CM3045 – 3D Graphics & Animation

Peer-Graded Assignment 3: Key Frame Animation

Report by Hristo Stantchev

Abstract

This document covers my submission the 3D graphics techniques behind it and the reasons for using them. All code and techniques are my own or acquired throughout this course and my 3 years of hobbyist game developer experience, all assets I borrowed are found in the ***References & Credits*** section.

Introduction

"Slime Slayer" is an endless 3D action game working where players fight endless waves of slimes.

Players first see title, instructions, author credits, start button, and background (extract from in-game level). On start, players spawn in a dungeon – 1 textured plane (floor), and 4 textured planes (walls). Walls hold torches with flames (particle system) and an animated light (more in ***Extension Tasks***). Players fight randomly spawned slimes. Slimes (model & materials by me) walk towards the player using *transform.LookAt(player) & AddForce(transform.forward)* (local Z axis). Force mode is impulse - instantaneous and uses mass. Slimes hold tiers that dictate mass & scale. Upon being hit by the player, they explode (particle system) and 2 smaller slimes are instantiated.

Through Unity events, every time a slime is hit, score is added for their "tier" (size & mass), and when all slimes are defeated, a new random set of slimes is spawned. UI features a HUD showing the score and "wave", a Pause menu (setting time scale to 0f) with a Resume (resetting time scale to 1f) and Main Menu button (using Scene Management). The player character (more in ***Task 2***) can move, attack, stay idle, and be defeated upon being hit by a slime. This shows them the game over menu featuring a Restart and Main Menu button, as well as comparing their current score to their persistent high score (using Player Prefs).

Task 1: Hand Animation

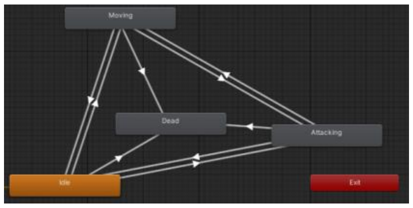
I was confused by the term "hand animation" in the task – whether it meant handmade animations or animations of a hand. Therefore, I did both. Each animation is made by me and the player character model has a rig attached to it that with a set of animations including hand motions (more in ***Task 2*** and ***Extension Task***).

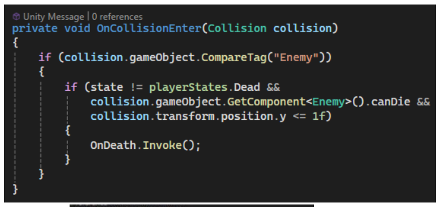
The slimes are animated to shrink and expand in size as they move using Transform Scale and linearly interpolating that through 3 key frames: 1 – original size, 2 – bigger, 3 – original size. These are spread out evenly across 1 second using 30fps.

The lights in the scene all fluctuate through key frame animation (more in ***Extension Tasks***).

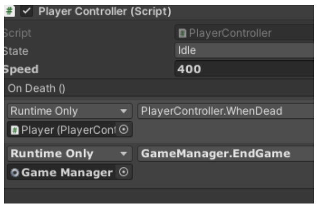
Task 2: Keyboard Controlled Animation State Machine

The player character works through a keyboard controlled FSM implemented through an enum and fed into the animator (animation controller). The states are Idle, Moving, Attacking, Dead. The logic is the following:

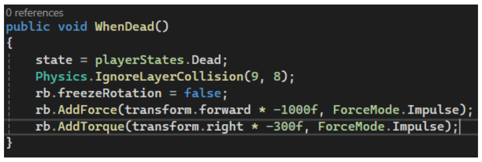




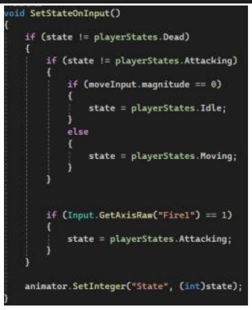
```
Unity Message | 0 references
private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.CompareTag("Enemy"))
    {
        if (state != playerStates.Dead &&
            collision.gameObject.GetComponent<Enemy>().canDie &&
            collision.transform.position.y <= 1f)
        {
            OnDeath.Invoke();
        }
    }
}
```



```
# ✓ Player Controller (Script)
Script          PlayerController
State           Idle
Speed           400
On Death ()
Runtime Only      PlayerController.WhenDead
Player (PlayerCont
Runtime Only      GameManager.EndGame
Game Manager
```



```
0 references
public void WhenDead()
{
    state = playerStates.Dead;
    Physics.IgnoreLayerCollision(9, 8);
    rb.freezeRotation = false;
    rb.AddForce(transform.forward * -1000f, ForceMode.Impulse);
    rb.AddTorque(transform.right * -300f, ForceMode.Impulse);
}
```

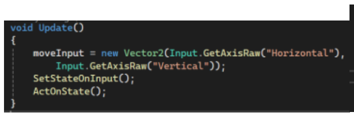


```
void SetStateOnInput()
{
    if (state != playerStates.Dead)
    {
        if (state != playerStates.Attacking)
        {
            if (moveInput.magnitude == 0)
            {
                state = playerStates.Idle;
            }
            else
            {
                state = playerStates.Moving;
            }
        }


        if (Input.GetAxisRaw("Fire1") == 1)
        {
            state = playerStates.Attacking;
        }
    }

    animator.SetInteger("State", (int)state);
}
```

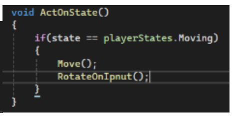
(Logic for movement & death, moveInput is simply X – Horizontal raw axis, Y – Vertical raw axis)

Extension Tasks: Animated Character, Animating beyond transforms

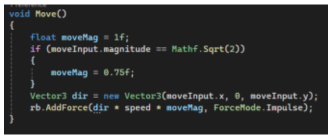
The character also moves and acts using animations:



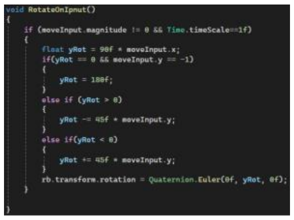
```
void Update()
{
    moveInput = new Vector2(Input.GetAxisRaw("Horizontal"),
        Input.GetAxisRaw("Vertical"));
    SetStateOnInput();
    ActOnState();
}
```



```
void ActOnState()
{
    if(state == playerStates.Moving)
    {
        Move();
        RotateOnIpnut();
    }
}
```

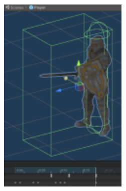
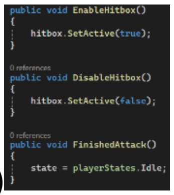


```
void Move()
{
    float moveMag = 1f;
    if (moveInput.magnitude == Mathf.Sqrt(2))
    {
        moveMag = 0.75f;
    }
    Vector3 dir = new Vector3(moveInput.x, 0, moveInput.y);
    rb.AddForce(dir * speed * moveMag, ForceMode.Impulse);
}
```

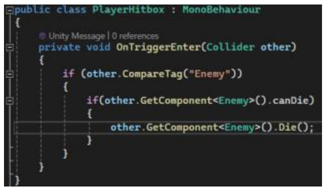


```
void RotateOnIpnut()
{
    if (moveInput.magnitude != 0 && Time.timeScale==1f)
    {
        float yRot = 90f * moveInput.x;
        if(yRot == 0 && moveInput.y == -1)
        {
            yRot = 180f;
        }
        else if (yRot > 0)
        {
            yRot -= 45f * moveInput.y;
        }
        else if(yRot < 0)
        {
            yRot += 45f * moveInput.y;
        }
        rb.transform.rotation = Quaternion.Euler(0f, yRot, 0f);
    }

}
```

(moveMag is used to slow down diagonal movement, RotateOnInput gets degrees to rotate the y Axis in 8 possible directions and transform them into Euler Angles for Rigid Body rotation)

 (stacked key frames for attack weight)

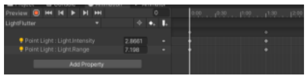
```
public void EnableHitbox()
{
    hitbox.SetActive(true);
}

0 references
public void DisableHitbox()
{
    hitbox.SetActive(false);
}

0 references
public void FinishedAttack()
{
    state = playerStates.Idle;
}
```



```
public class PlayerHitbox : MonoBehaviour
{
    Unity Message | 0 references
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Enemy"))
        {
            if(other.GetComponent<Enemy>().canDie)
            {
                other.GetComponent<Enemy>().Die();
            }
        }
    }
}
```

(Logic for enabling & disabling Box Trigger to attack enemies using Animation Events & for Trigger)

Finally, torches change intensity & range for a pulsating light effect using key frames:



```
Preview                          0
LightFlutter
  Point Light : Light.Intensity  2.8661
  Point Light : Light.Range      7.198
           Add Property
```

Conclusion & Evaluation

These are the basic building blocks of the game. I am ultimately proud of my work yet I think it is better to animate humanoid characters in an external program & then fine-tune in Unity. It would have made the player animations more believable. Thank you for your time and attention

References & Credits:

Textures – PolyHaven, CC0 license - Floor, Walls; Prop – Old Torch - Unity Asset Store, Standard Unity Asset Store EULA, Michele Marcelli; Player Character Model & Rig – Paladin W Prop, J Nordstrom, Mixamo