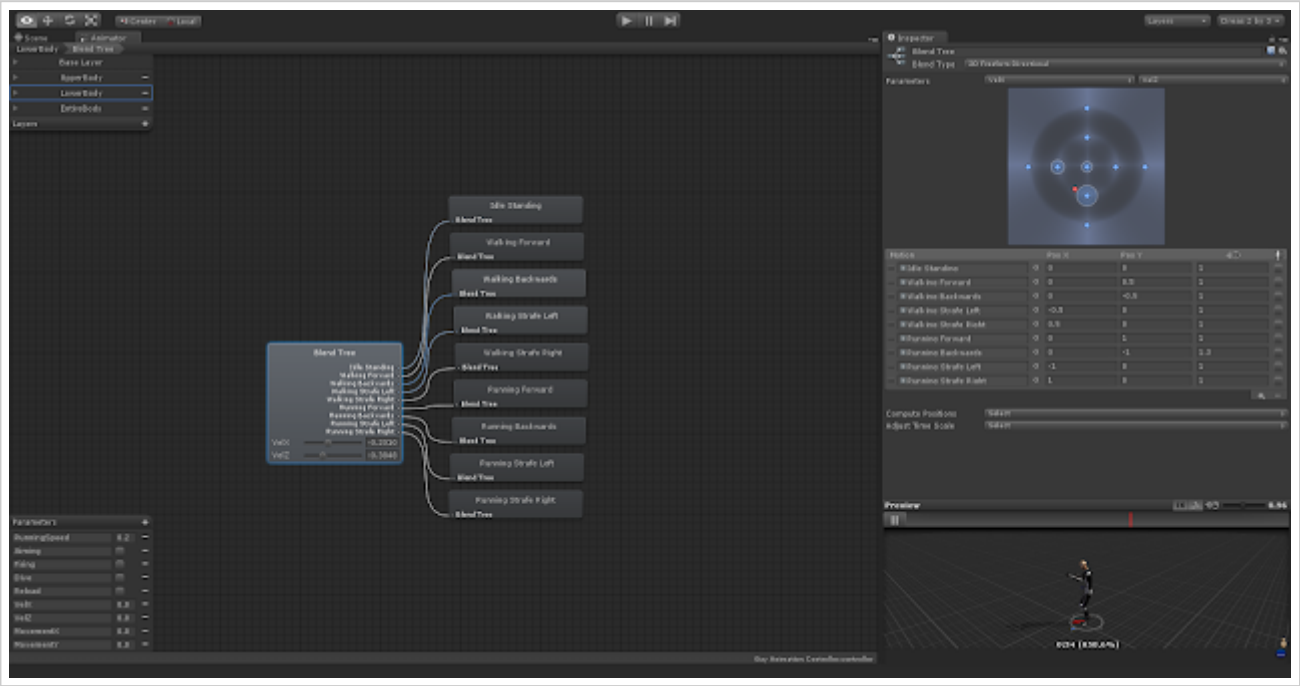
With a gamepad, I'm moving the character with the left stick and rotating with the right stick.

I am using a total of 9 animations for the blend tree. Four for each axis and the middle one for idle:

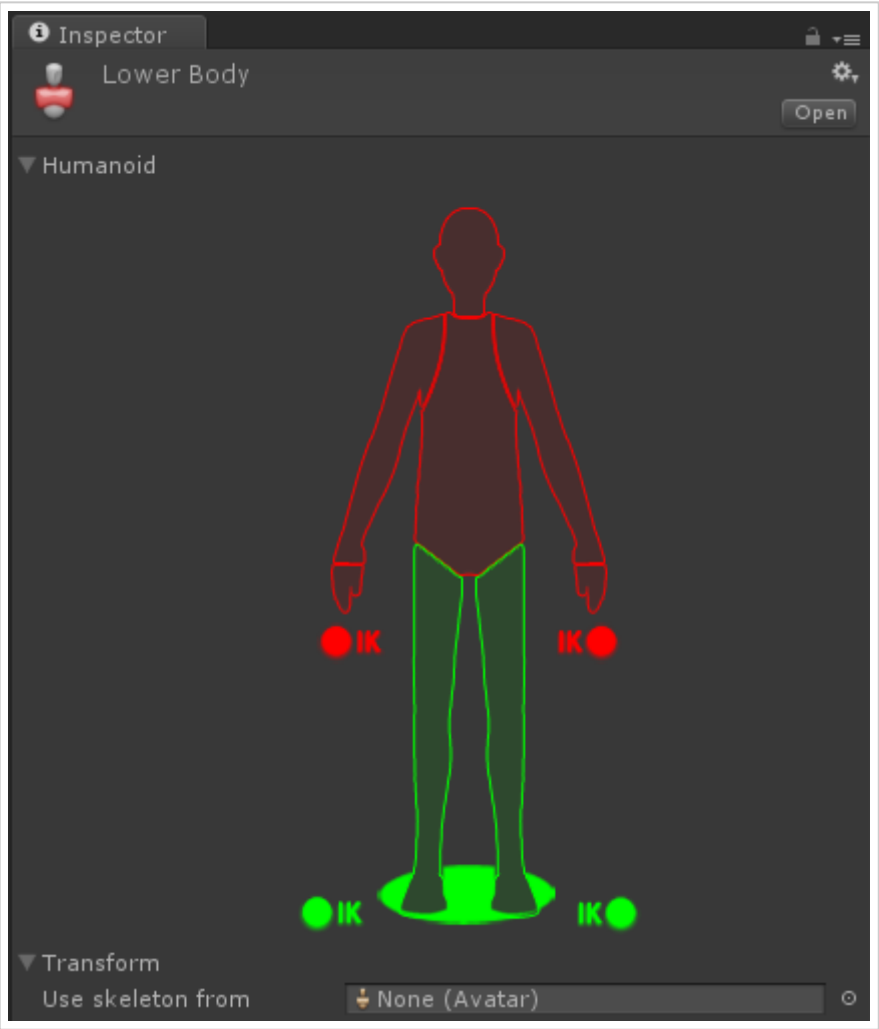
1. Idle
2. Forward Walk
3. Backward Walk
4. Left Strafe Walk
5. Right Strafe Walk
6. Forward Run
7. Backward Run
8. Left Strafe Run
9. Right Strafe Run

Here's a closer look at the blend tree and its motions:



This locomotive blend tree exists on an `Animator Layer`called LowerBody which has has an `Avatar Mask` applied with only the legs selected (including IK). By having layers representing different sections of the body, I can combine upper-body animations with lower-body animations. In my case, the character can walk/run and aim at the same time using a single animation for AIMING (which exists on the upper-body layer).

This is the Avatar Mask I use on the Lower-Body layer on which the locomotion blend tree exists:



I am controlling the blend tree with two parameters representing a direction vector: `VelX` and `VelZ`.

This direction vector determines the locomotive state of the character i.e. which direction his legs should be moving.

- If (VelX, VelZ) is `(0, 1)`, the character runs forward in a straight line.
- If (VelX, VelZ) is `(-1, 0.5)`, the character walks forward while strafing to the left.
- If (VelX, VelZ) is `(0.5, -1)`, the character runs backwards while slightly strafing to the right
- etc...

This is the gist of the code which both a) sets the (VelX, VelZ) direction vector [Left Stick] i.e. LOCOMOTION and b) rotates the model to aim [Right Stick] i.e ROTATION:

```
1  private Vector3 lastLeftStickInputAxis;  // stores the axis input from the left stick between frames
2
3  private void Update()
4  {
5      /* START LOCOMOTION */
6
7      // Get the axis from the left stick (a Vector2 with the left stick's direction)
8      var leftStickInputAxis = inputManager.LeftAxis;
9
10     // Get the angle between the the direction the model is facing and the input axis vector
11     var a = SignedAngle(new Vector3(leftStickInputAxis.x, 0, leftStickInputAxis.y), model.transform.fo
12
13     // Normalize the angle
14
```

```
    if (a < 0)
    {
        a *= -1;
    }
    else
    {
        a = 360 - a;
    }

    // Take into consideration the angle of the camera
    a += Camera.main.transform.eulerAngles.y;

    var aRad = Mathf.Deg2Rad*a; // degrees to radians

    // If there is some form of input, calculate the new axis relative to the rotation of the model
    if (leftStickInputAxis.x != 0 || leftStickInputAxis.y != 0)
    {
        leftStickInputAxis = new Vector2(Mathf.Sin(aRad), Mathf.Cos(aRad));
    }

    float xVelocity = 0f, yVelocity = 0f;
    float smoothTime = 0.05f;

    // Interpolate between the input axis from the last frame and the new input axis we calculated
    leftStickInputAxis = new Vector2(Mathf.SmoothDamp(lastLeftStickInputAxis.x, leftStickInputAxis.x,

    // Update the Animator with our values so that the blend tree updates
    animator.SetFloat("VelX", leftStickInputAxis.x);
    animator.SetFloat("VelZ", leftStickInputAxis.y);

    lastLeftStickInputAxis = leftStickInputAxis;

    /* END LOCOMOTION */



    /* START ROTATION */

    // Get the axis from the right stick (a Vector2 with the right stick's direction)
    var rightStickInputAxis = inputManager.RightAxis;
    if (rightStickInputAxis.x != 0 || rightStickInputAxis.y != 0)
    {
        float angle2 = 0;
        if (rightStickInputAxis.x != 0 || rightStickInputAxis.y != 0)
        {
            angle2 = Mathf.Atan2(rightStickInputAxis.x, rightStickInputAxis.y)*Mathf.Rad2Deg;
            if (angle2 < 0)
            {
                angle2 = 360 + angle2;
            }
        }

        // Calculate the new rotation for the model and apply it
        var rotationTo = Quaternion.Euler(0, angle2 + Camera.main.transform.eulerAngles.y, 0);
        model.transform.rotation = Quaternion.Slerp(model.transform.rotation, rotationTo, Time.deltaTi
    }

    /* END ROTATION */
}

private float SignedAngle(Vector3 a, Vector3 b)
{
    return Vector3.Angle(a, b) * Mathf.Sign(Vector3.Cross(a, b).y);
}
```

The [soldier model] and animations are from Mixamo.