Computer Graphics 1

Tutorial

Assignment 6

Summer Semester 2024

Ludwig-Maximilians-Universität München

Contact

If you have any questions:

cg1ss24@medien.ifi.lmu.de

Organization

- **Theoretical part:**
  - Prepares you for the exam
  - Solve the tasks at home
  - We present the solutions during the tutorial sessions
- **Practical part:**
  - We will indicate which parts you need to do at home
  - Else this will be done & explained during the tutorial session
  - Ask questions!

Task 1: Interpolation

- **Linear Interpolation:**
  - straight line between two points and take values on the straight line as approximation
  - Advantage: quick and easy
  - Disadvantage: not very precise
- **Non-Linear Interpolation:**
  - Piecewise described by polynomials
  - dividing sections
  - Advantage: smoother, more accurate
  - Disadvantage: complex

Task 2: Bilinear Interpolation

- Problem:

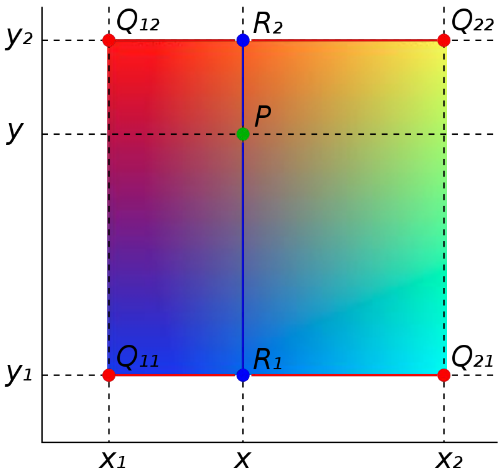
  We have the colors at each corner of the rectangle (Q_{11} - Q_{22})

  → How can we figure out the color at Point P?

- Solution:

  Linear Interpolation in 2 Dimensions!

  - First do linear interpolation in x-direction between Q_{12} and Q_{22} to color value at R_2
  - Do the same for Q_{11} and Q_{21} for R_1
  - Lastly, perform a linear interpolation between R_1 and R_2 to get value at P



Task 2: Bilinear Interpolation

- Linear Interpolation in x-direction for R_1 and R_2

$$\begin{aligned} f(x, y_1) &= \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}), \\ f(x, y_2) &= \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}). \end{aligned}$$

- Linear Interpolation in y-direction

$$f(x, y) = \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2)$$

- Combined formula to do calculation in one step (Bilinear interpolation):

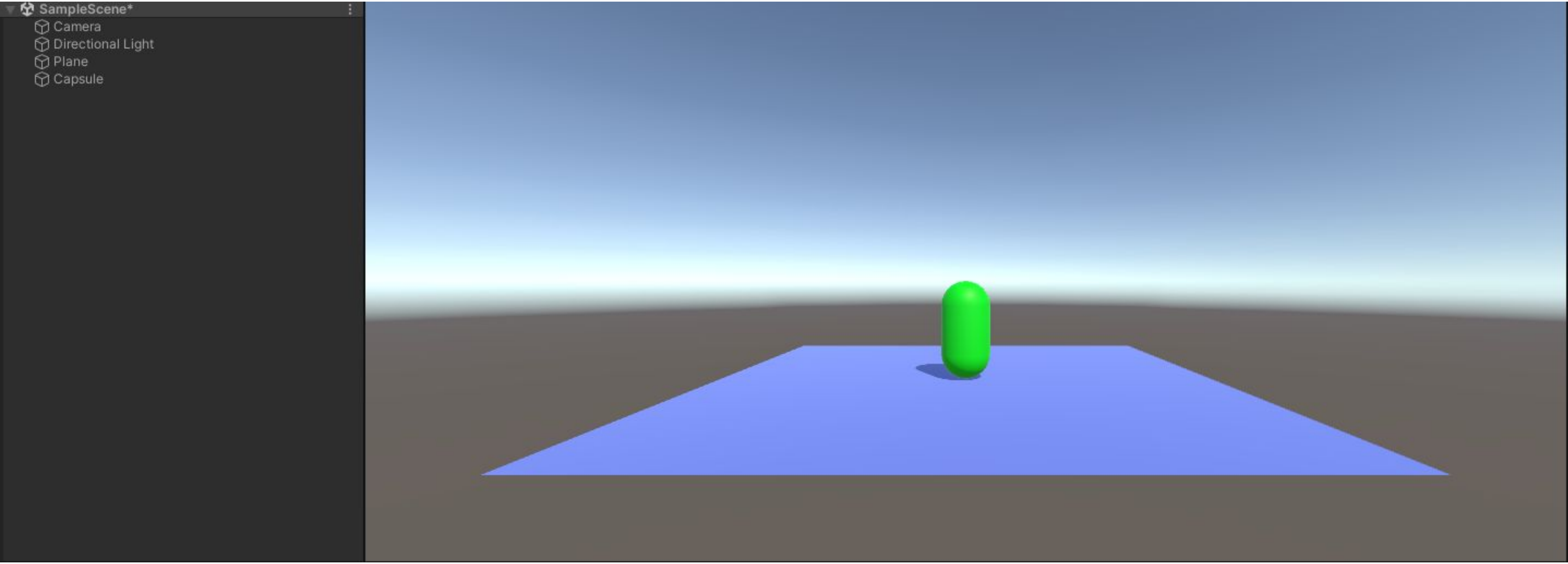
$$f(x, y) = \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right)$$

Task 2: Bilinear Interpolation

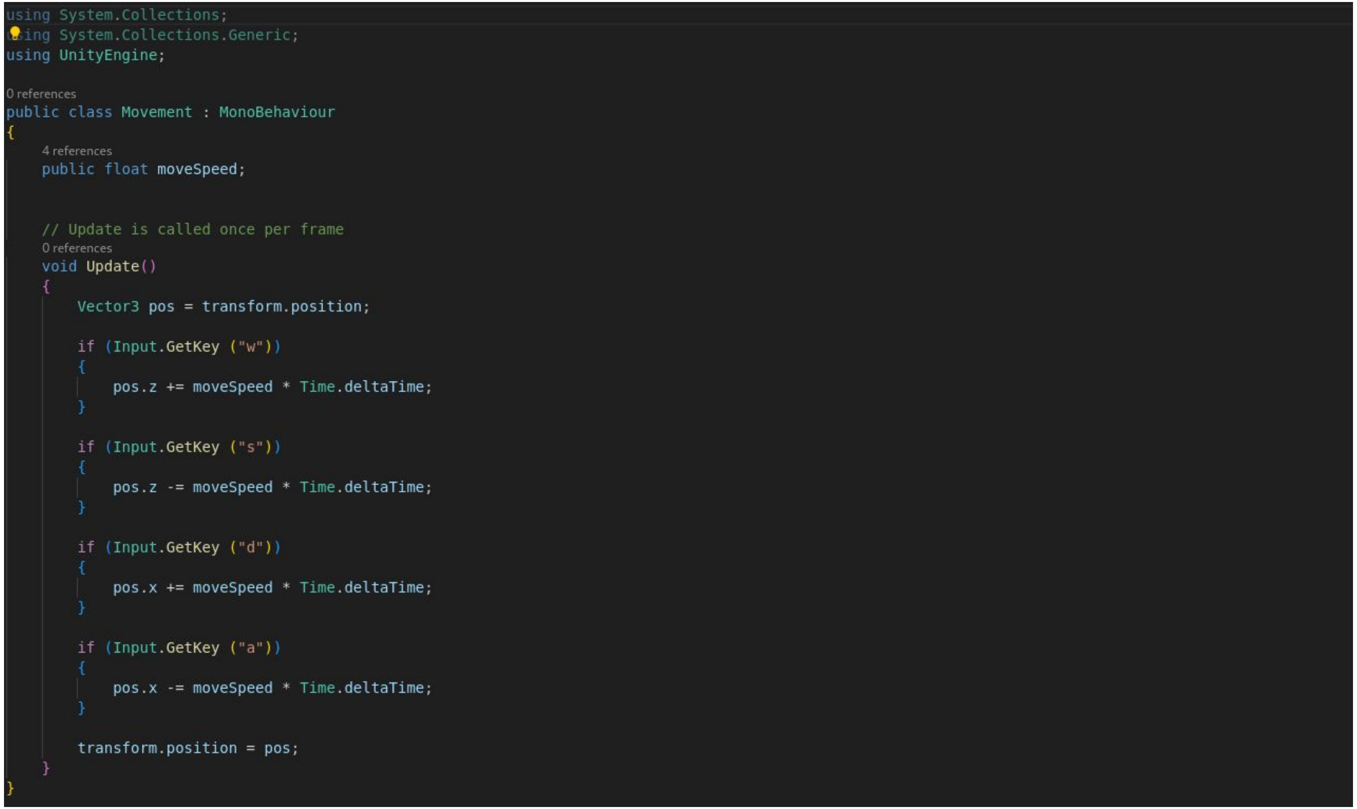
- Color at $P_1 = (0.3, 0.3)$: $P_1 = (76.5, 76.5, 178.5)$

- Color at $P_2 = (0.5, 0.5)$: $P_2 = (127.5, 127.5, 127.5)$

Task 3: Interaction

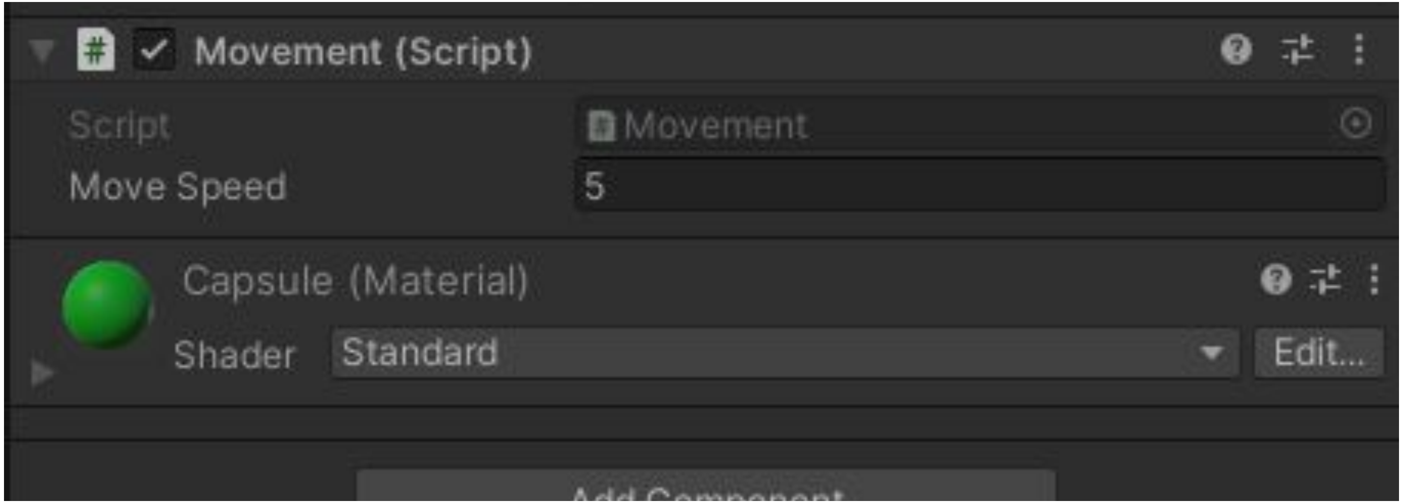


Task 3: Interaction



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 references
public class Movement : MonoBehaviour
{
    4 references
    public float moveSpeed;


    // Update is called once per frame
    0 references
    void Update()
    {
        Vector3 pos = transform.position;

        if (Input.GetKey ("w"))
        {
            pos.z += moveSpeed * Time.deltaTime;
        }

        if (Input.GetKey ("s"))
        {
            pos.z -= moveSpeed * Time.deltaTime;
        }

        if (Input.GetKey ("d"))
        {
            pos.x += moveSpeed * Time.deltaTime;
        }

        if (Input.GetKey ("a"))
        {
            pos.x -= moveSpeed * Time.deltaTime;
        }

        transform.position = pos;
    }
}
```

Task 3: Interaction



Movement (Script)

Script: Movement

Move Speed: 5

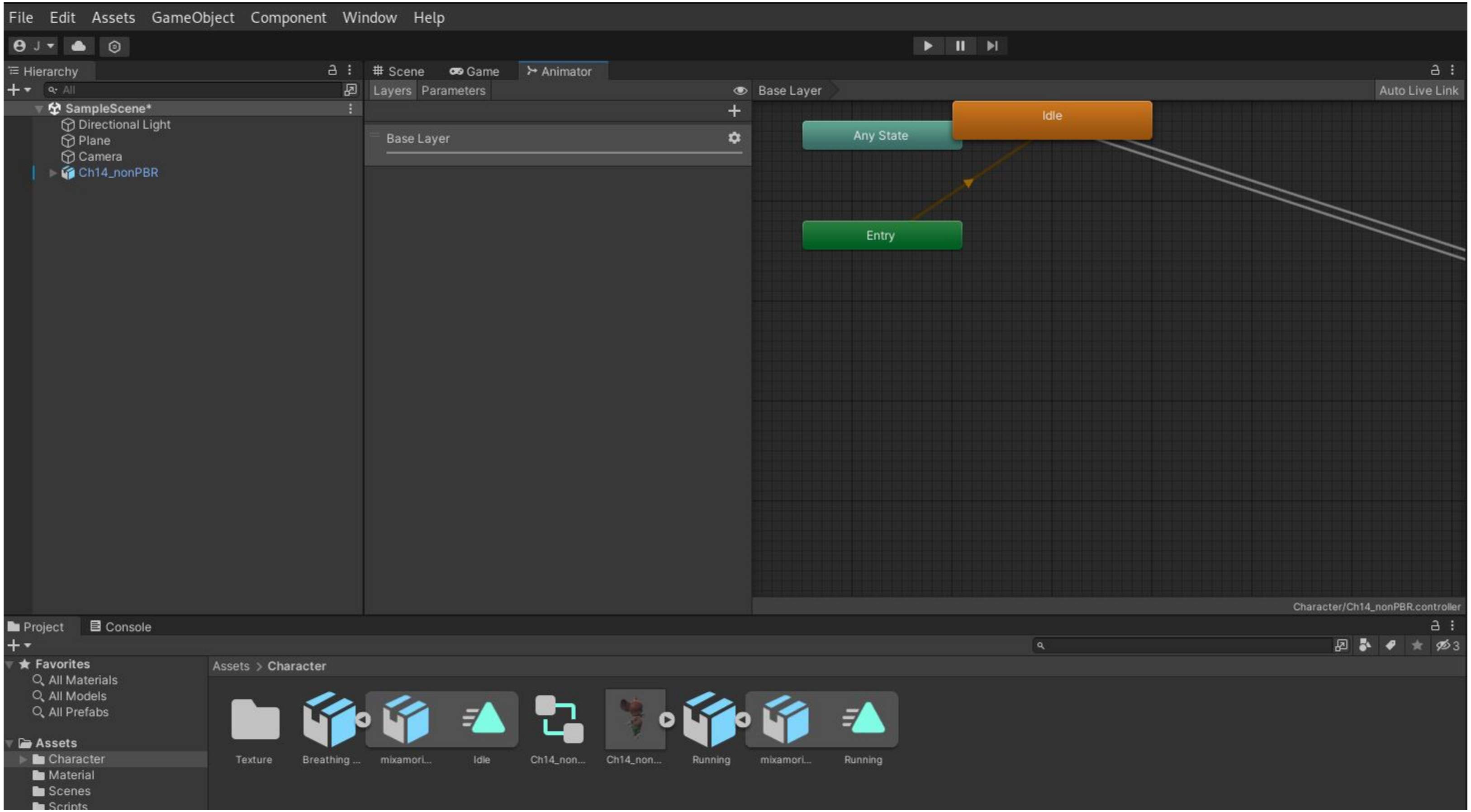
Capsule (Material)

Shader: Standard

Task 4: Animation

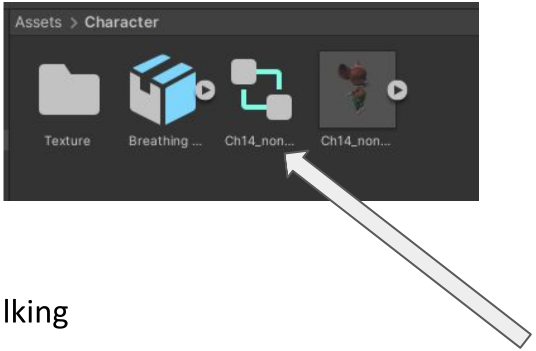
- **Mixamo to get character (registration necessary)**
  - Select character you like (e.g. Mousey)
  - Download -> FBX for Unity -> T-Pose -> Save in Unity Asset Folder
  - No Material Fix: Character -> Inspector -> Material -> Extract Texture
    - If Normal Map Error -> Fix now
- **Mixamo to get animations:**
  - Search for Idle, Running (for running check “In Place”)
  - Download -> everything as selected except Skin -> set Skin to without Skin
  - Import Idle -> Inspector -> Animation -> Check Loop Time and Loop Pose -> Apply
  - Expand Animation to see Triangle -> Drag and Drop Triangle on Character in hierarchy
  - Repeat with Running

Task 4: Animation

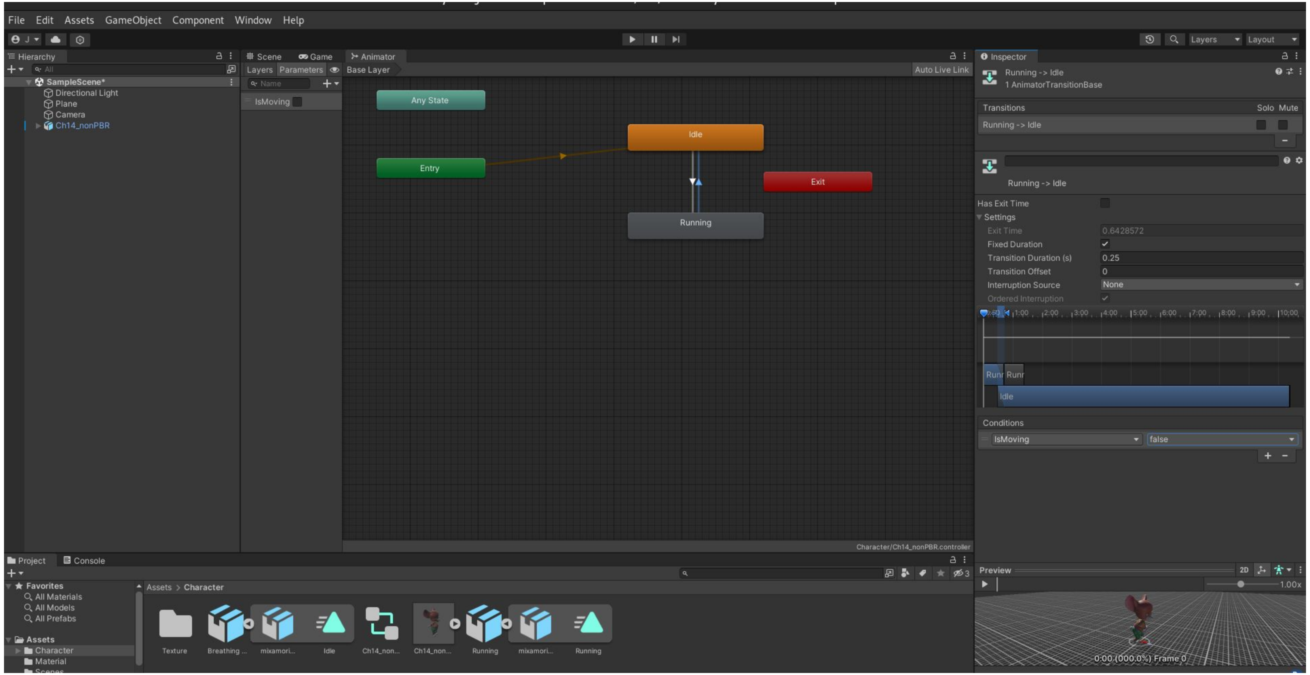


Task 4: Animation

- **Animator**
  - Double Click on controller
- **Transition from Idle to Running**
  - Right Click on Idle Tile -> Make Transition -> Click on Running
  - Click on Transition -> Add Bool (+ in Parameters) -> Name: IsWalking
  - Click on Transition -> Settings -> Uncheck Has Exit Time -> + on Conditions
  - Right Click Running -> Make Transition -> Click on Idle
  - again uncheck Has Exit Time -> + on Conditions -> set to false



Task 4: Animation



Running -> Idle
1 AnimatorTransitionBase

Transitions	Solo	Mute
Running -> Idle	□	□

Running -> Idle

Has Exit Time □

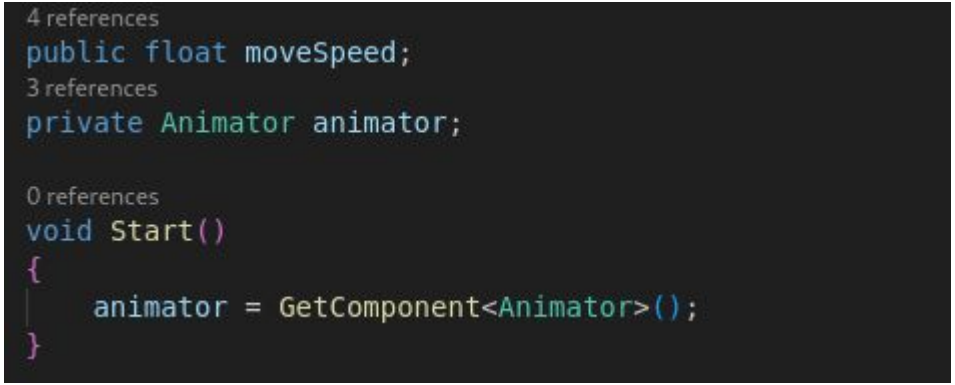
Settings

Exit Time	0.6428572
Fixed Duration	☑
Transition Duration (s)	0.25
Transition Offset	0
Interruption Source	None
Ordered Interruption	☑

Conditions

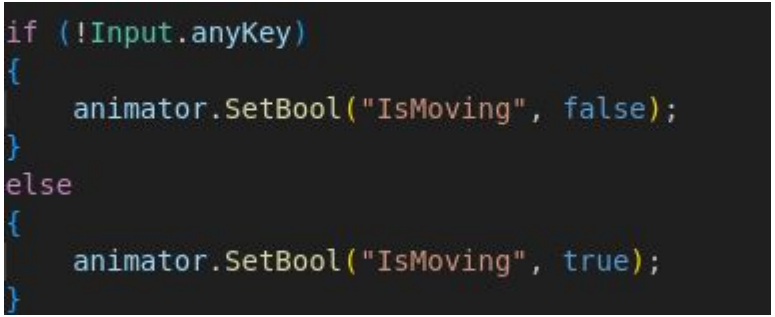
IsMoving	false

Task 4: Animation - Update Script



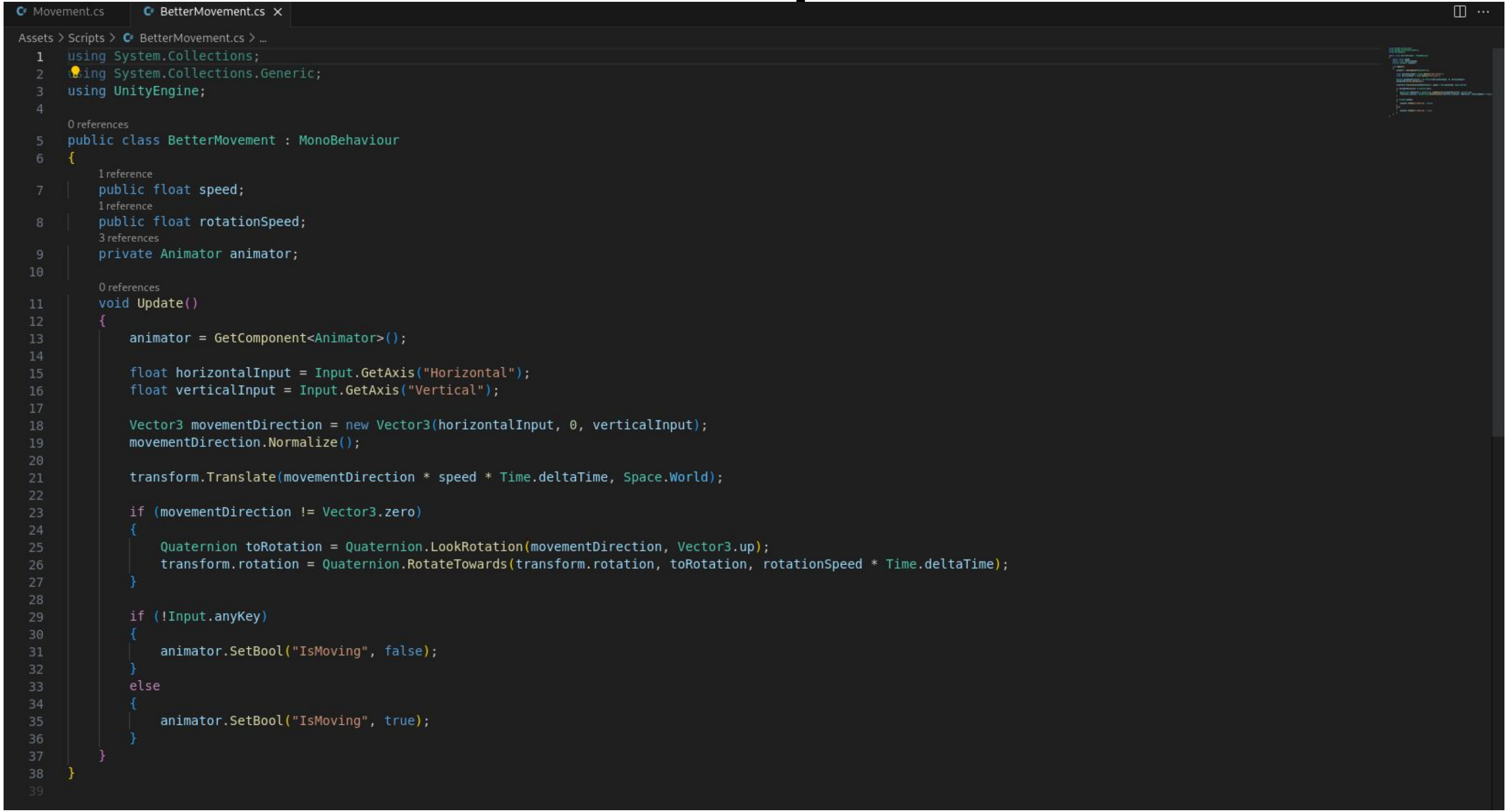
```
4 references
public float moveSpeed;
3 references
private Animator animator;

0 references
void Start()
{
    animator = GetComponent<Animator>();
}
```

In Update:



```
if (!Input.anyKey)
{
    animator.SetBool("IsMoving", false);
}
else
{
    animator.SetBool("IsMoving", true);
}
```

Task 4: Animation - Better Script with rotation



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BetterMovement : MonoBehaviour
{
    public float speed;
    public float rotationSpeed;
    private Animator animator;

    void Update()
    {
        animator = GetComponent<Animator>();

        float horizontalInput = Input.GetAxis("Horizontal");
        float verticalInput = Input.GetAxis("Vertical");

        Vector3 movementDirection = new Vector3(horizontalInput, 0, verticalInput);
        movementDirection.Normalize();

        transform.Translate(movementDirection * speed * Time.deltaTime, Space.World);

        if (movementDirection != Vector3.zero)
        {
            Quaternion toRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
            transform.rotation = Quaternion.RotateTowards(transform.rotation, toRotation, rotationSpeed * Time.deltaTime);
        }

        if (!Input.anyKey)
        {
            animator.SetBool("IsMoving", false);
        }
        else
        {
            animator.SetBool("IsMoving", true);
        }
    }
}
```