Eyes of the Dragon Tutorials
Part 47
Combat Engine – Part Two

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. Things are working okay but there are a few issues. First, there is no way to win or lose a combat, only flee. Second, there is no way to use a spell or talent. Also, a mob cannot hit back if it is hit. This tutorial will address some of these issues.

I am going to tackle the third issue first, a mob hitting back. Ideally what we want is the entity with the highest dexterity to hit first and then the other to hit back, if it is still alive. In the event of a tie I will give first strike to the player. First, I want to make changes to the Mob and Bandit classes. I am going to change the signature of the Attack method, that is called when the player attacks a mob. I am also going to add another abstract method DoAttack that is called when the mob attacks an entity. Update the Mob class to the following.

```
using MGRpgLibrary.ItemClasses;
using MGRpgLibrary.SpriteClasses;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using RpgLibrary.CharacterClasses;
using System;
using System.Collections.Generic;
using System.Text;

namespace MGRpgLibrary.Mobs
{
    public abstract class Mob
    {
        #region Constant Region

        public const int AttackRadius = 32;

        #endregion

        #region Field Region

        protected Entity entity;
        protected AnimatedSprite sprite;

        #endregion

        #region Property Region

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

```
public AnimatedSprite Sprite
{
    get { return sprite; }
}

#endregion

#region Money Region

private int _gold;

public int Gold { get => _gold; }

public void UpdateGold(int amount)
{
    _gold += amount;
}

#endregion

#region Item Drop Region

protected List<GameItem> _drops = new List<GameItem>();

public List<GameItem> Drops { get => _drops; }

#endregion

#region Constructor Region

public Mob(Entity entity, AnimatedSprite sprite)
{
    this.entity = entity;
    this.sprite = sprite;
}

#endregion

#region Method Region
#endregion

#region Virtual Method region

public virtual void Update(GameTime gameTime)
{
    entity.Update(gameTime.ElapsedGameTime);
    sprite.Update(gameTime);
}

public virtual void Draw(GameTime gameTime, SpriteBatch spriteBatch)
{
    sprite.Draw(gameTime, spriteBatch);
}

#endregion

#region Abstract Method Region

public abstract void Attack(Entity source);
```

```
        public abstract void DoAttack(Entity target);

        #endregion
    }
}
```

I just renamed the parameter of the Attack method from target to source. I then added a new method DoAttack that takes an Entity parameter that is the target for the attack.

Now, the Attack method for the Bandit class changed, and I added a DoAttack method. Replace the Bandit class with the following code.

```
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)
        {
        }

        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 Attack method has changed slightly, other than the change in its signature. It rolls a D20 and compares it to 10 plus the dexterity modifier of the bandit. If that is greater than or equal to that value I call the ApplyDamage method passing in the main hand weapon of the source. The DoAttack method works in essentially the same way. It checks to see if the roll of a D20 is greater than or equal to 10 plus the dexterity modifier of the target. If it is, it calls the ApplyDamage method of the target passing in the main hand weapon of the bandit.

With that out of the way, lets turn our attention back to the combat screen. What we want to do is check to see if the mob or the player have the highest dexterity, giving a tie to the player. If the player

wins we will call the Attack method on the mob and then the DoAttack method on the mob, if it is still alive. Modify the Update method of the CombatScreen class to the following.

```
public override void Update(GameTime gameTime)
{
    _scene.Update(gameTime, PlayerIndex.One);

    if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Space))
    {
        switch (_scene.SelectedIndex)
        {
            case 0:
                if (GamePlayScreen.Player.Character.Entity.Dexterity >= _mob.Entity.Dexterity)
                {
                    _mob.Attack(GamePlayScreen.Player.Character.Entity);

                    if (_mob.Entity.Health.CurrentValue <= 0)
                    {
                        StateManager.PopState();
                        return;
                    }

                    _mob.DoAttack(GamePlayScreen.Player.Character.Entity);
                }
                else
                {
                    _mob.DoAttack(GamePlayScreen.Player.Character.Entity);
                    _mob.Attack(GamePlayScreen.Player.Character.Entity);

                    if (_mob.Entity.Health.CurrentValue <= 0)
                    {
                        StateManager.PopState();
                        return;
                    }
                }
                break;
            case 2:
                Vector2 center = GamePlayScreen.Player.Sprite.Center;
                Vector2 mobCenter = _mob.Sprite.Center;

                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;
```

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

If you run and build now you can initiate combat with the bandit, run away and win the combat. There is an issue, though. Even if you win the combat you will be thrown right back in. What we need to do is remove mobs from the map that have health less than or equal to 0. I did that in the Update method of the GamePlayState. Change that method as follows.

```
public override void Update(GameTime gameTime)
{
    world.Update(gameTime);
    player.Update(gameTime);
    player.Camera.LockToSprite(player.Sprite);

    if (InputHandler.KeyReleased(Keys.Space) ||
        InputHandler.ButtonReleased(Buttons.A, PlayerIndex.One))
    {
        foreach (ILayer layer in World.Levels[World.CurrentLevel].Map.Layers)
        {
            if (layer is CharacterLayer)
            {
                foreach (Character c in ((CharacterLayer)layer).Characters.Values)
                {
                    float distance = Vector2.Distance(
                        player.Sprite.Center,
                        c.Sprite.Center);

                    if (distance < Character.SpeakingRadius && c is NonPlayerCharacter)
                    {
                        NonPlayerCharacter npc = (NonPlayerCharacter)c;

                        if (npc.HasConversation)
                        {
                            StateManager.PushState(GameRef.ConversationScreen);

                            GameRef.ConversationScreen.SetConversation(
                                player,
                                npc,
                                npc.CurrentConversation);

                            GameRef.ConversationScreen.StartConversation();
                        }
                    }
                    else if (distance < Character.SpeakingRadius && c is Merchant)
                    {
                        StateManager.PushState(GameRef.ShopScreen);
                        GameRef.ShopScreen.SetMerchant(c as Merchant);
                    }
                }
            }
        }
    }

    MobLayer mobLayer = World.Levels[World.CurrentLevel].Map.Layers.Find(x => x is MobLayer) as
MobLayer;
```

```
    foreach (var mob in mobLayer.Mobs.Where(kv => kv.Value.Entity.Health.CurrentValue <=
0).ToList())
    {
        mobLayer.Mobs.Remove(mob.Key);
    }

    foreach (Rectangle r in mobLayer.Mobs.Keys)
    {
        float distance = Vector2.Distance(mobLayer.Mobs[r].Sprite.Center,
player.Sprite.Center);

        if (distance < Mob.AttackRadius)
        {
            GameRef.CombatScreen.SetMob(mobLayer.Mobs[r]);

            StateManager.PushState(GameRef.CombatScreen);
            Visible = true;
        }
    }

    if (InputHandler.KeyReleased(Keys.I))
    {
        StateManager.PushState(GameRef.InventoryScreen);
    }

    base.Update(gameTime);
}
```

So, what happens is there is a foreach loop above the code that checks for combat that loops over all of the mobs in the mob layer where the current health is less than or equal to zero as a List. It then calls the Remove method of the mob collection to remove the mob, passing in the key. I had to do it this way because the Dictionary class does not have a RemoveAll method.

If you build and run now, you can kill the bandit but you never die. Also, I will comment that it is pointless to try and attack the bandit if you haven't purchased a sword from the merchant and equipped it. You would do no damage against it and it would kill you, except you can't die right now. I will fix the dying part first. Right click the GameScreens folder, select Add and then Class. Name this new class GameOverScreen. Here is the code for that class.

```
using MGRpgLibrary;
using Microsoft.Xna.Framework;
using System;
using System.Collections.Generic;
using System.Text;

namespace EyesOfTheDragon.GameScreens
{
    public class GameOverScreen : BaseGameState
    {
        public GameOverScreen(Game game, GameStateManager manager) : base(game, manager)
        {
        }

        public override void Update(GameTime gameTime)
        {
            if (InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Space) ||
                InputHandler.KeyReleased(Microsoft.Xna.Framework.Input.Keys.Enter) ||
```

```
                InputHandler.CheckMouseReleased(MouseButton.Left))
            {
                StateManager.ChangeState(GameRef.StartMenuScreen);
            }

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

A very simple state, for now. It just waits for the player to hit the spacebar, enter key or click the left mouse button. I will add a graphic for it in a future tutorial. For now, it suits out purposes. If one of those cases is true it calls ChangeState of the state manager to change the state back to the start menu.

Like all of the other states we need a property for it and to create it. Add the following property to the Game1 class and update the constructor to the following.

```
public GameOverScreen GameOverScreen { 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);

    _gameStateManager.ChangeState(TitleScreen);

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

The next step will be to switch to this state if the player dies. That will be done in the CombatScreen class. In particular, the Update method. Change the Update method of the CombatScreen class to the following.

```
public override void Update(GameTime gameTime)
{
    _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))
    {
        switch (_scene.SelectedIndex)
        {
            case 0:
                if (GamePlayScreen.Player.Character.Entity.Dexterity >= _mob.Entity.Dexterity)
                {
                    _mob.Attack(GamePlayScreen.Player.Character.Entity);

                    if (_mob.Entity.Health.CurrentValue <= 0)
                    {
                        StateManager.PopState();
                        return;
                    }

                    _mob.DoAttack(GamePlayScreen.Player.Character.Entity);
                }
                else
                {
                    _mob.DoAttack(GamePlayScreen.Player.Character.Entity);
                    _mob.Attack(GamePlayScreen.Player.Character.Entity);

                    if (_mob.Entity.Health.CurrentValue <= 0)
                    {
                        StateManager.PopState();
                        return;
                    }
                }
                break;
            case 2:
                Vector2 center = GamePlayScreen.Player.Sprite.Center;
                Vector2 mobCenter = _mob.Sprite.Center;

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

All the new code does is check to see if the player's current health is less than or equal to 0. If it is, it changes the state to the new game over state.

I need to make an update to the Equip method of the Entity class. I need to check to see if the entity that is equipping the item is a mob when checking to see if the entity can equip in item or not. Replace that method with the following code.

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

    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;
        }
    }
}
```

If you build and run now, you can battle the bandit and win or lose the fight. I was going to tackle spells and talents this tutorial but I think I will wait and give them a tutorial of there own.

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