Procedural Pipeline: Procedural 3D character design using *Blender*

Alejandro Valverde Mahou
Technical University of Denmark (DTU)
Copenhagen, Denmark
s212728@dtu.dk

Abstract

The objective of this study is to develop a procedural pipeline for generating fully rigged and animated 3D character models, with the ability to apply random variations to the models. Although the technologies and methods employed in this project are not entirely new, their combination has not been previously explored in the literature. The aim is to optimize the time and effort required to generate multiple creatures that exhibit similar behaviour, for use in populating 3D worlds. The whole pipeline involves the usage of *Blender* for the 3D model of the characters, as well as the rigging, and the usage of *Unity* for developing the procedural animation and the flock behaviour. This report specifically focuses on the initial phase of the pipeline, which involves designing the character using the software tool *Blender*.

1 Motivation and Goals

The motivation behind this project stems from the objective of developing a universally applicable procedural walking animation for characters with diverse designs. To achieve this goal, it was necessary to obtain a 3D model for animation purposes. Consequently, the concept of a fully automated pipeline was conceived in order to reduce the workload required for creating multiple models. The creation of a procedural animation offers many advantages, although it also poses several challenges, particularly the need for development and calibration for each animation. To address this issue, we aimed to create a singular movement animation that could be utilized for a wide range of characters. By doing so, we could minimize the manual workload required and fully automate the pipeline to produce any number of characters.

The primary goal of this report is to present the first part of the procedural pipeline, that utilizes *Blender* Geometry Nodes and Scripting, to automate the creation of fully rigged 3D models of characters. By combining these tools, we have developed a streamlined process for generating 3D models that can be used to populate 3D worlds with characters exhibiting a range of behaviours and characteristics.

To demonstrate the potential of our pipeline, we have created a prototype design resembling a robotic ant or insect. While the variation in this initial design is limited, we illustrate how our approach can be applied to more complex and diverse character designs, and how the amount of variation can be increased as needed.

By following the procedures outlined in this report, users can create 3D models with specific proportions and characteristics, as well as apply a range of movements and animations to them. This pipeline has the potential to significantly reduce the time and effort required to generate multiple 3D models for 3D world population, while allowing for a high degree of variability in the final designs.

The full project can be found in the GitHub repository: akua21/procedural-robot-modelling.

2 Methods

This section details the process of creating fully rigged 3D models of characters using *Blender* [1]. Our methodology involves utilizing the Geometry Nodes tool, which enables node manipulation instead of traditional 3D modelling techniques. However, due to the limitations of the Geometry Nodes tool, we also make use of *Blender* Scripting to define the rigging of the character.

The creation of our modular 3D model, which consists of a *Head*, a *Body Segment*, and a *Leg Segment*, was the initial step in our pipeline. Subsequently, we developed a straightforward geometry node structure that combined these three objects to create a single model. This allowed for variability in both body length and leg length, with the potential for further complexity and variation in future designs. The use of the Geometry Nodes tool can be seen in Appendix A: Blender Geometry Nodes.

Once we have a 3D model, we need to add rigging. Unfortunately, Geometry Nodes does not allow rig bones modification, and we are required to use *Blender* Scripting for this purpose. The *Blender* Python API is a very powerful tool that allows for the replacement of all the operations that are usually done by hand with code, which greatly aids in the automation of the process. Thanks to this, we were able to fully rig the procedural 3D model, and additionally, we can change the input parameters of the model from it. We use a random seed given from the user's input to randomize the character creation.

To learn more about *Blender*, its capabilities, and its Python API, we referenced two sources: "Taking Blender to the Next Level: Implement advanced workflows such as geometry

nodes, simulations, and motion tracking for Blender production pipelines" by Ruan Lotter [3] and "Blender Python API: Precision 3D Modelling and Add-on Development" by Chris Conlan [2]. These sources provided valuable information and guidance throughout the creation of our procedural character pipeline.

3 Results

3.1 Geometry Nodes

The use of Geometry Nodes in the character creation pipeline allowed for a more efficient workflow, as compared to traditional 3D modeling techniques. By utilizing node manipulation, we were able to easily create modular components of the character, which could be quickly and easily combined to generate a wide range of unique and varied characters. This resulted in a significant reduction in the amount of time and effort required to create each character, as well as an increase in the level of detail and complexity that could be achieved. Overall, the use of Geometry Nodes proved to be a highly effective tool in the creation of fully rigged 3D models of characters. In Figure 1 some examples of the output of the geometry nodes structure is shown.

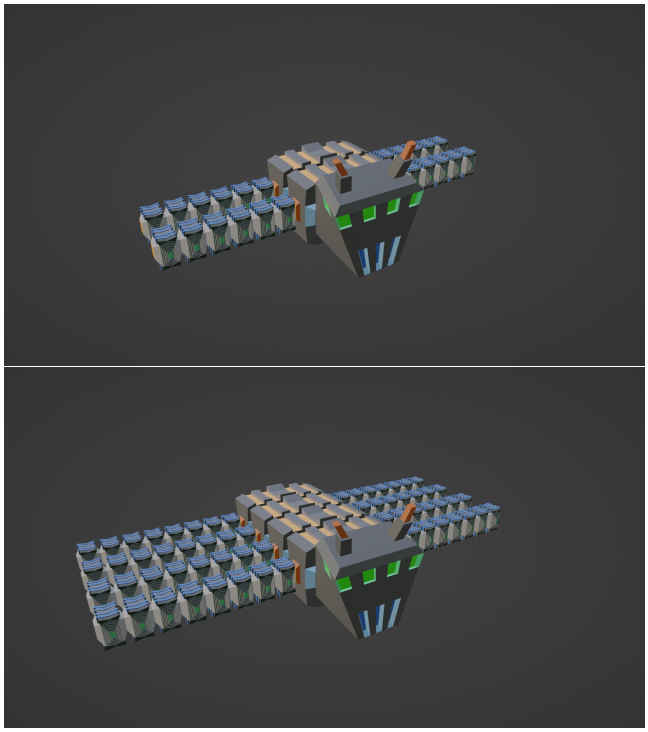

Figure 1. Two examples of 3D models created with Geometry Nodes. From top to bottom, seed 42 (*SegmentSize* = 2; *LegLength* = 6) and seed 555 (*SegmentSize* = 4; *LegLength* = 8)

3.2 Blender Scripting

The use of *Blender* Scripting for the character creation permitted us to define the rigging of the 3D models, which was not possible using the Geometry Nodes tool alone. *Blender* Python API proved to be very useful for automating many of the manual processes involved in rigging the model, resulting in a significant reduction in the time and effort required for this task. Additionally, the use of scripting allowed for greater control and flexibility in the rigging process, with the ability to modify and customize the rig to meet specific requirements.

We also set a default pose for each of the models to detect if the model was behaving properly. We use a single bone for the *Head*, a bone for each of the *Body Segments* and 3 bones for each *Leg*: A shoulder bone, connected to the *Body Segment*, an anterior bone and a posterior bone. These two last bones could have different sizes, and it is also set randomly by the script.

Overall, the use of *Blender* Scripting proved to be a crucial component in the creation of fully rigged 3D models of characters. In Figure 2 some examples of the output of the scripting can be seen, showing the character with fully rigged bones.

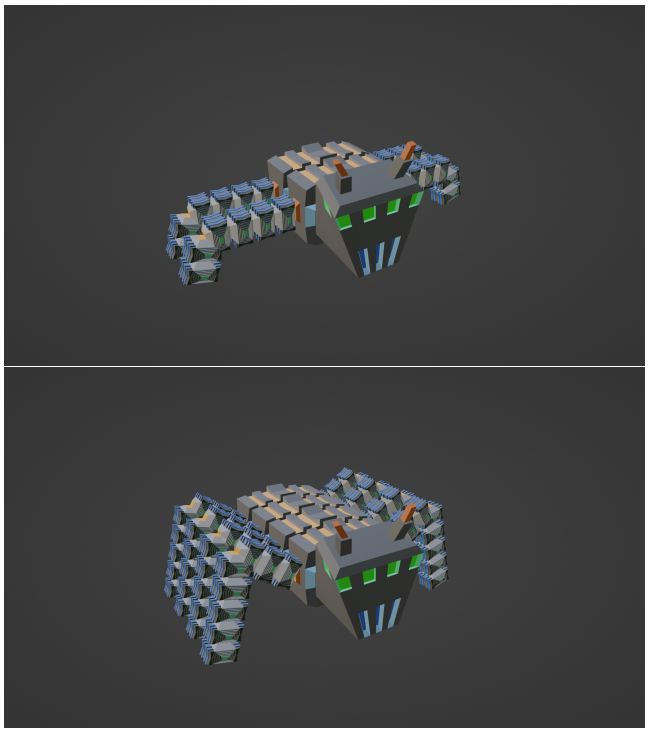

Figure 2. Two examples of 3D models created with Geometry Nodes and rigged by *Blender* Scripting. From top to bottom, seed 42 (*AnteriorLegLength* = 4; *PosteriorLegLength* = 2) and seed 555 (*AnteriorLegLength* = 3; *PosteriorLegLength* = 5)

References

[1] Blender Online Community. 2018. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam. http://www.blender.org

[2] Chris Conlan. 2017. *The blender python API: Precision 3D modeling and add-on development*. Apress.

[3] Ruan Lotter. 2022. *Taking Blender to the Next Level: Implement advanced workflows such as geometry nodes, simulations, and motion tracking for Blender production pipelines*. Packt Publishing Ltd.

A Blender Geometry Nodes

In this appendix, the geometry nodes used are shown in Figure 3, Figure 4, Figure 5 and Figure 6

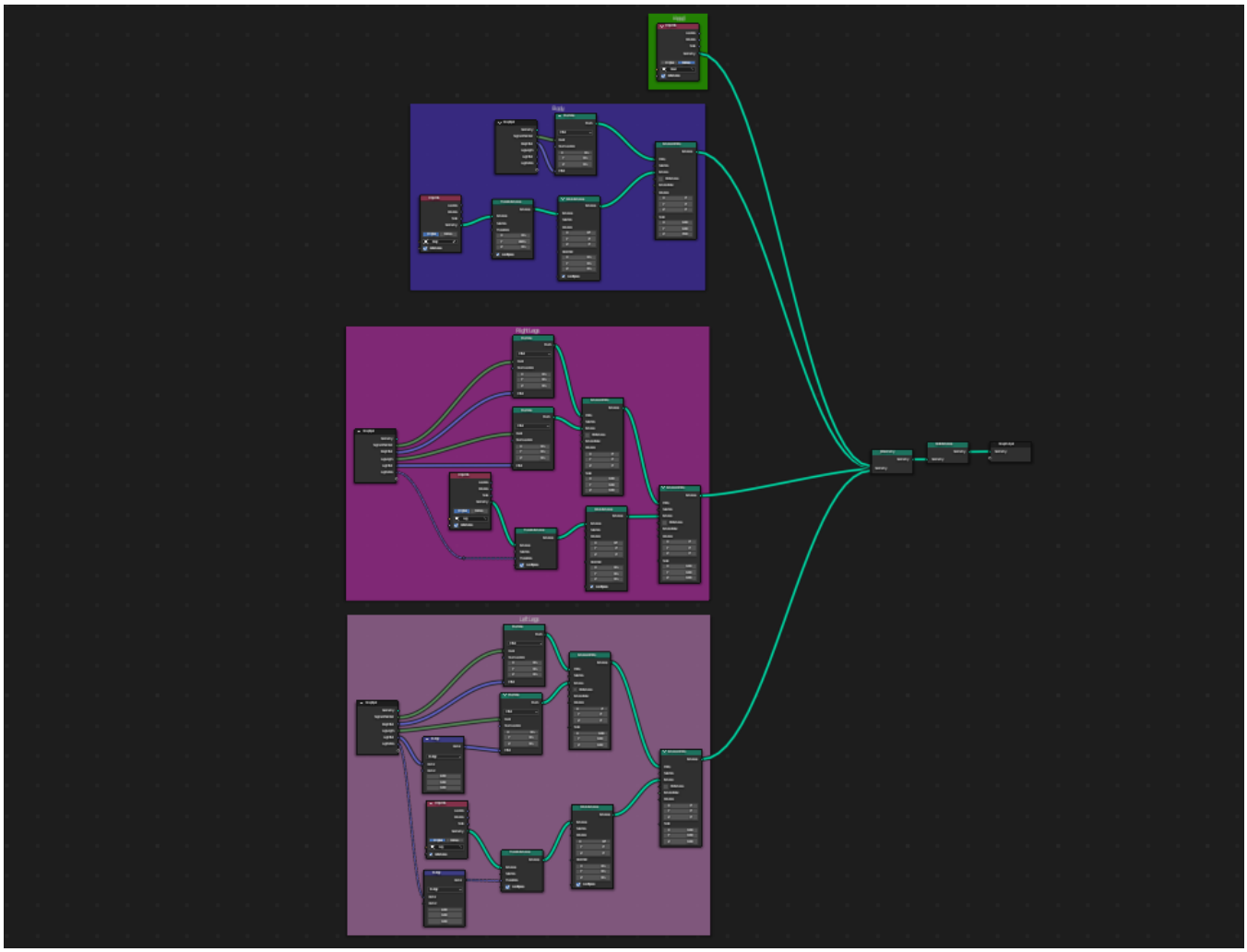

Figure 3. General view of the Geometry Nodes structure

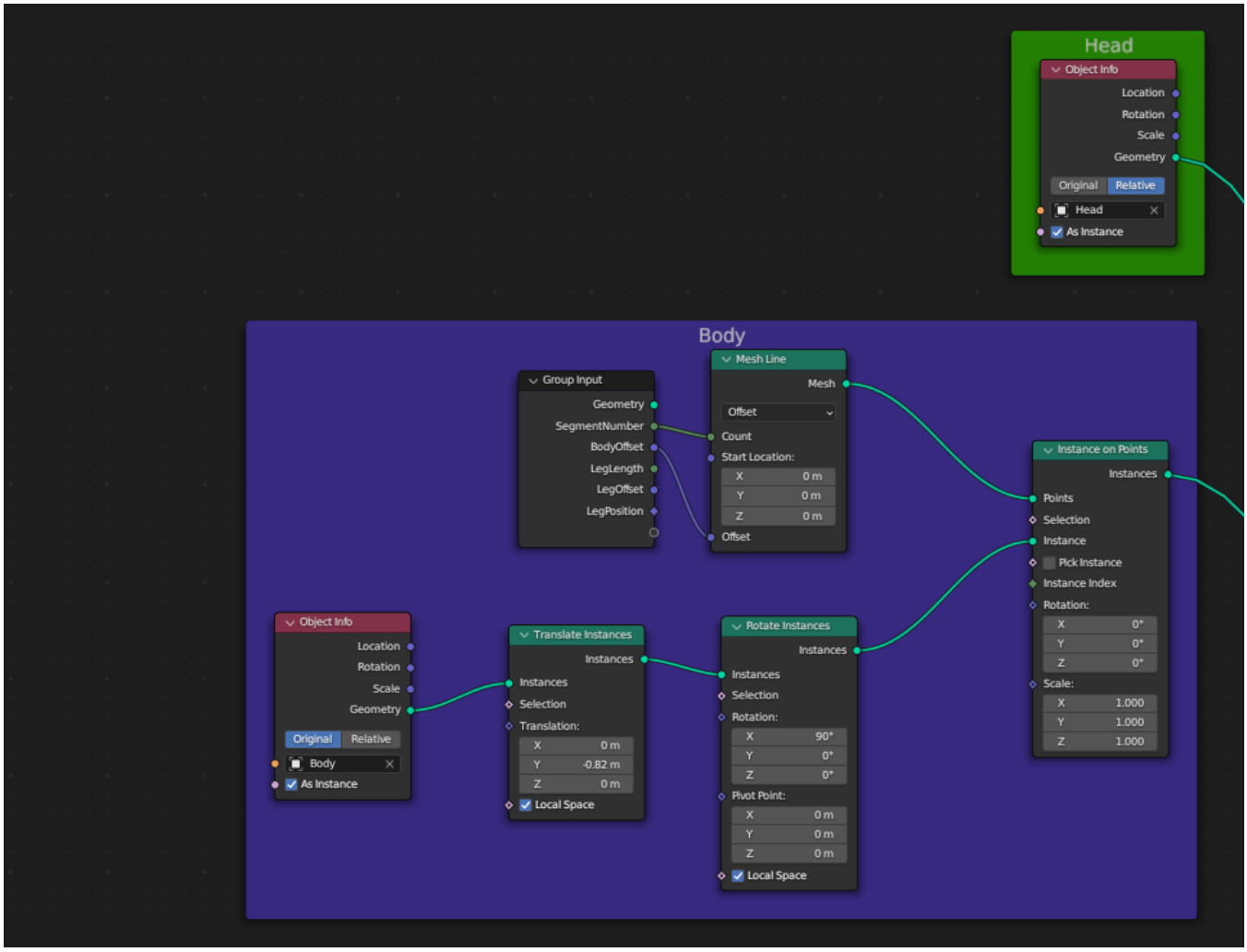

Figure 4. Head and Body of the Geometry Nodes structure

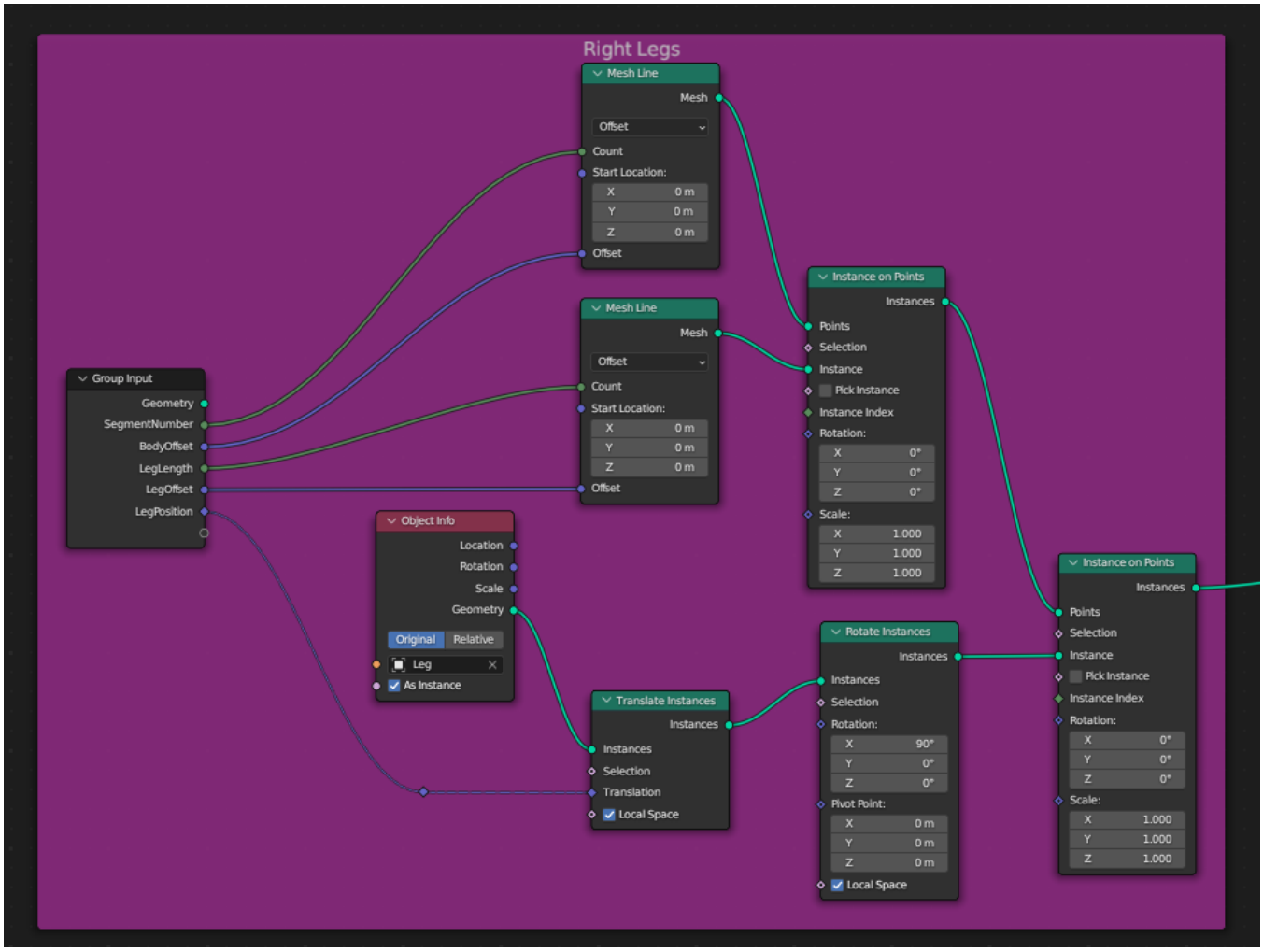

Figure 5. Right Legs of the Geometry Nodes structure

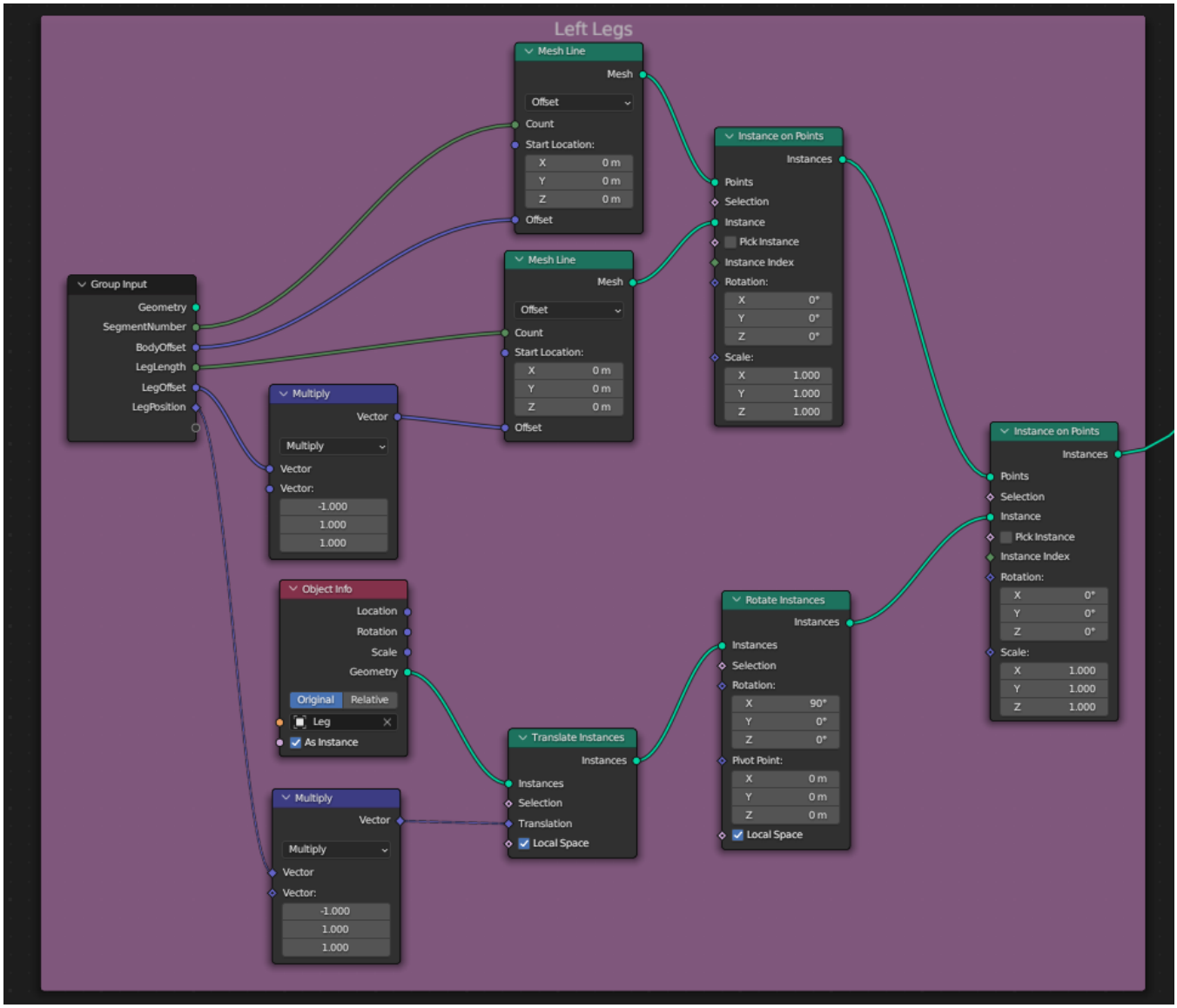

Figure 6. Left Legs of the Geometry Nodes structure