DREAM CATCHER

Made in Unity Engine

in 16 hours

---

1 Code

1.1 Movement

```
private void MovingCharacter() {

    if (!isMovementEnabled) {
    return; // Do nothing if movement is disabled
    }
    //Cursor properties
    if(Time.timeScale == 0){
      Cursor.lockState = CursorLockMode.None;
      Cursor.visible = true;
    }else {
      Cursor.lockState = CursorLockMode.Locked;
      Cursor.visible = false;
    }

    if(Time.timeScale != 0f){
      characterRotationLeftRight = Input.GetAxis("Mouse X") * mouseSensivity;
      transform.Rotate(0, characterRotationLeftRight, 0); //Rotate whole character with camera view.
    }

    vertInput = Input.GetAxis("Vertical"); // W, S
    horizInput = Input.GetAxis("Horizontal"); // A, D

    Vector3 forwardMovement = transform.forward * vertInput;
    Vector3 sideMovement = transform.right * horizInput;

   //SimpleMove realisation, ClampMagnitude is used to fix diagonal acceleration
    characterController.SimpleMove
   (Vector3.ClampMagnitude(forwardMovement + sideMovement, 1.0f) * movementSpeed);
 }
```

1.2 Character Animation

Example

```
void AnimationOn(){
    if(playerAnimator != null){

      movementSpeed = 2.5f;
      AnimationCheck(0);

      if(Input.GetKey(KeyCode.W))
      {
        //Run
        if(Input.GetKey(KeyCode.LeftShift)){
        movementSpeed = 6f;
        AnimationCheck(5);
        }else{

        //Move
        AnimationCheck(3);
        }
      }
    }
  }
```

```
private void AnimationCheck(int numOfAnimation){

      for(int i = 0; i < AnimationsArray.Length; i++){
        if(i == numOfAnimation){
            playerAnimator.SetTrigger(AnimationsArray[numOfAnimation]);
        } else if(i!= 8)
        playerAnimator.ResetTrigger(AnimationsArray[i]);
      }
  }
```

1.3 Character Spell Cast

```
private void OnTriggerEnter(Collider other)
 {
   if (other.CompareTag("BadDream") || other.CompareTag("Boss"))
   {
     //Sound
     audioSource.PlayOneShot(audioClip);

     //Effect
     if (particles != null) {
       particles.Play();
     }
     StartCoroutine(ParticleCoroutine(particles, 0.7f));
     //Punch
     AnimationCheck(8);

     isMovementEnabled = false; // Disable movement when the player collides with the dream
     StartCoroutine(EnableMovementAfterDelay(0.8f)); // Re-enable movement after a delay of 0.7 seconds

     Rigidbody otherRigidbody = other.GetComponent<Rigidbody>();
     if (otherRigidbody != null) {
     otherRigidbody.velocity = Vector3.zero;
     }

     // Destroy the dream and increase score
     StartCoroutine(PunchCoroutine(other.gameObject, 0.7f));
     if (other.CompareTag("Boss")){dreamCatched = dreamCatched + 50;}else dreamCatched++;

   }
 }
```

1.4 Character Delays

```
IEnumerator PunchCoroutine(GameObject objectToDestroy, float delay) {
    yield return new WaitForSeconds(delay);
    Destroy(objectToDestroy);
  }

  IEnumerator ParticleCoroutine(ParticleSystem particle, float delay) {
    yield return new WaitForSeconds(delay);
    particle.Stop();
  }

  IEnumerator EnableMovementAfterDelay(float delay) {
    yield return new WaitForSeconds(delay);
    isMovementEnabled = true;
  }
```

2 Dream Generator

```
// Update the time since the last ball spawn
    timeSinceLastSpawn += Time.deltaTime;
    timeDifficulty += Time.deltaTime;

    if(timeDifficulty >= 30){
      spawnInterval = spawnInterval - 1;
      dreamSpeed++;
      timeDifficulty = 0;
    }

    // If the spawn interval has passed, spawn a ball
    if (timeSinceLastSpawn >= spawnInterval)
    {
      // Reset the time since the last ball spawn
      timeSinceLastSpawn = 0f;

      // Generate a random X position for the ball
      float spawnX = Random.Range(minSpawnX, maxSpawnX);

      // Spawn the ball at the random X position and at the same Y and Z positions as the generator
      GameObject newDream = Instantiate(DreamPrefab, new Vector3(spawnX, transform.position.y,
                                                transform.position.z), Quaternion.identity);

      // Move the ball to the target position
      Vector3 targetPosition = new Vector3(0f, 1f, 0f); // replace with your desired target position
      Rigidbody dreamRigidbody = newDream.GetComponent<Rigidbody>();
      Vector3 direction = (targetPosition - newDream.transform.position).normalized;
      dreamRigidbody.velocity = direction * dreamSpeed;
    }
```