Creature IK

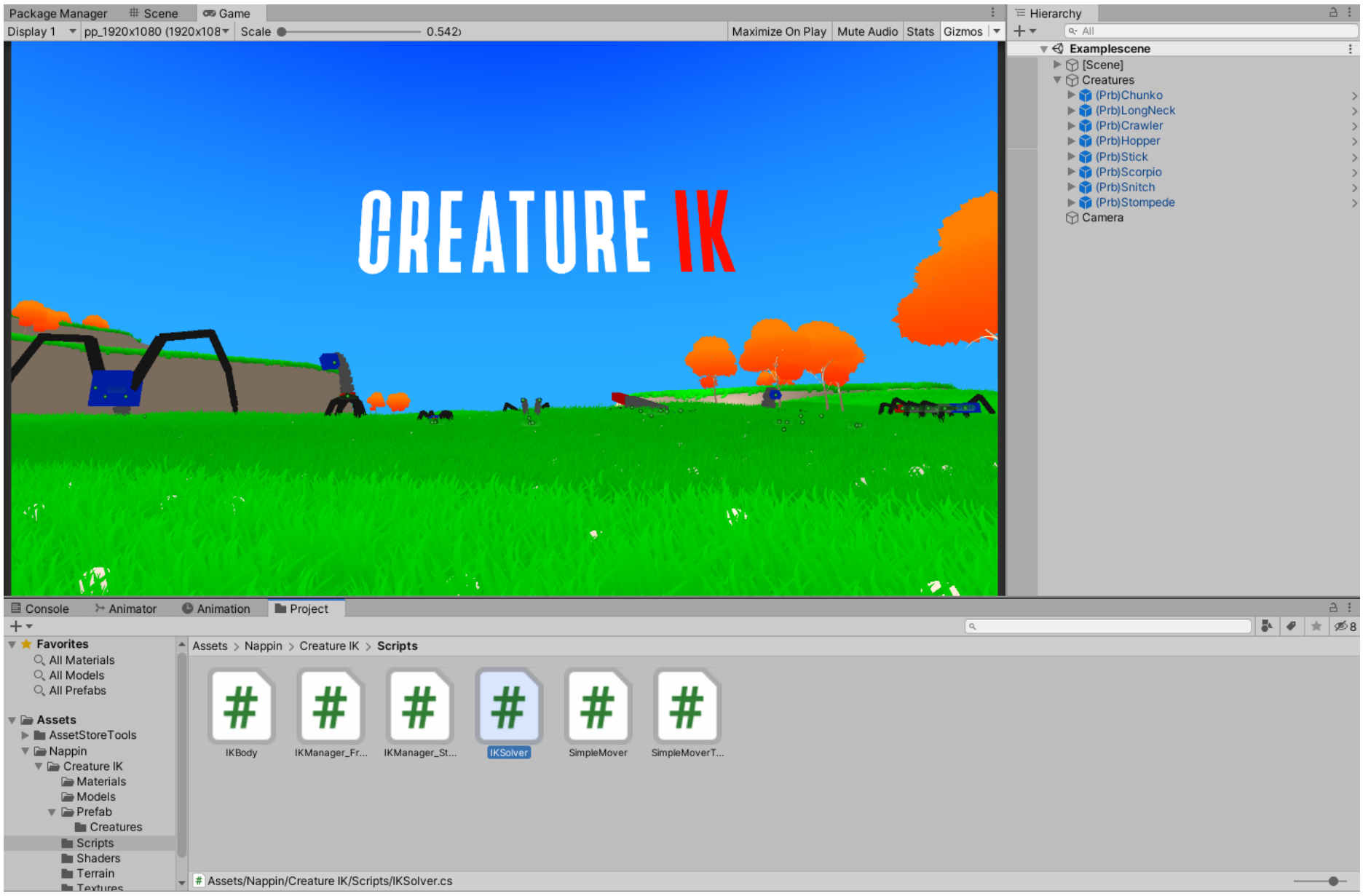
*Support email: **nappin.1bit@gmail.com***

Table of Contents

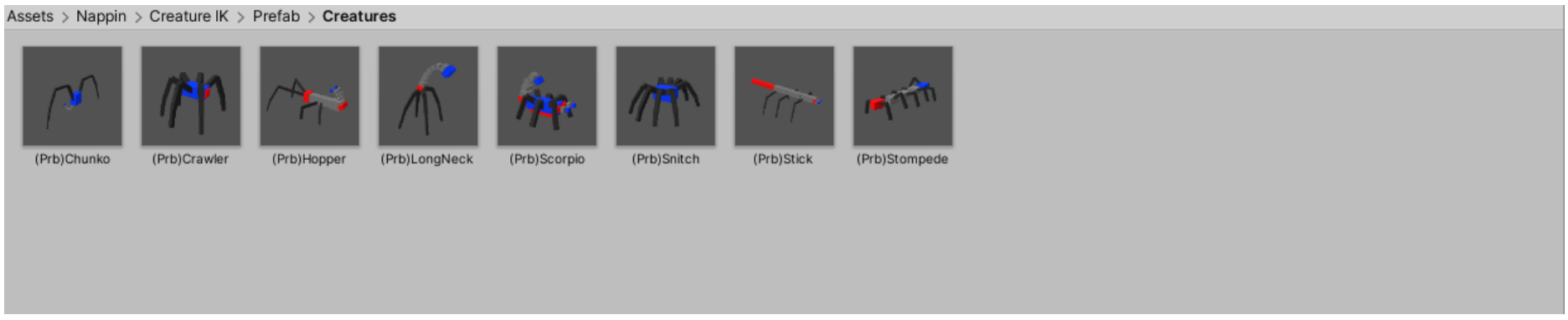
Asset content

The asset contains multiple prefabs of creatures that use the IK solver, from centipedes to two legged creatures with tails. These prefabs allow you to explore and understand how hierarchies and scripts operate. The core content of the asset can be found in the *Prefabs* folder and in the *Scripts* folder.

In the *Scripts* folder you can find the core solver **IKSolver** and the three scripts that allow you to create the creatures: **IKBody**, **IKManager_Freeform (anchored and unanchored)** and **IKManager_Step**. In this folder you can also find two simple mover scripts that allow you to test the creatures movement without having to move them manually in playmode.



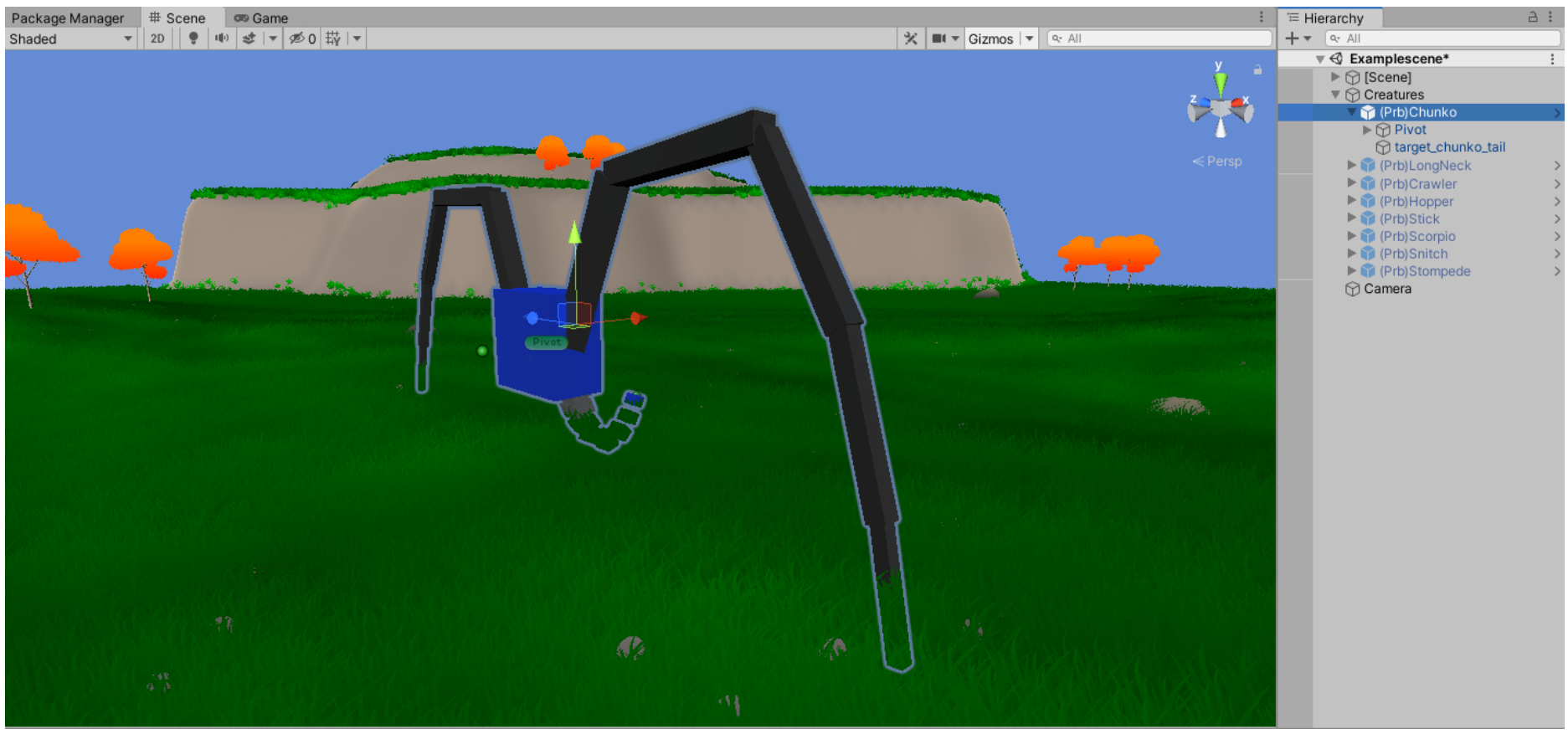
In the prefab folder you can find multiple sample creatures and the following prefabs: **(Prb)Body** (a standard body with the **IKBody** script), **(Prb)Leg** (a standard leg with the **IKManager_Step** script) and **(Prb)SolverAnchored / (Prb)SolverUnanchored** (vertebrae with the **IKManager_Freeform** script).



N.B. It's highly recommended to experiment with the prefabs to learn how the creatures operate and to learn how to setup their hierarchy

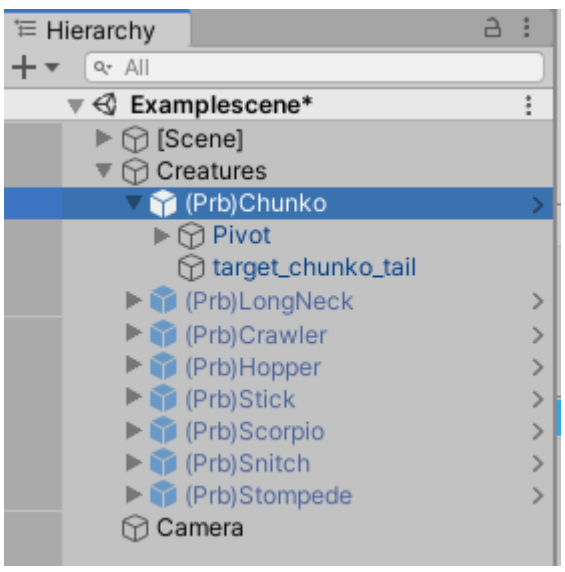
Single body creatures

There are 2 main types of creatures: the ones that have a single body and the one that have a complex structure made of multiple units (like centipedes). In this chapter I'll focus on single body creatures, its components and its structure.



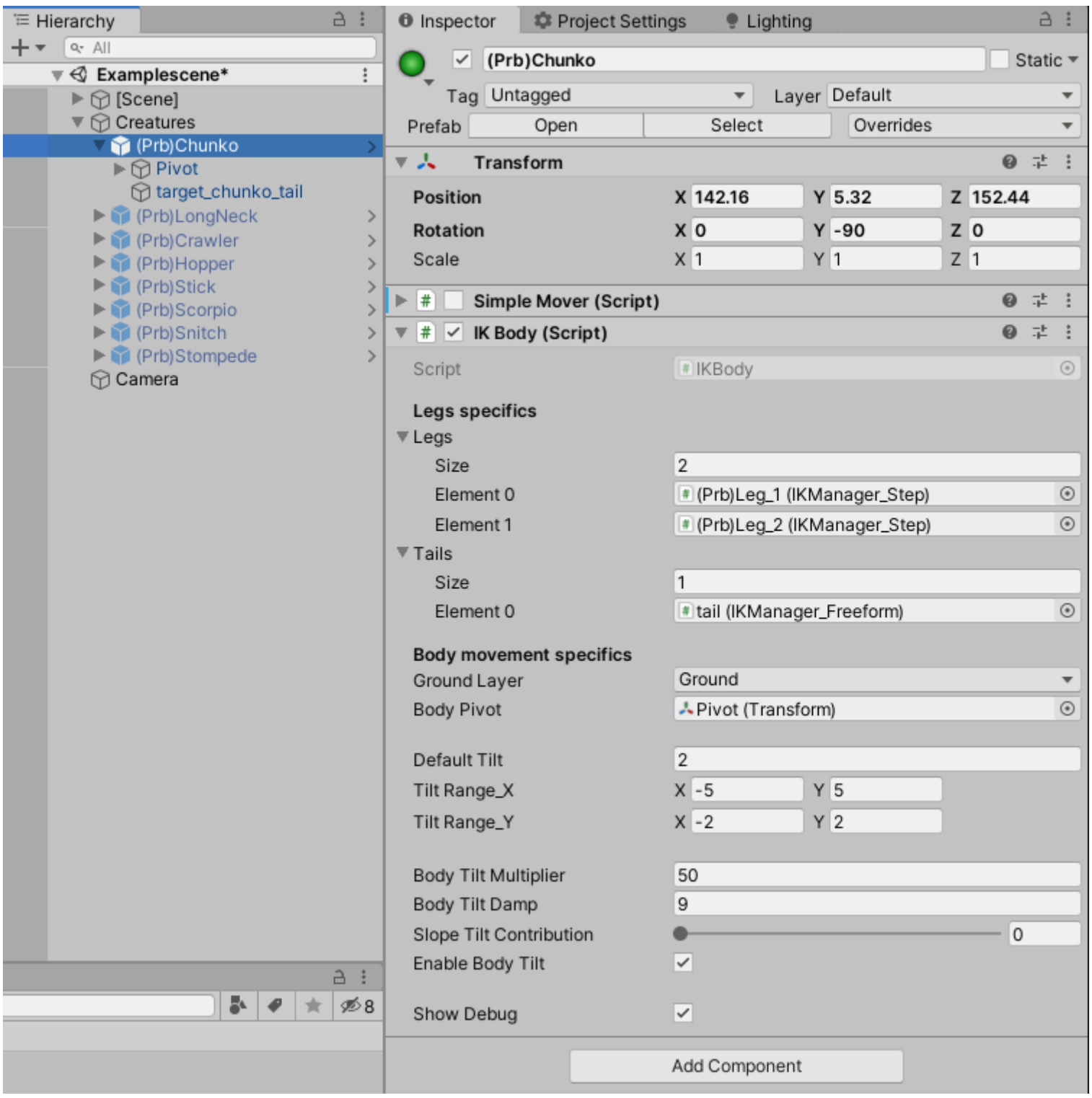
Root

The **Root** GameObject of the creature contains the script **IKBody**. Every creature requires this component because it determines when to update the leg position and how to handle the creature motion. Here we also have the creature Pivot and one or multiple targets used for tail, vertebrae, antennae etc… (if the creature has them).



IKBody

Let's see what the **IKBody** does and let's understand its exposed public variables. You can see a short description of what each variable does in the editor as well simply hovering on it.



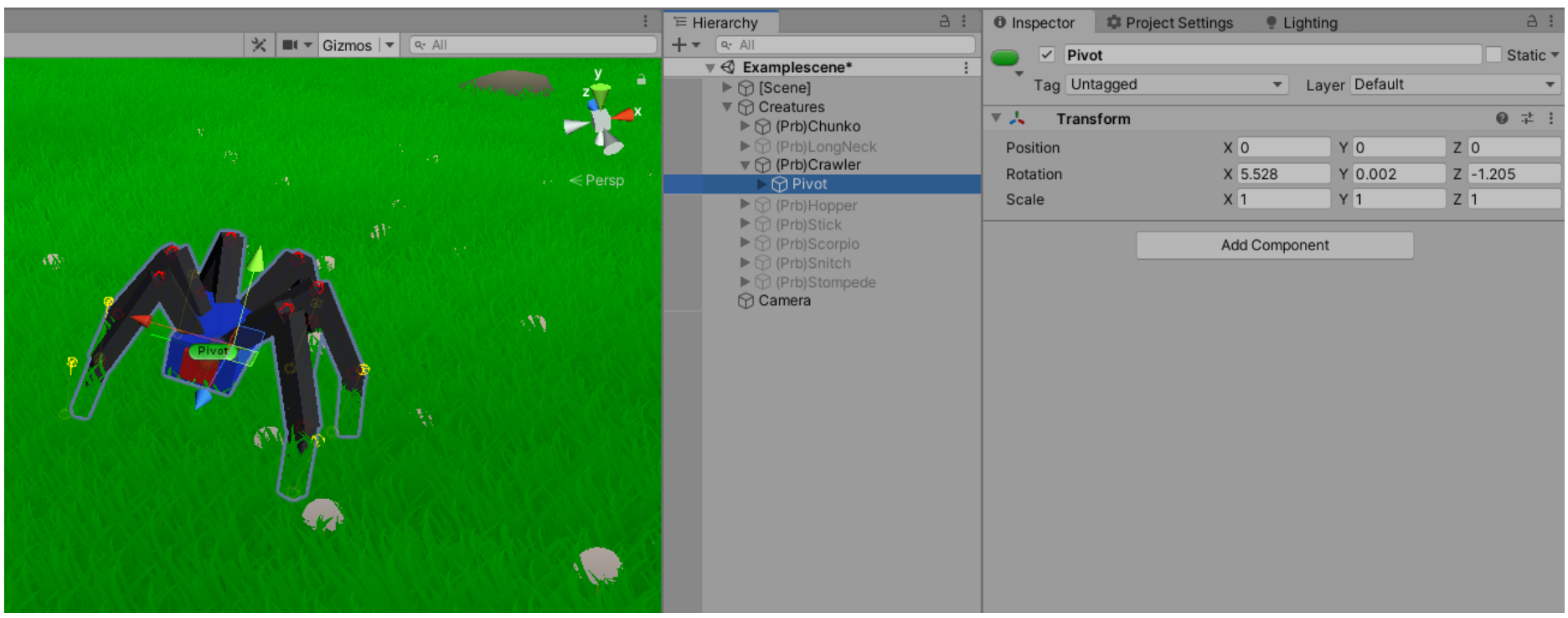
In the *LegsSpecific* section, its variables are :

- ***Legs***: an array containing the creature legs referenced by their script **IKManager_Step**. There needs to be at least a single leg reference for the script to work
- ***Tails:*** an array containing creature tail, vertebrae, antennae etc… referenced by their script **IKManager_Freeform**. This variable can be left empty.

In the *BodyMovementSpecific* section its variables are :

- ***GroundLayer***: the layer that is considered ground
- ***BodyPivot***: reference to the **Pivot** of the creature. Why should the **Pivot** ("*Pivot*" in the image below) be separated from the creature **Root** (*green circle* in the image below)?

They are separated so that the creature can keep a fixed distance from the target and to allow the creature to rotate around a point rather than on itself. If you prefer to rotate the creature on its local Y axis and have the distance from the target equal to zero you can set the **Pivot** coordinate to (0,0,0) so that it overlaps with the **Root**.



- ***Default tilt***: the basic tilt angle applied to the player pivot
- ***Tilt range X:*** the angle range of the creature **Pivot** on the X axis
- ***Tilt range Y:*** the angle range of the creature **Pivot** on the Y axis
- ***Body tilt multiplier:*** how much the creature motion and momentum impact its **Pivot** tilt

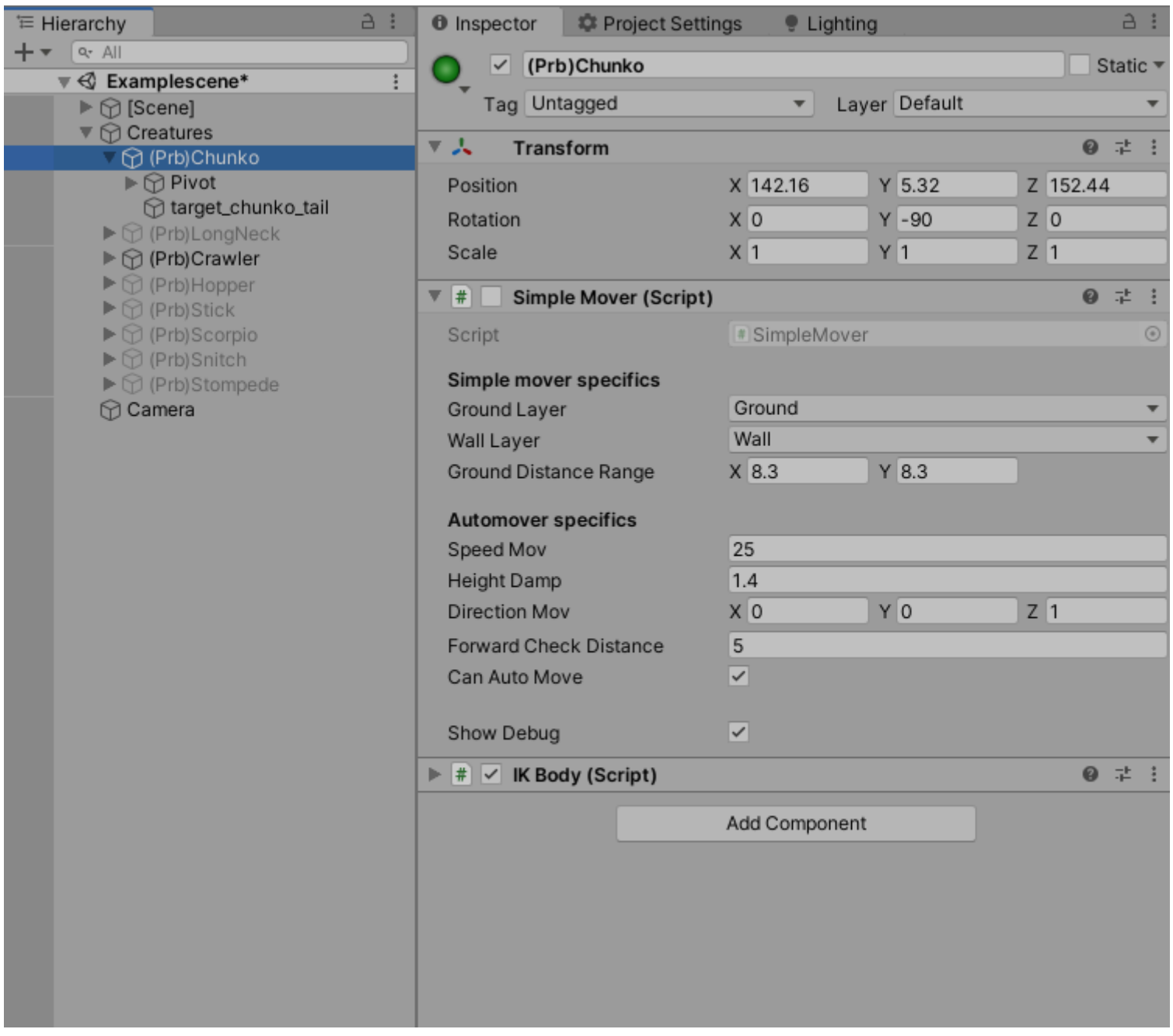
- ***Body tilt damp:*** how smooth is the **Pivot** tilt
- ***Slope tilt contribution:*** if this value is set to 1 the creature tries to have its tilt equal to the ground normal angle, if the value is set to 0 the ground angle is ignored.
- ***Enable body tilt***
- ***Debug***: allows you to see the debug infos and gizmos

P.S. It's recommended to experiment with the body tilt range to avoid unnatural motion on really steep slopes or ramps

Simple Mover

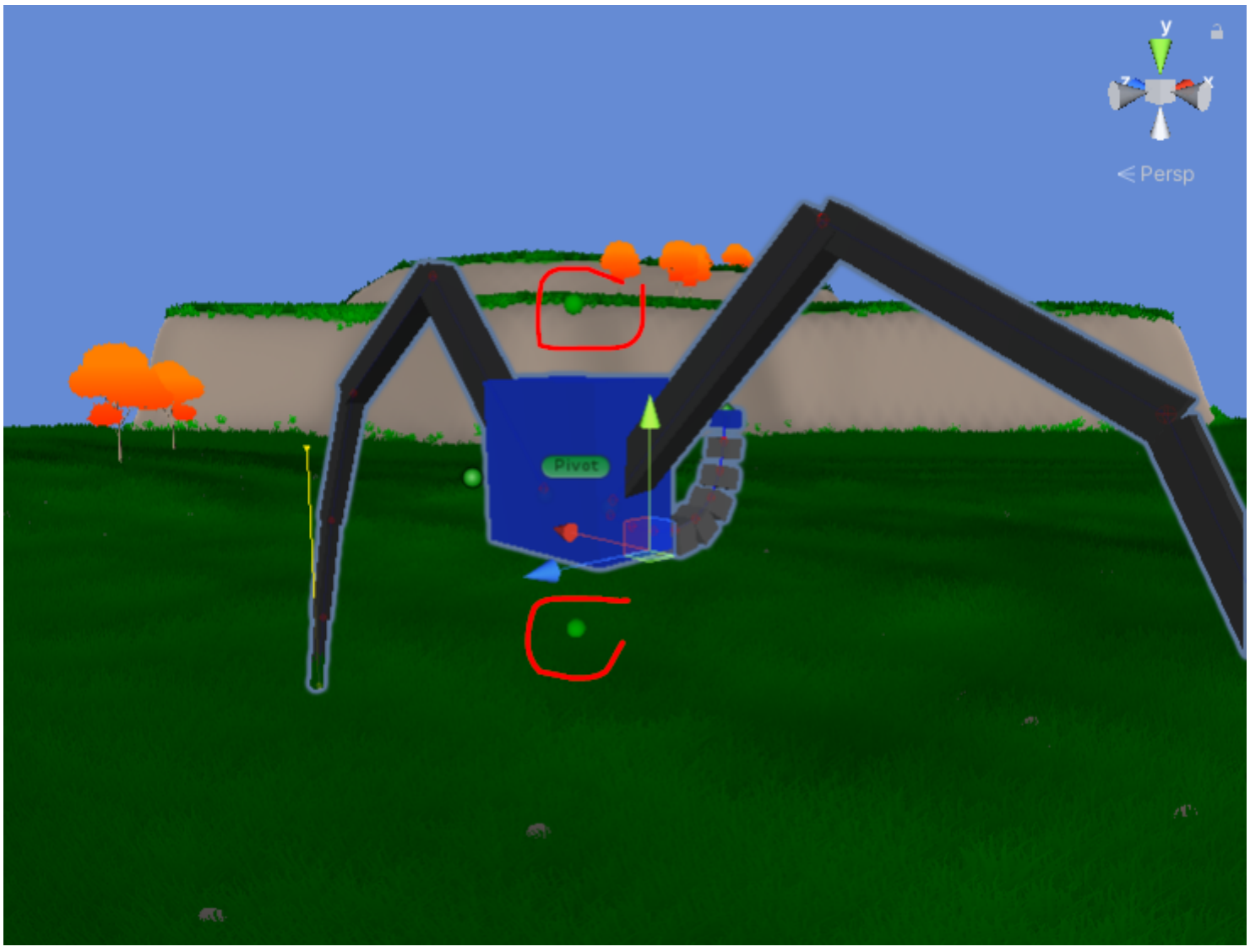
In the **Root** we also have a **SimpleMover**. The purpose of the script is to test the creature motion in the example scene. Of course in your project you would use this script and change the direction according to your need or implement your own mover solution. Regardless, what the script does is quite simple: it moves the **Root**.

Let's explore the script public variables but remember that you can also see a short description of what each variable does in the editor simply hovering on it.



In the *SimpleMoverSpecifics* section its variables are :

- ***GroundLayer***: the layer that is considered ground
- ***WallLayer***: the layer that is considered wall
- ***GroundDistanceRange:*** the range distance from the ground of the creature pivot. In the screenshot above the creature is locked at 8.3 distance from the ground. If you set a range bigger then 0 you can see it with this *green gizmos*



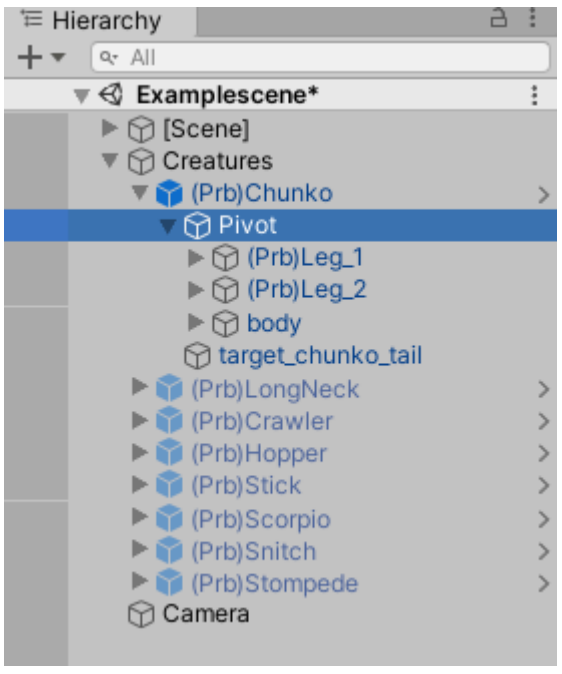
In the *AutomoverSpecifics* section its variables are :

- ***SpeedMov***: the speed of the creature
- ***HeightDamp***: the smoothness used to move the pivot y position
- ***DirectionMov:*** the direction in which the creature is moving
- ***ForwardCheckDistance***: distance used by the script to check if there is a wall in front of it or a drop / empty area
- ***CanAutoMove:*** allows you to enable / disable the motion

- ***Debug***: allows you to see the debug infos and gizmos

Pivot

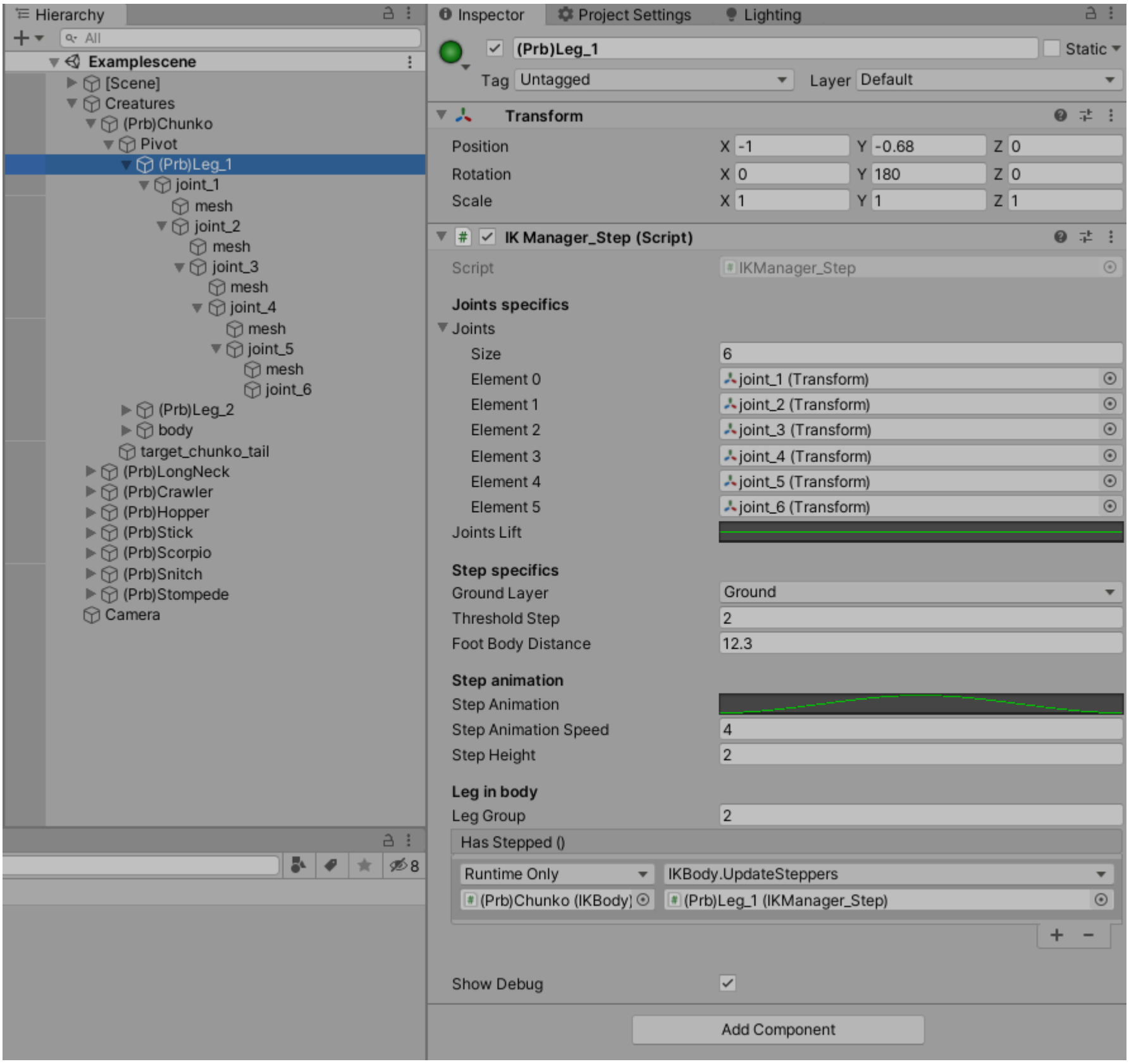
The **Pivot** contains the creature legs and its body. The Legs are inside the **Pivot** so that they can move and tilt with the body. If you are interested in keeping the legs position locked and independent from the body tilt you can simply extract them from the **Pivot** and place them directly in the creature **Root**.



Legs

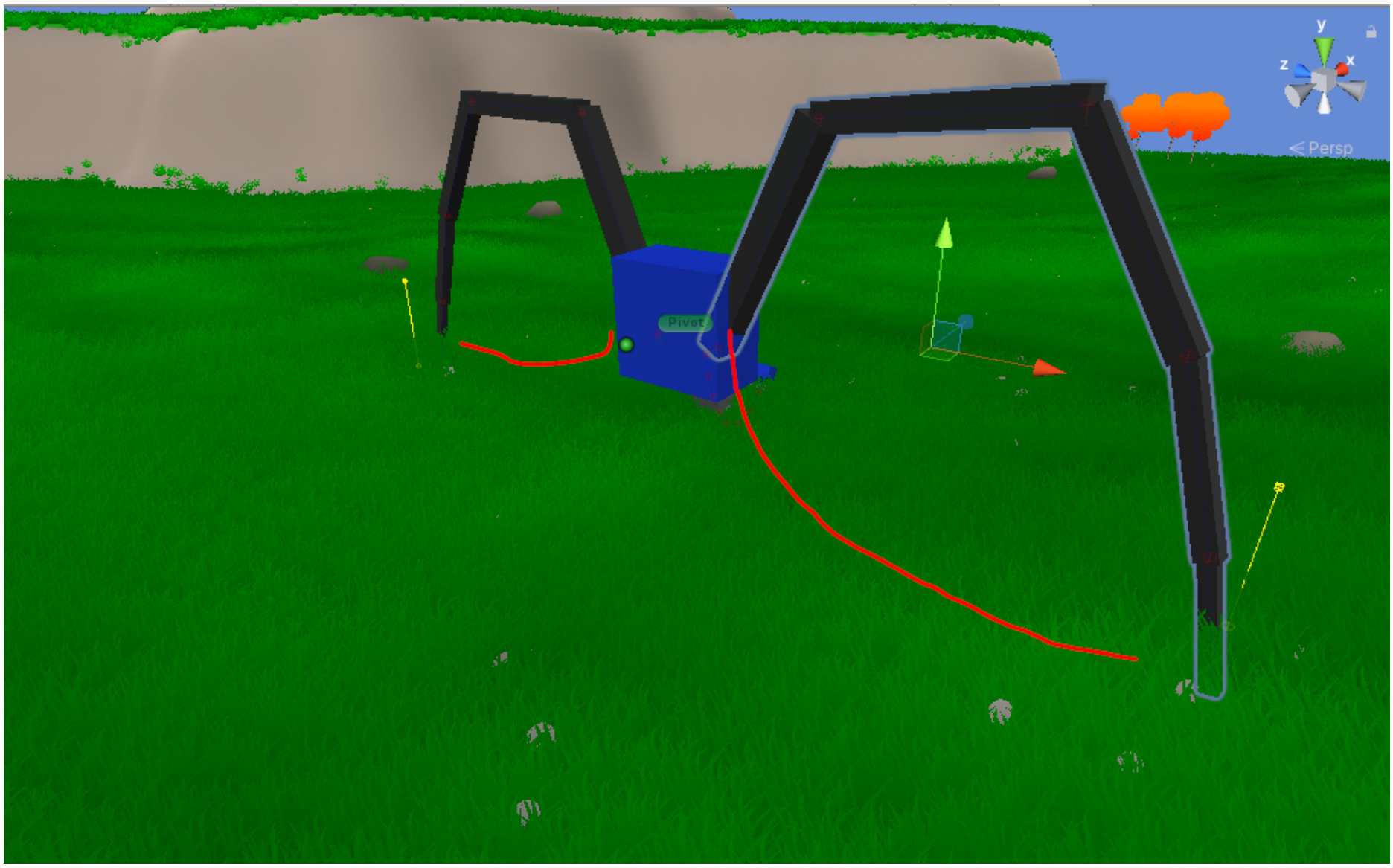
The **Leg** and it's script **IKManager_Step** are essential for the creature motion. The hierarchical structure of the **Leg** is quite simple: a nested sequence of joints and meshes. The solver supports virtually infinite joints so you can create whatever you like.

Let's explore the script public variables but remember that you can also see a short description of what each variable does in the editor simply hovering on it.

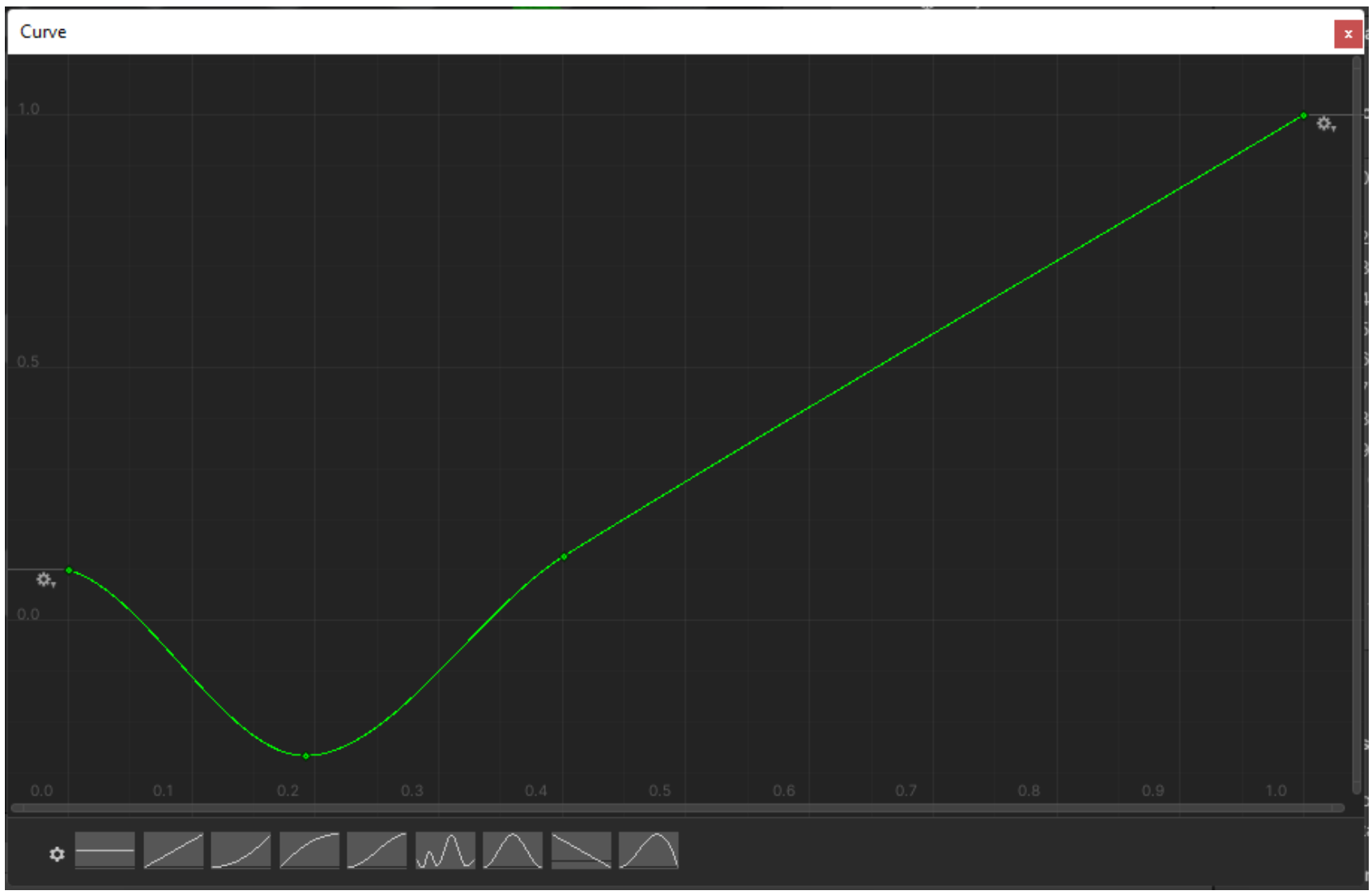


In the *JointSpecifics* section its variables are :

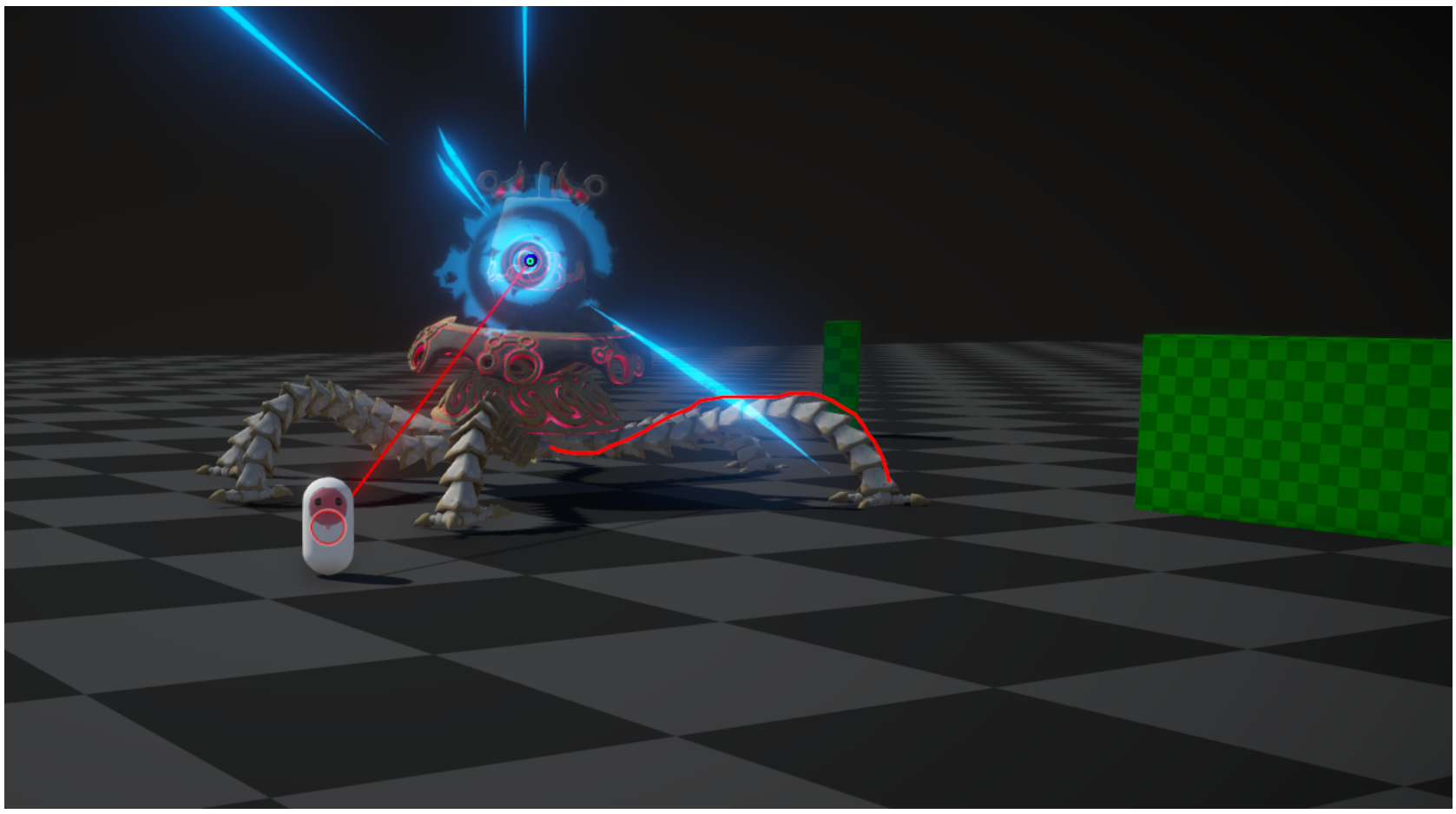
- ***Joints***: an array containing the legs joints. They have to fill the array from the hip to the foot
- ***JointsLift:*** the solver sets the joints position in the most efficient way possible but sometimes we want to force the joint position in a certain way. For example the solver might say that the most efficient way to place the joints in the pic below is according to the *red line*



The JointsLift variable allows us to apply a correction on the y axis and arrange the joints in a more organic and realistic way. For instance arranging the joint lift curve this way…



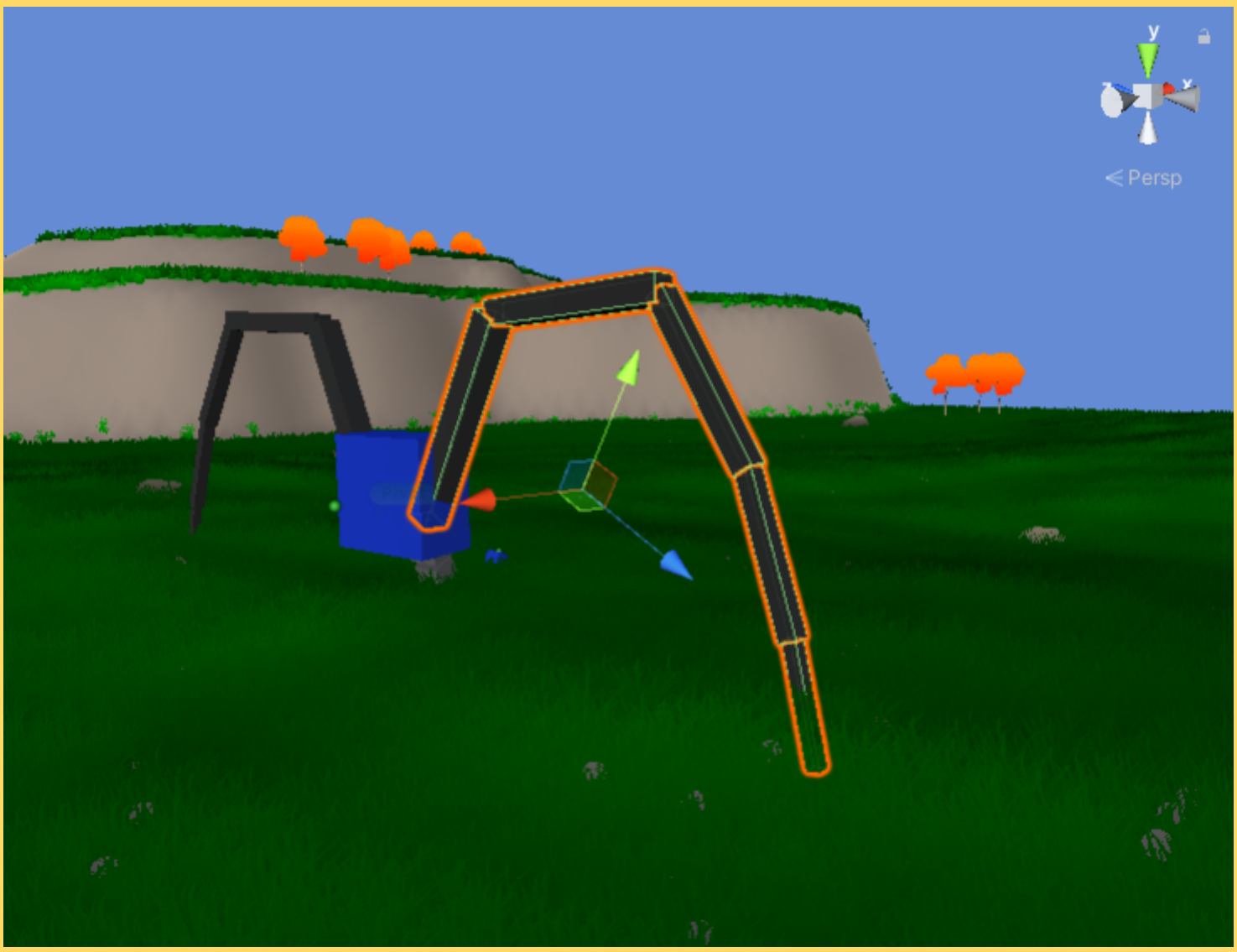
…allows you to obtain a more organic result (**FOR LEGAL REASONS** this asset is not included in the project).
If you want to learn how this specific setup was created you can check out the walkthrough here.



In the *StepSpecifics* section its variables are :

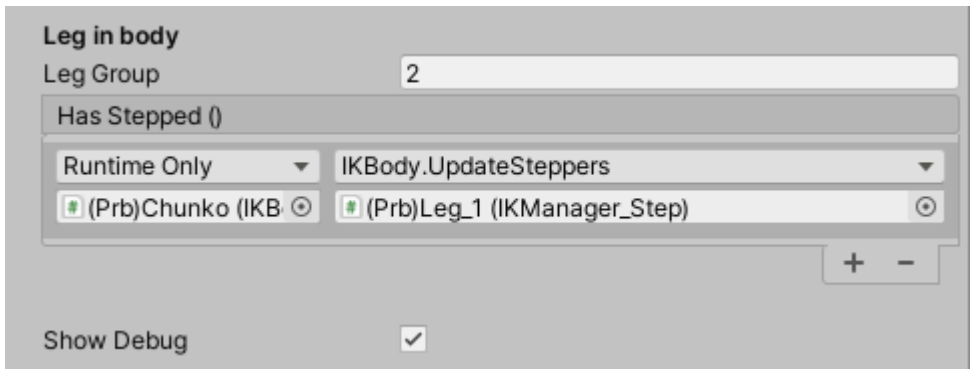
- ***GroundLayer***: the layer that is considered ground. It's required by the script to allow the creature to move the leg accordingly
- ***ThresholdStep***: when the distance between the hip and the foot reaches this value the creature attempts to take a step
- ***FootBodyDistance***: the distance on the X and Z axis of the foot from the hip

N.B. the leg meshes can have a collider component assigned to them so that they can collide and interact with the environment



Leg grouping

The following section is *critical* to integrate the leg motion into the body. If for some reason you are interested in using a leg alone without anything else you don't need to fill the variables in this section.

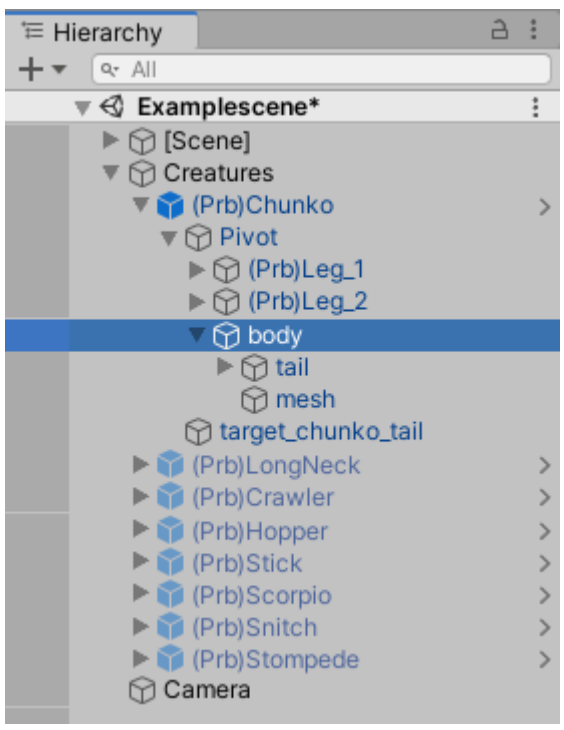


In the *LegInBody* section its variables are :

- ***LegGroup***: legs with the same value are grouped. When the body decides that it's time to take a step, different groups take the step at different times. For example if I have a creature with two legs and one leg has a **LegGroup** of 1 and the other of 2, both legs will never be off the ground at the same time.
- ***HasStepped():*** the events calls the body and tells it that this leg has accomplished a step
- ***Debug***: allows you to see the debug infos and gizmos

Body

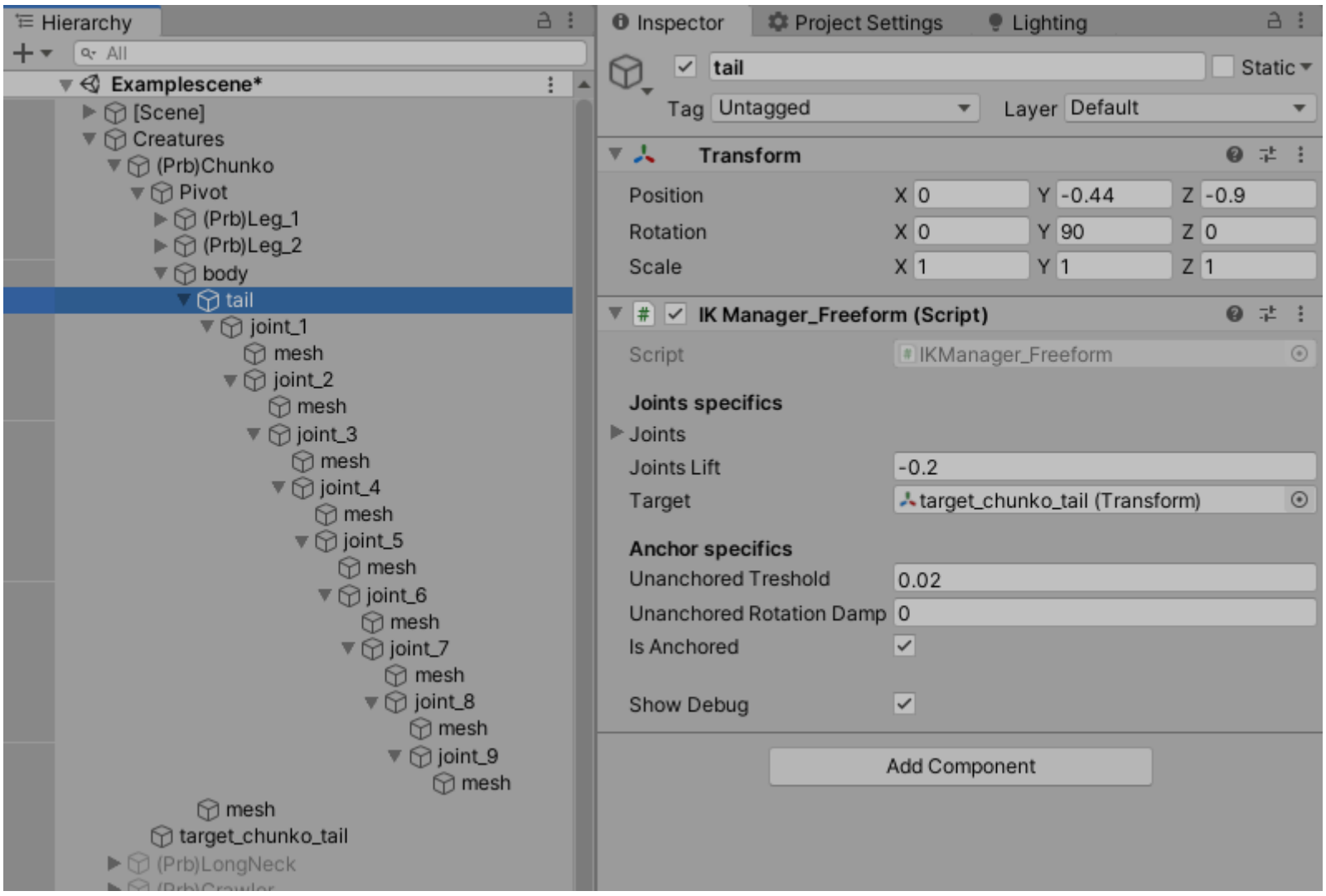
The **Body** of the creature contains the creature legs and its body. This GameObject is simply a container so let's explore its children and see how they work.



Tail

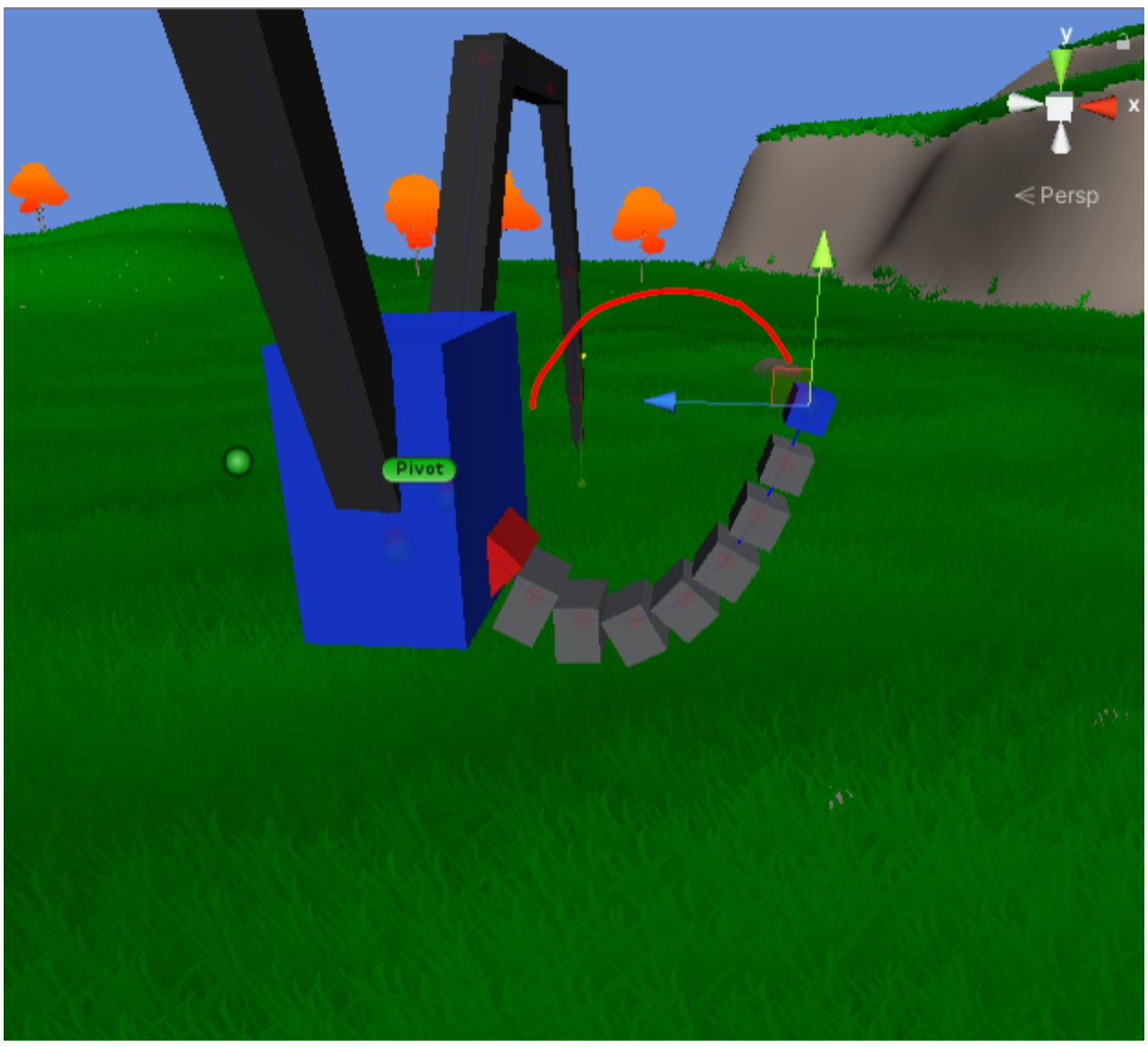
The **Tail** contains the script **IKManager_Freeform**. The **Tail** is inside the **Body** that is inside the **Pivot** so that they can move and tilt with the body. If you are interested in keeping the **Tail** position locked and independent from the body tilt you can simply extract it from the **Pivot** and place them directly in the creature **Root**.

Let's explore the script public variables but remember that you can also see a short description of what each variable does in the editor simply hovering on it.



In the *JointSpecifics* section its variables are :

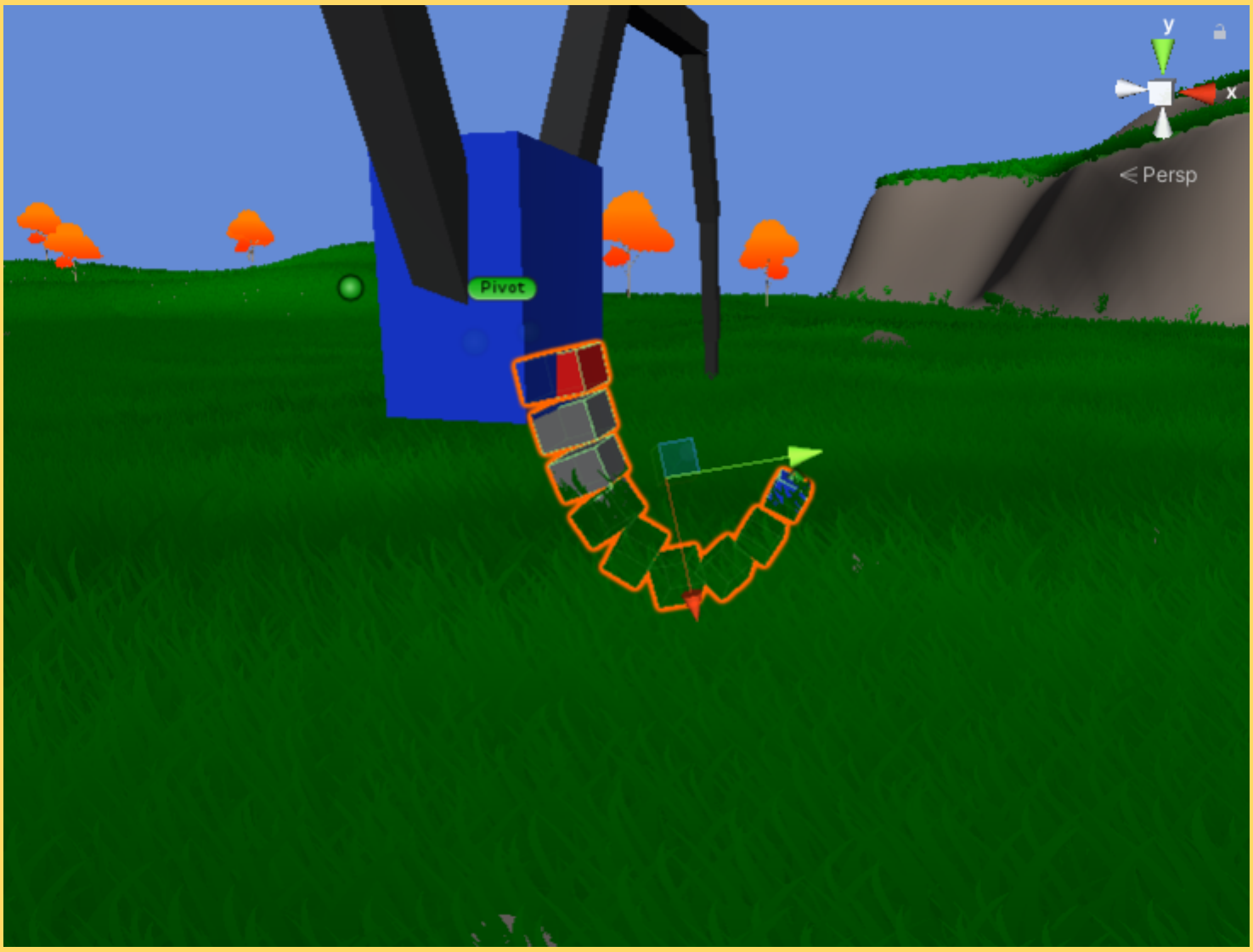
- ***Joints***: an array containing the vertebrae joints. They have to fill the array following the hierarchical structure
- ***JointsLift:*** the solver sets the joints position in the most efficient way possible but sometimes we want to force the joint position in a certain way. For example the solver might say that the most efficient way to place the joints in the pic below is according to the *red line*



The JointsLift variable allows us to apply a correction on the y axis and arrange the joints in a more organic and realistic way

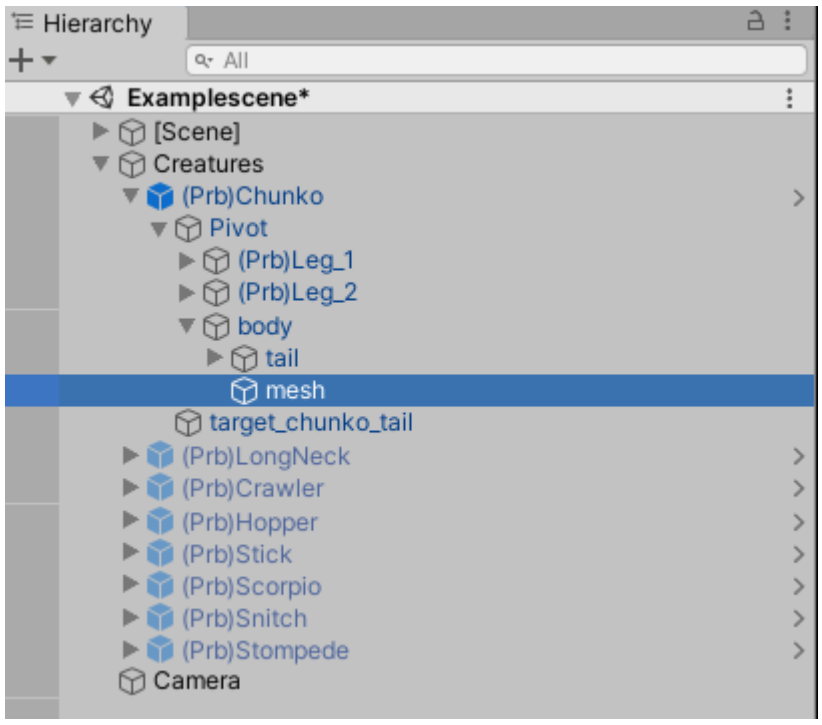
As you can see the boolean ***isAnchored*** is activated, this allows the first joint of the tail to be looked in position. When this boolean is unchecked the joints move freely without being anchored. You can see how to use the ***Unanchored*** setup and what the remaining variables do here.

N.B. the tail meshes can have a collider component assigned to them so that they can collide and interact with the environment

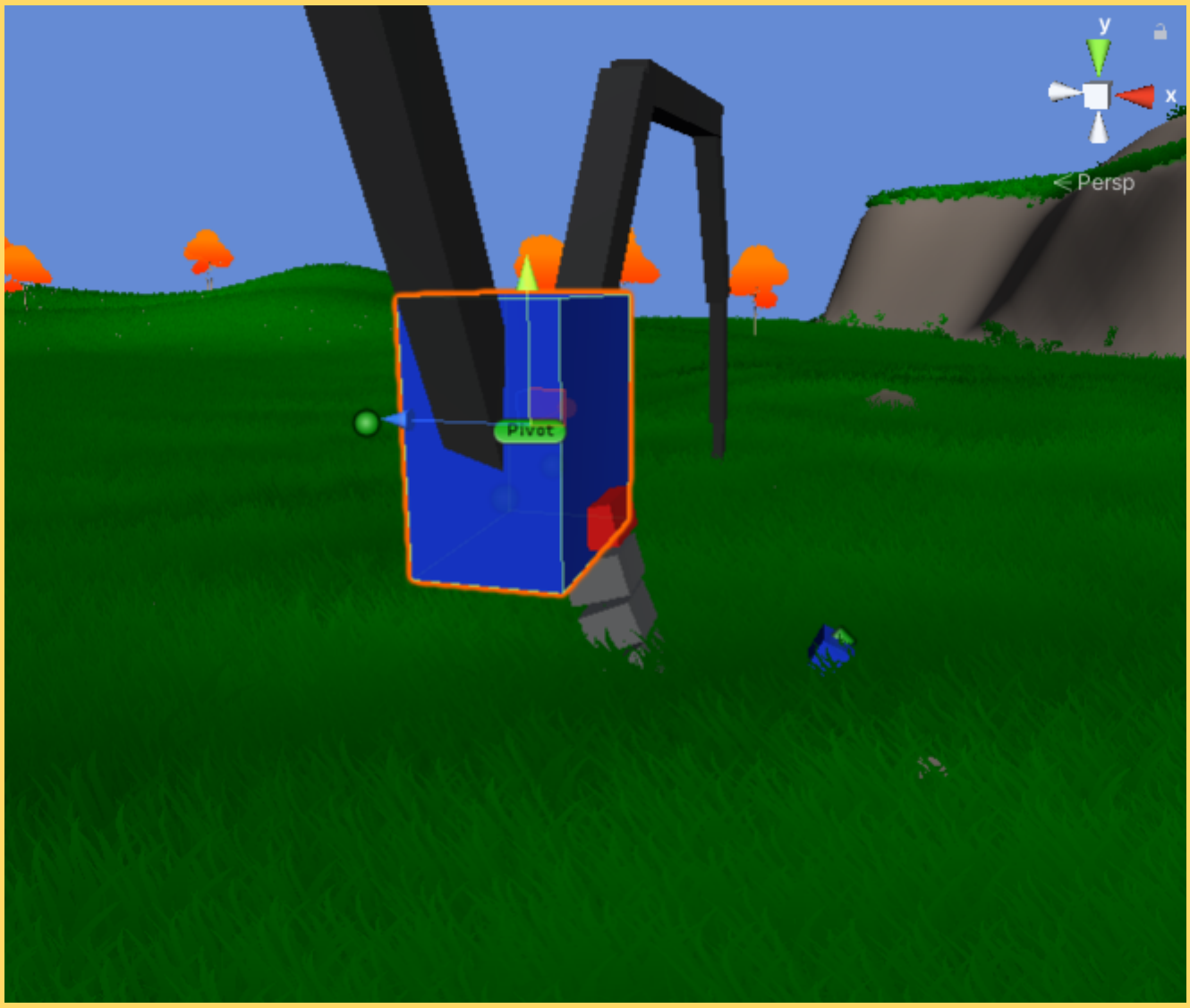


Mesh

Is a simple GameObject that can contain a single or multiple meshes.

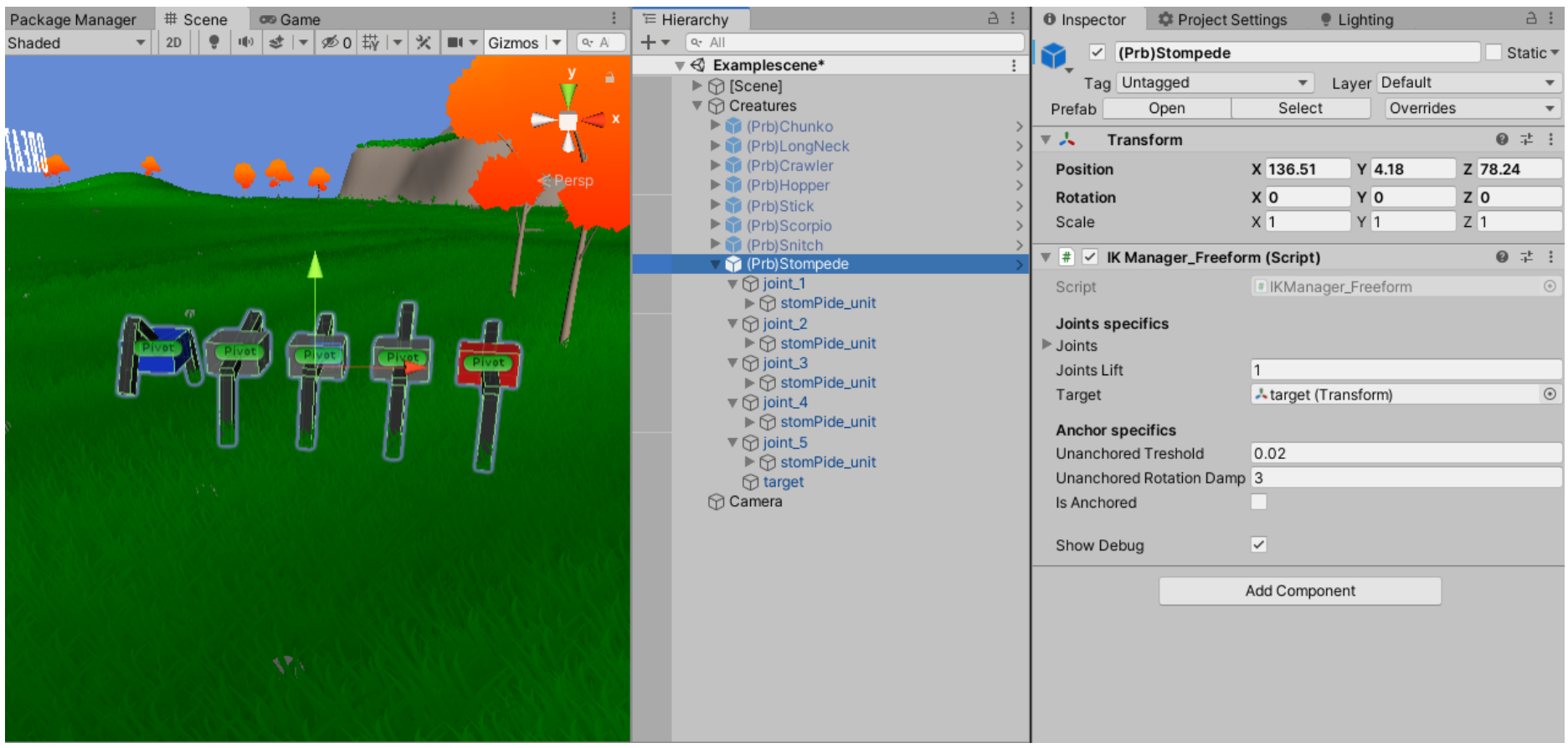


N.B. the body meshes can have a collider component assigned to them so that they can collide and interact with the environment



Multi body creature

A multibody creature is a creature made up of multiple body units (like a centipede). Let's explore the creature **(Prb)Stompede** to understand how the hierarchy works

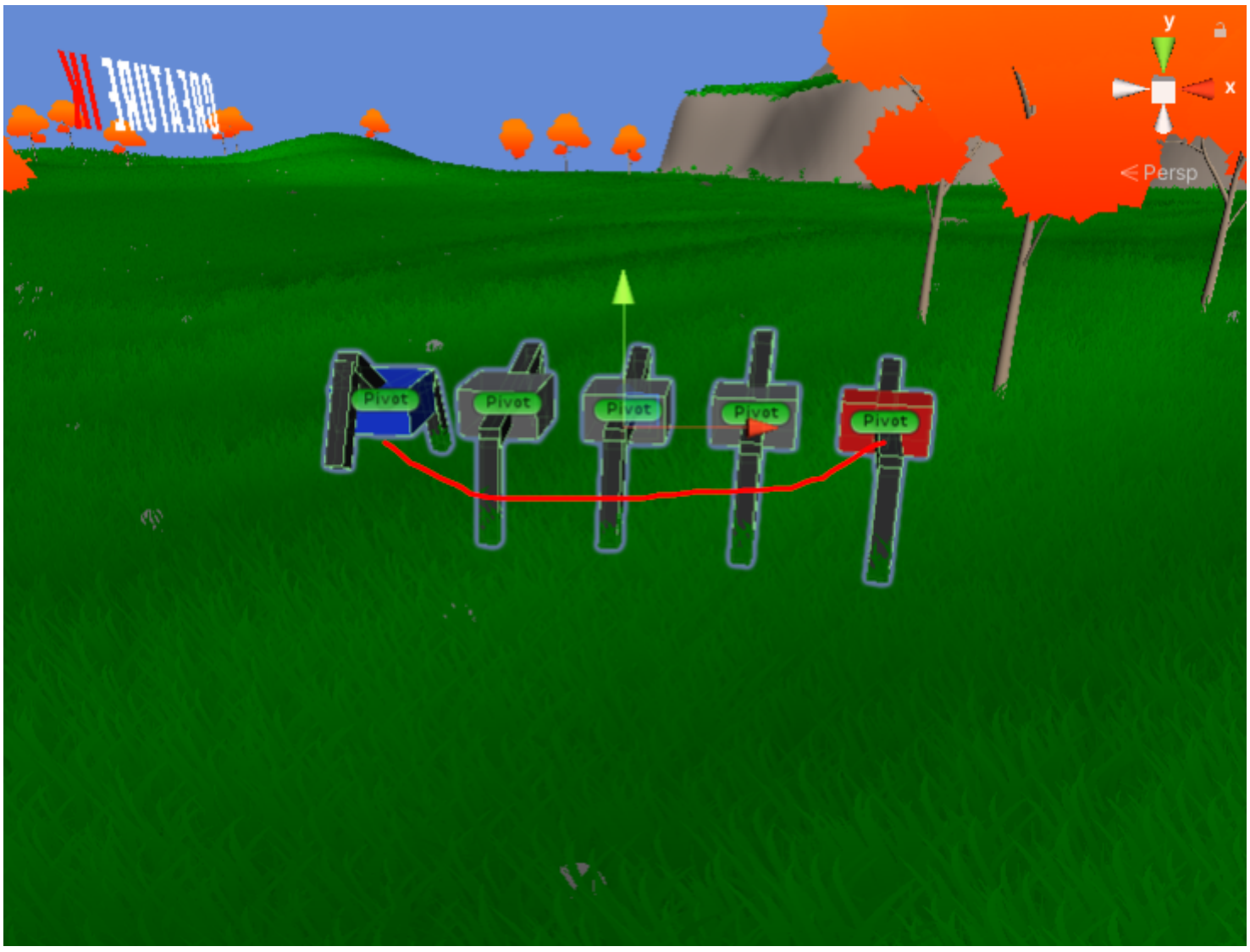


As you can see differently from *Single Body Creatures* the **Root** contains multiple sibling joints each containing a body. In the **Root** we also have a target gameobject that guides the creature motion. The script that manages the joints movements and ultimately the bodies movement is **IKManager_Freeform.**

Let's explore the script public variables but remember that you can also see a short description of what each variable does in the editor simply hovering on it.

In the *JointSpecifics* section its variables are :

- ***Joints***: an array containing the vertebrae joints. They have to fill the array following the hierarchical structure
- ***JointsLift:*** the solver sets the joints position in the most efficient way possible but sometimes we want to force the joint position in a certain way. For example the solver might say that the most efficient way to place the joints in the pic below is according to the *red line*



The JointsLift variable allows us to apply a correction on the y axis and arrange the joints in a more organic and realistic way

In the *AnchorSpecifics* section its variables are :

- ***Unanchored Threshold***: the minimum motion needed to recalculate the joints position
- ***Unanchored Rotation Damp:*** the smoothness used to rotate the joints
- ***IsAnchored***: as you can see the boolean ***isAnchored*** is disabled, this allows the first joint of the tail to be free and allow the creature joint siblings to move. You can see what happens when the solver is anchored here.
- ***Debug***: allows you to see the debug infos and gizmos

Contact

If you found this guide useful but need further help feel free to contact me at the email ***nappin.1bit@gmail.com***
P.S. A positive review of the asset would help a lot!

Cheers