Procedural walk animation Documentation v1.1

Procedural walk animation includes two main scripts to achieve realistic look of the animation.

- Procedural Animation
- Procedural Body Controller

The Starter Assets packages are compatible with **Unity 2019 LTS, 2020 LTS, 2021 LTS and 2022.1**.

The procedural walk animation requires Animation Rigging to work.

To run demo scene, you need to install **Cinemachiene** and **Input system.**

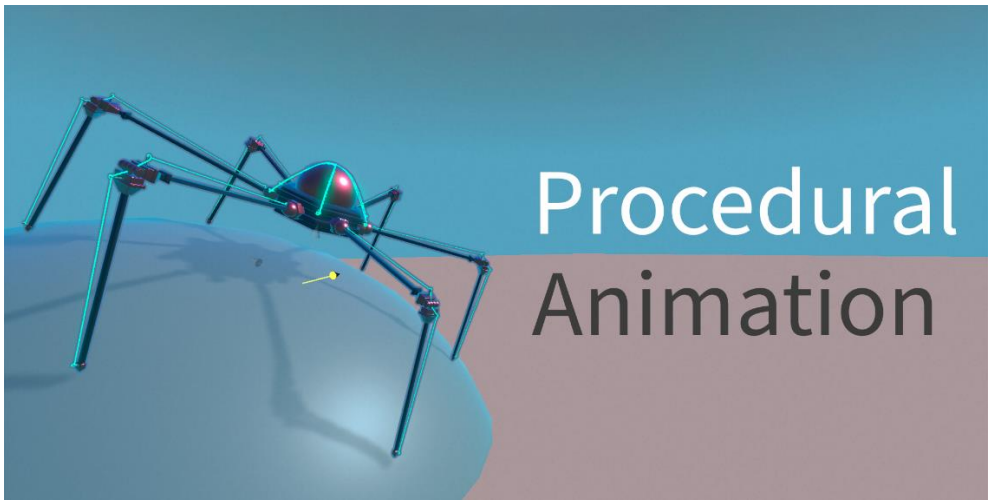


Table of contents:

Important note on package dependencies

To run demo scene, you need to install **Cinemachiene** and **Input system.**

To use the procedural animation and procedural body controller only you need to Install animation rigging package or Fast IK from the asset store. I recommend you to use animation rigging because it’s much easier to set up.

Please accept the Input System pop-up and Editor restart to successfully install the New Input System. If you accidentally decline, you can install the New Input System through the Package Manager.

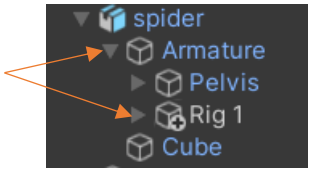
Demo Scene

in Assets/ Procedural Walk Animation/Scenes/Demo you will find two demo scenes. Here you can see procedural animation with unity third person controller or with RTS movement in a simple playground environment.

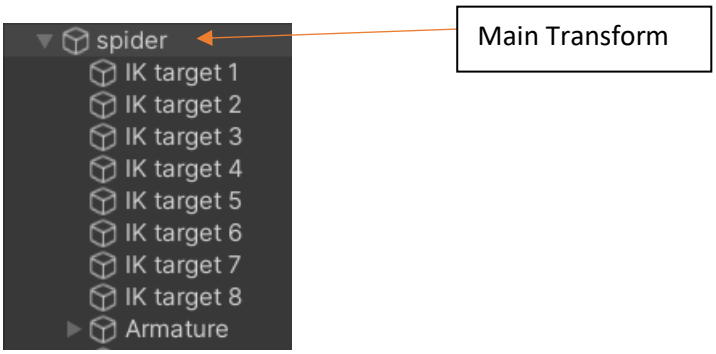
Procedural Animation

If you want to add procedural animation to your model, you need to complete a simple setup.

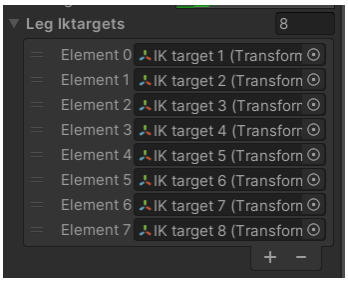
- Import your rigged model in unity.
- Set up the IK using animation rigging package or Fast IK. **It is very important to add rig builder component on the armature of your model!** (You can find a full video tutorial at my YouTube channel https://www.youtube.com/channel/UCNs5UkNwgnBU_R0B0dpNFJw). **It is very important to reset Rig transform if you are using animation rigging!**



- Make all target IK points children of the main Character Transform and add procedural animation component to it.



- Add procedural body controller to the armature. **Armarure should be parent of all bones!**
- Select all the target ponts and drag them to the Leg Ik targets.



- Add a Ground layer and add it to every walkable surface then set the layer mask to ground.

Now script will create a default leg positions at the start method.

```
defaultLegPositions[i] = legIktargets[i].localPosition;
```

By default every second Legs will move to that position + ofsset.

```
for (int i = 0; i < nbLegs; ++i)
{
    //move legs when everytime time reaches limit
    footTimings[i] += Time.deltaTime * cycleSpeed * cycleSpeedMultiplyer;

    if(footTimings[i] >= cycleLimit)
    {

        footTimings[i] = 0;

        indexTomove = i;
        SetUp(i);
    }
}
```

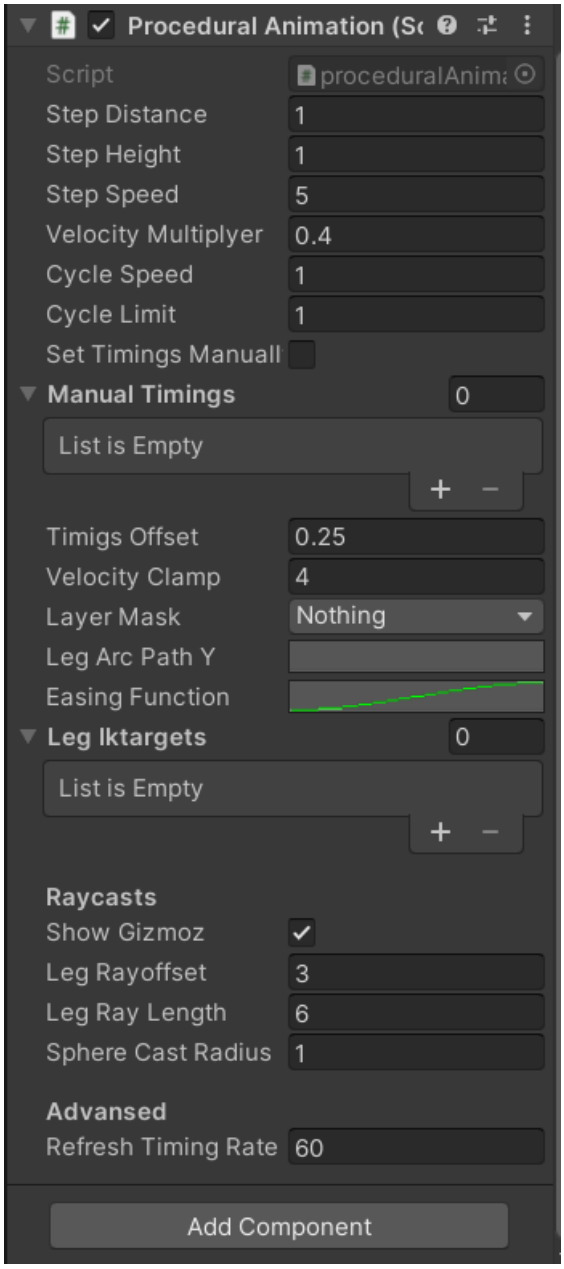
Every leg Makes step when time.deltaTime reaches cycle limit. You can ofsset this cycle for every leg so they don't moeve at the same time.

SphereCast is used to find target step point

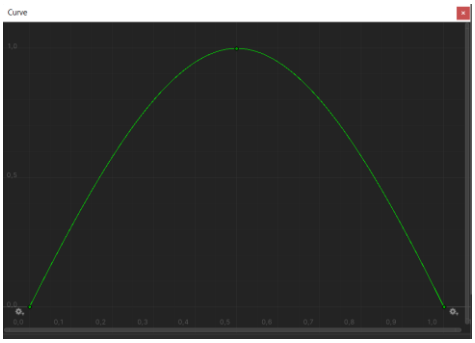
```
// finding target step point based on body velocity
Vector3 v = transform.TransformPoint(
    defaultLegPositions[index]) +
    velocity.normalized * Mathf.Clamp(velocity.magnitude, 0, velocityClamp * clampDevider) * velocityMultiplier;
targetStepPosition[index] = TargetPoint.FitToTheGround(v, layerMask, legRayoffset, legRayLength, sphereCastRadius);
```

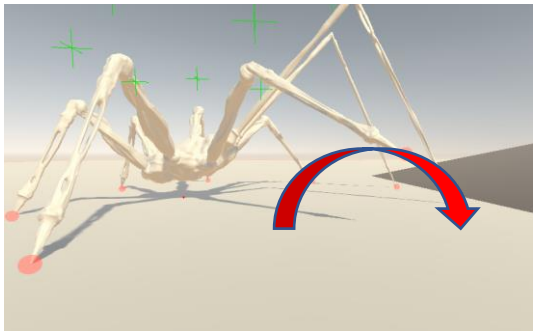
```
3 references
public static Vector3 FitToTheGround(Vector3 origin, LayerMask layerMask, float yOffset, float rayLength, float sphereCastRadius)
{
    RaycastHit hit;

    if(Physics.Raycast(origin + Vector3.up * yOffset, -Vector3.up, out hit, rayLength, layerMask))
    {
        return hit.point;
    }
    else if(Physics.SphereCast(origin + Vector3.up * yOffset, sphereCastRadius, -Vector3.up, out hit, rayLength, layerMask))
    {
        return hit.point;
    }
    else
    {
        return origin;
    }
}
```

Procedural animation script

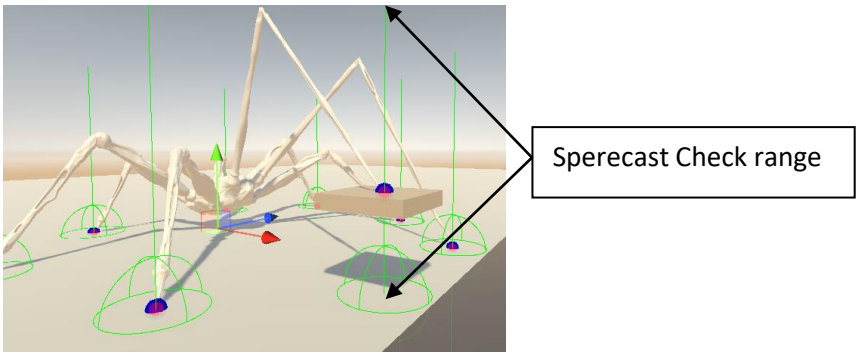


- Step distance is used to calculate step height. When character makes a short step there is no need to rase foot all the way up so if the current step distance is less then this step distance value step height will be lover then usual.
- Step Height and Step Speed it pretty straight forward.
- Velocity multiplier used to make step wider when moving on high speed (if you toggle the show gizmoz below and move your model around you could clearly see what this does. The blue spheres represent the target step points and will move further ahead if you increase velocity multiplier)
- In the update method there is cycle that calls make step function every second the duration of the cycle is controlled by Cycle limit (if you set it to two your character will make step every two seconds) Cycle speed controls the speed of this cycle.
- If you want only one leg to move at a time then set Timings offset as one divided by the number of legs. For example: if your character has four legs you need to set this as ¼ = 0.25. The script will offset the cycle of every leg by 0.25 seconds.
- If you want some legs to move together enable the Set Timings Manually. And add as many timings as your model has legs. The first Manual Timing is relative to the first leg in the leg IK targets array etc. For example: if your character has four legs and you want two left legs move first and two right to move second you need to set timings to [0.5, 0.5, 0, 0]. That means that first two legs will move and only 0.5 second later the second two will move.
- Velocity clamp limits the step distance while moving on high speed.
- Set every walkable surface in layer mask.
- Leg Arc is path of the leg on Y axis during the step (should start at 0 with value of 0 and end at one with value of 0).





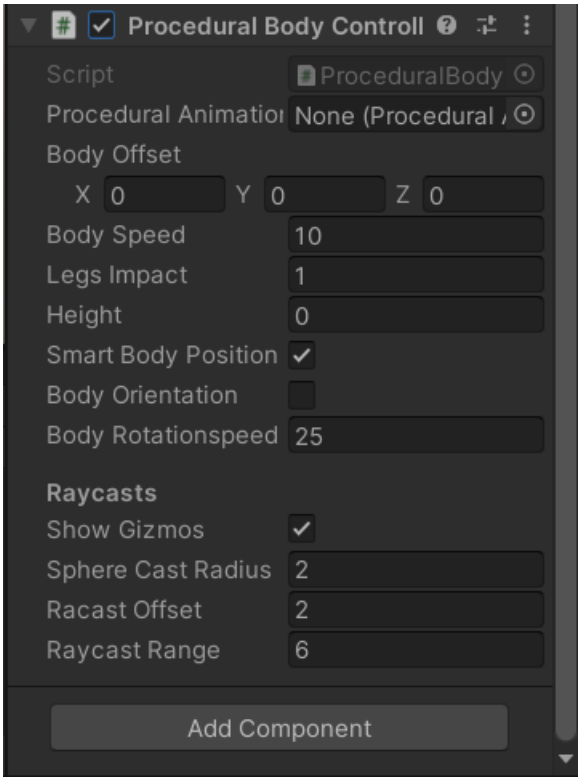
- Drag all IK targets to the leg IK targets field.
- All Raycast settings is a Ground check for every leg. Leg will make a step to the hit point of Spherecast. You can clearly see this range in Playmode if you toggle show gizmoz.



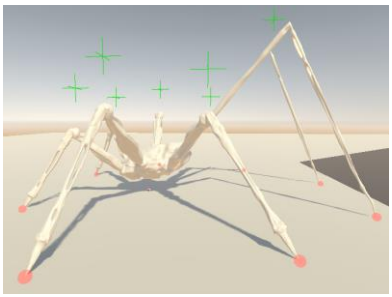
- Refresh Timings rate updates timings and sets it to default value to make sure every leg is making step at the right time. If your character moves slowly, you can set it as some big value like 100 so it updates only every 100 seconds but if not, you need to lower this value. For example: fast pink robot in demo scene has this value set as 10.

Procedural body controller

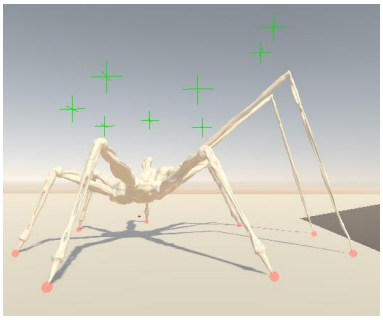
Add this script to the armature of your model.



- Drag the main object with procedural animation component to the first field.
- Body speed controls how fast body moves to the new position.
- Legs impact controls the body behavior during the step. The body raises up when character makes step.
- Height controls the body height.
  Value of zero:



  Value of one:



- **Smart body position.** I highly recommend to use this function. It makes everything more realistic and brings life to the animation by calculating body position based on average leg positions. Sometimes you need to offset body using this function. Use body offset vector to do this.
- If body orientation is enabled body will rotate toward the ground.
- Raycast settings do the same thing as in the procedural animation script but this time with the body.

Step effects script

Every time any leg reaches target position procedural animation script Invokes event. The Step effect script is a simple event listener that plays audio and instantiates the particle prefab.

```
private void ProceduralAnimation_OnStepFinished(object sender, Vector3 LegPosition)
{
    if (audioSource != null)
    {
        audioSource.PlayOneShot(stepSVX[UnityEngine.Random.Range(0, stepSVX.Length - 1)]);
    }

    if(stepVFX != null)
    {
        Instantiate(stepVFX, LegPosition, Quaternion.identity);
    }
}
```

Technical documentation

For more details about how the different Scripts work, please refer to the comments within the Scripts themselves.