

    Components.Add(new InputHandler(this));

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

    _ = new TextureManager();

    TitleScreen = new TitleScreen(this, _gameStateManager);
    StartMenuScreen = new StartMenuScreen(this, _gameStateManager);
    GamePlayScreen = new GamePlayScreen(this, _gameStateManager);
    CharacterGeneratorScreen = new CharacterGeneratorScreen(this, _gameStateManager);
    SkillScreen = new SkillScreen(this, _gameStateManager);
    LoadGameScreen = new LoadGameScreen(this, _gameStateManager);
    ConversationScreen = new ConversationScreen(this, _gameStateManager);
    ShopScreen = new ShopState(this, _gameStateManager);
    InventoryScreen = new InventoryScreen(this, _gameStateManager);
    CombatScreen = new CombatScreen(this, _gameStateManager);
    GameOverScreen = new GameOverScreen(this, _gameStateManager);
    LootScreen = new LootScreen(this, _gameStateManager);

    _gameStateManager.ChangeState(TitleScreen);

    IsFixedTimeStep = false;
    _graphics.SynchronizeWithVerticalRetrace = false;
}
```

Nothing there that you haven't seen before. The next step will be to display the screen when a mob dies. That will happen in the CombatScreen. I do that where I pop the state off the stack. Replace the Update method of the CombatScreen with the following.

```
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);

                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;
            }
        }
    }

    base.Update(gameTime);
}
```

All the new code does is if the mob has died after popping the combat state off the stack is push the loot state onto the stack. It then loops over the Drops collection and adds each item to the Items field of the LootScreen.

I don't handle mobs dying if a talent or spell is used. I added a check in the DoAction method that checks to see if the mob's current health is less than or equal to zero. If it is, I do exactly what I just did in the Update method. Replace the DoAction method of the CombatScreen to the following.

```
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)
    {
        StateManager.PopState();
        StateManager.PushState(GameRef.LootScreen);

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

The issue now is there is no way to leave the LootScreen. What I did is update the Update method to exit if the escape key is pressed or the link label is selected. Update the Update LoadContent methods of the LootScreen class to the following. Also, add the following event handler for the link label being selected.

```
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)
{
    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))
    {
        StateManager.PopState();
    }
}
```

I discovered one bug. The control manager is not being drawn, or updated. Change the Update and Draw methods to the following.

```
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))
    {
        StateManager.PopState();
    }

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

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

    GameRef.SpriteBatch.Begin();

    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();
}
```

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