Computer Simulation of Deforesting Device

Ludvig Wendelius[*]
UMIT Research Lab

November 17, 2011

[*] ludvig.wendelius@gmail.com

UMIT Research Lab Ludvig Wendelius
November 17, 2011

Sammanfattning

Vid SLU i Umeå har en prototyp på ett skogsgallrings aggregat som kan skörda flera träd i en rörelse framtagits. Aggregatets syfte är att skörda oönskvärda träd på återplanterade kalhyggen. Detta projekt bygger på ett tidigare projekt med syfte att simulera och optimera detta aggregat. Under projektet har en modell implementerats och testats i fysik motorn AgX och viktiga variabler för optimerings processen har kunnat identifieras. Bland dessa är tillhör aggregatets hastighet.

Abstract

At SLU in Umeå a deforesting device that can harvest multiple trees in one motion have been constructed. The devices' function is to clear unwanted trees from replanted clear cut areas. This project is a continuation from another project to simulate and optimize such a device. During the project a model was implemented and tested in the physics engine AgX and certain important variables for the devices optimization was identified. Among these the velocity of the device.

Contents

1 Introduction

The two researchers Julia Forsberg and Rikard Wennberg have, at SLU, developed a device prototype for cutting and collecting several smaller trees in one motion. This is referring to trees between 6 to 12 meters in height. The reason for building the prototype is that most tree harvesting machines on the market are designed to harvest one fully grown tree at a time. When an area in a forest of pine or spruce is clear cut and then replanted other unwanted trees will also start growing in this area. After some time these unwanted trees must be cleared so that the planted trees may grow undisturbed. Today this clearing is done manually although the project with the prototypes' purpose was to investigate the possibility use deforesting machines instead.

The simulation project that is presented in this report is a continuation of another project that was made by the students M. Lundqvist, A. Hanga, T. Holmlund Olsson, H. Forsberg. These four students made a model in the software physics engine AgX developed by Algoryx with the goal to simulate and optimize the prototype.[0]

However the goal of this project was to use the results from the previous student project together with collaboration with the two researchers at SLU, make an improved model of the prototype using the same software. This model was to be presented at the date of the unveiling of the prototype.

[0]Simulering av ett skördaraggregat: M. Lundqvist, A. Hanga, T. Holmlund Olsson, H. Forsberg

2 Method

When this project started the *CAD prints* for the prototype was studied and compared to the physical model made by the students. With this study a decision was made to implement a completely new model. This decision was based on the fact that a lot had been changed to the prototype after the completion of the students report. Although the results from that report was taken into account with the new model. You can see the new and old model in Figure 1 and the CAD prints that was used as a design tool can be seen in Figure 2

From the CAD prints mentioned above the measurements for the implemented model could be found.

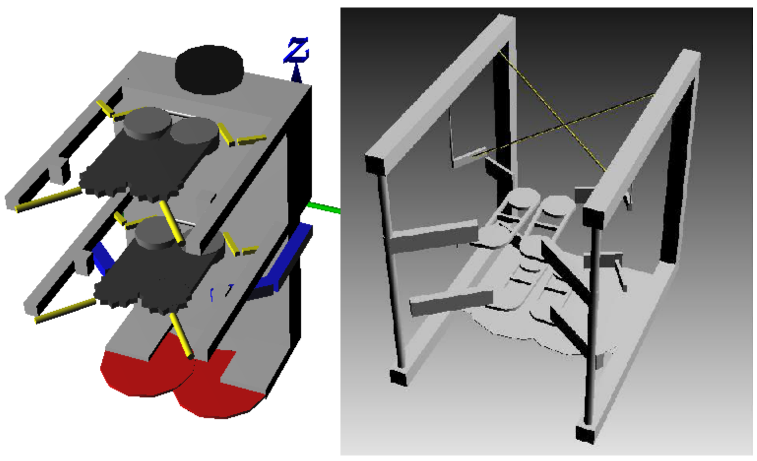

Figure 1: To the left is the new model, to the right is the older model.

The general idea behind the device is to be able to continually accumulate trees. It does this with the help of its feeding arms. When a tree have been cut the feeding mechanisms will grip it and transport it backwards. The tree is then caught by the wire stretching between the feeder pair. This is then repeated for multiple trees which are all accumulated in the chains. The prototype had an estimated maximum load of 300 kg of trees before having to be emptied. Figure 3 shows the accumulation of one tree.

Figure 1 is a little bit misleading because the real prototype did not have two pairs of feeding mechanisms (black area in new model) but only one. The reason for this is that one of the results from the student project was that two pair of feeder arms were needed in order for the device to catch the tree trunks before the cutting process. As well as having a tight controlled grip around the trunks during the accumulation. The two researchers from SLU also wanted to have this tested as they were curious of the behaviour of the device with one more feeder pair.

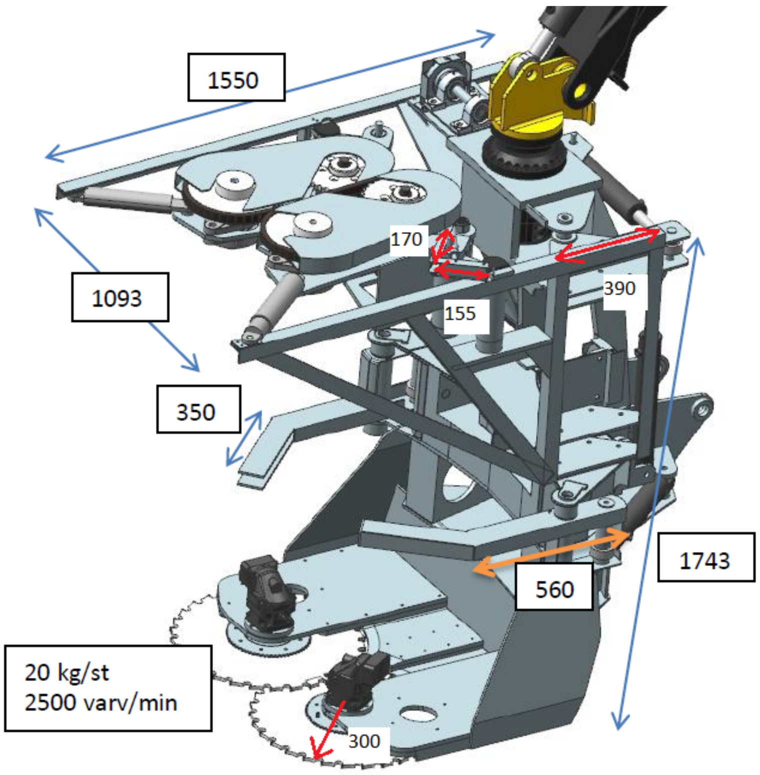

Figure 2: The CAD prints used for the design in AgX.

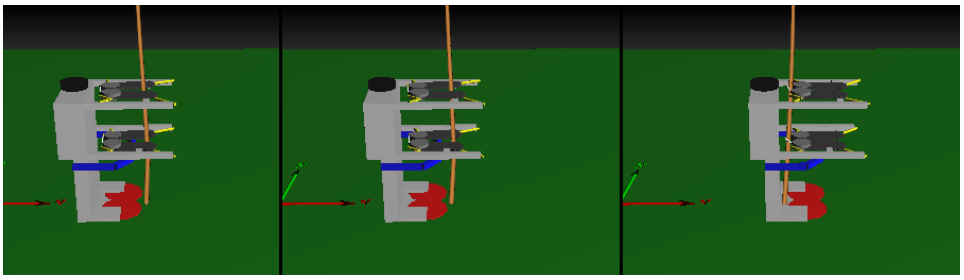

Figure 3: From left to right the process for cutting and accumulating one tree is shown.

3 Results

In the model that I implemented it is easy to switch between a model with two pairs of feeders and only one pair. It is also easy to change many of the sizes of the models parts, see Appendix A for more details about the script. I made test runs with both two pairs and one pair of feeders. The outcome of the two scenarios are shown in Figures 4 and 5.

After more testing with different lengths of the distance between the

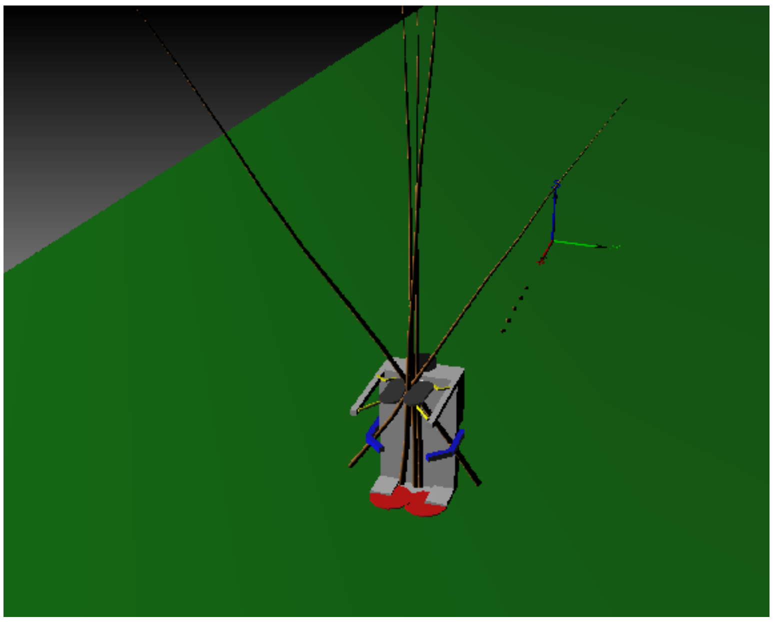

Figure 4: Shows what can happen if only one pair of feeders are used.

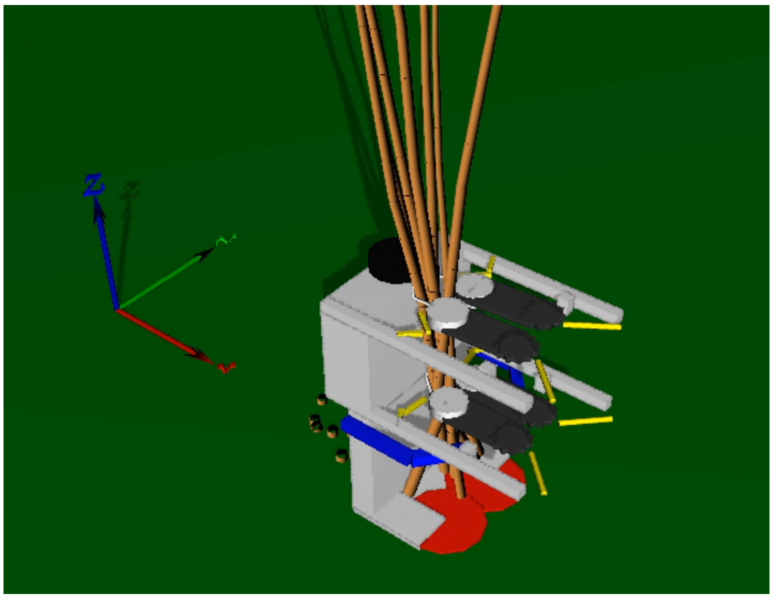

Figure 5: Shows the device with several trees accumulated.

cogs in the feeding mechanism I could see another result. This result was that the longer distance the cogs have between them the more displaced the trees could be from the center of the device when being cut. In a certain test run with the length between the cogs enhanced by 30% and the trees displaced $0.25m$ from the center of the device five out of five trees managed to be accumulated. This was simulated with the device having a speed of $0.15m/s$. Unlike using the cog distance from the CAD prints where the trees could not be displaced at all if they were to be accumulated.

So if the device had two feeding mechanisms which are longer. This model shows that the device could harvest trees that are further apart from each other in one single motion.

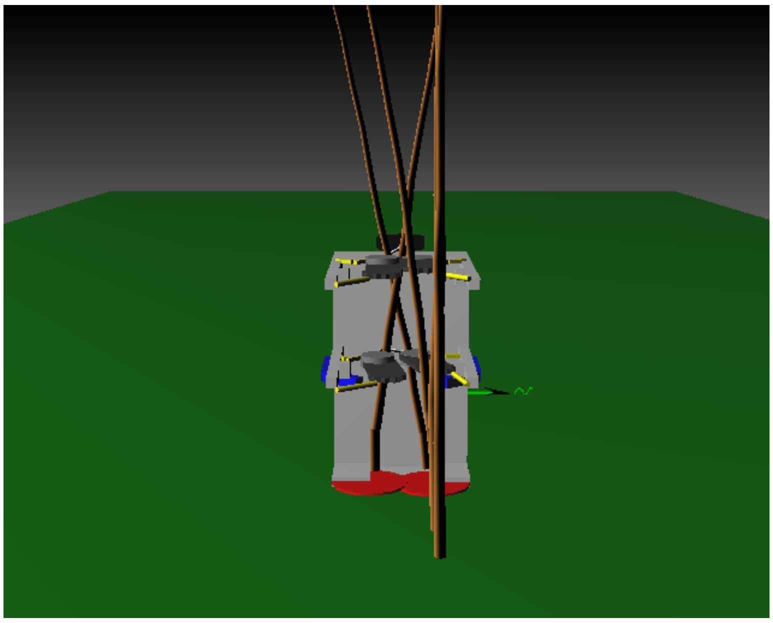

Figure 6: The device harvesting trees that are displaced from the center of the device's velocity direction.

4 Discussion

I agree with the previous group's conclusion that two pairs of feeder mechanisms are needed to hold the tree trunks in place. I.e. for the device to be able to control the trees and a have tight grip around them during the accumulation process. I am however unable to present the optimal parameters for the length of the mechanism. This requires further testing and optimization. As I see it from my test runs the interesting parameters are the following (see Figure 7 for explanation):

1. The distance between the two cogs driving the chain in the feeder.
2. The length of the cylinder attaching the front of the feeder chain to the frame work.
3. The spring constants. There are two springs in every feeder both of which can be optimized. In order for me to get my model to work we had to introduce a third spring between the big cylinders in the back in order for the feeder to grip the tree trunks properly. This spring is marked as 3'.

The reason for the third spring to be introduced is that the model had problems with the feeder could *be forced* outside the steel frame when impacting with a tree at high velocity. Also, the model did not grab the trees in the way it was supposed to. E.g. a tree that had been caught in between the two feeder chains was still able to fall out of the mechanism because the force clamping the tree was not large enough.

The speed of the device is also important. If the speed is too high the trees must almost hit the device in the center in order to be gripped and accumulated. Even higher speed and even hitting the device in the center is not enough. I suggest that this is subjected to further simulations as well.

One big problem I encountered has to do with the collisions between the trees and the front cog of the feeder. I you look at Figure 1 then you can see that the older model has thin cylinders in front of the feeder mechanism, whilst in the new model these are modelled as three thin boxes evenly rotated. Giving the cog the shape of a 12 edged star. When this cog was modelled as a cylinder the collision between the cylinder shaped tree did not result in the tree being forced in between the feeder. This can have to do with the collisions between two cylinders generates too small surface areas for the friction to bend the trees into the feeding mechanism. Changing to a star shaped cog did not eliminate the problem, but made it better.

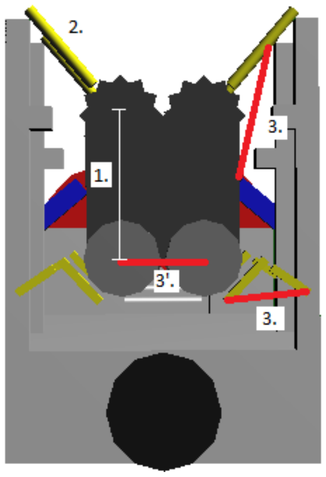

Figure 7: A top view of the device the numbers are corresponding to the enumeration list. (1) being distance between the cogs, (2) is the cylinder and (3) are the springs.

A Documentation: Device2Feed.lua

In this appendix the script for the device with two pairs of feeders will be documented. All functions in the file `Device2Feed.lua` will be given an over all explanation together with their inputs and outputs.

All of the functions are designed to construct one particular part of the device. In one specific function all of the certain variables needed for that particular part is defined. The diagram in Figure 8 shows how the different functions are called in the script. **Note** that one arrow in Figure 8 does not mean that the function is called only one. It could be called multiple times.

A.1 AssembleDevice

This is the main function that is called when you want to create the device model. Here you specify the initial values of the device such as position and velocity etc. It does not however create the materials used in the device. The material objects must be constructed and added to the simulation before

It is this function attaches the upper to the lower frame calling the functions that creates them. Then the two wires are created and attached to the feeders.

Input

name	type	description
root	*agxOSG.Group	–
sim	*agxSDK.Simulation	–

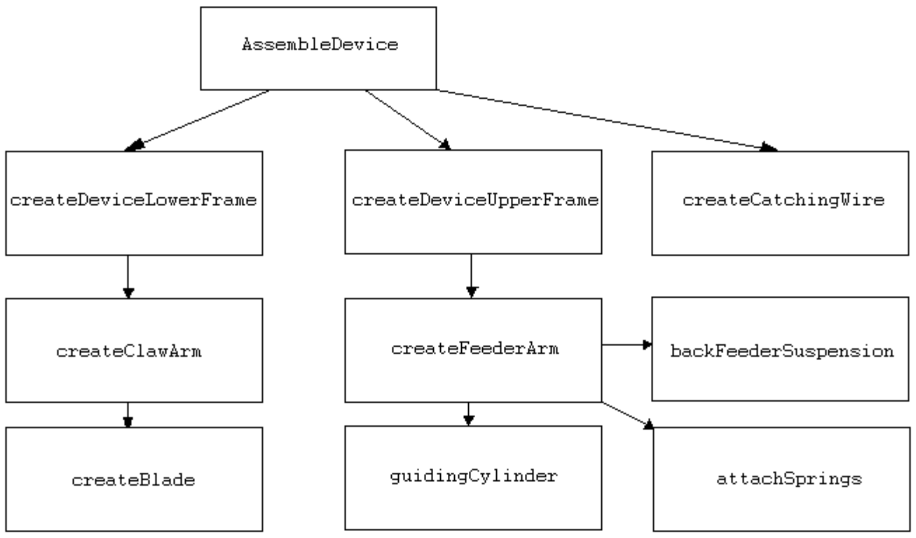

Figure 8: The chain in which the functions creating the device are called.

A.2 createDeviceLowerFrame

This function is called in the AssembleDevice function and creates everything that is necessary for the lower half of the device. Which is the steel frame, the two circular saw blades and the claw arms. Blades and claw arms are created in separate functions that are called here.

Parameters

name	type	description
depth	float	Steel frame x
length	float	Steel frame y
height	float	Steel frame z
backLength	float	Steel frames back y
plateLength	float	Length of the bottomplate
plateThickness	float	Thickness of the bottomplate

Input

name	type	description
root	*agxOSG.Group	–
deviceMaterial	*agx.Material	The material of the device
bladeMaterial	*agx.Material	The material of the blade

Output

name	type	description
lowerAssembly	*agxSDK.Assembly	The assembly containing all rigid bodies and constraints of the lower frame
lowerFrameBody	*agx.RigidBody	Pointer to the steel frame rigid body

A.2.1 createBlade

This function creates one of the spinning blades.

Parameters

name	type	description
bladeRadius	float	Radius of the blade
bladeThickness	float	The thickness of the blade
bladeRotation	float	The rotation speed of the blade

Input

name	type	description
root	*agxOSG.Group	–
bladeMaterial	*agx.Material	The material of the blade
attachmentBody	*agx.RigidBody	The rigid body that the blade is attached to
pos	*agx.Vec3	Blade CM position in attachmenBody's frame coordinates
isRight	bool	True if right blade, in cut direction

Output

name	type	description
bladeBody	*agx.RigidBody	The Blade rigid body
hinge	*agx.Hinge	The Hinge attaching the blade to the attachmentBody

A.2.2 createClawArm

This function creates one of the claw arms used when the trees are dump from the device.

Parameters

name	type	description
upperLength	float	Length of the arm geometry closest to the steel frame
lowerLength	float	Length of the other arm
barThickness	float	Thickness of the arm
depthDisp	float	The arm displacement in x
heightDisp	float	The arm displacement in z
angle	float	The angle between the two arm geometries

Input

name	type	description
root	*agxOSG.group	–
deviceMaterial	*agx.Material	Material of the device
attachmentBody	*agx.RigidBody	The rigid body that the blade is attached to
isRight	bool	True means right, in cut direction

Output

name	type	description
clawBody	*agx.RigidBody	The Blade rigid body
hinge	*agx.Hinge	The Hinge attaching the claw arm to the attachmentBody

A.3 createDeviceUpperFrame

This function is called in the AssembleDevice function and creates everything that is necessary for the upper half of the device. Such as the four feeders together with suspension and springs.

Parameters

name	type	description
depth	float	Steel frame x
length	float	Steel frame y
height	float	Steel frame z
backLength	float	Steel frame back y
barThickness	float	Thickness of the bars

Input

name	type	description
root	*agxOSG.group	–
deviceMaterial	*agx.Material	The material of the device
feederMaterial	*agx.Material	The material of the feeder

Output

name	type	description
upperAssembly	*agx.Assembly	The assembly containing all rigid bodies and constraints
upperFrameBody	*agx.RigidBody	The steel frame
chainRight	*agx.RigidBody	The chain body to the right top feeder
chainLeft	*agx.RigidBody	The chain body to the left top feeder
cogRight	*agx.RigidBody	The distance cog body to the right top feeder
cogLeft	*agx.RigidBody	The distance cog body to the left top feeder
chainRight2	*agx.RigidBody	The chain body to the right bottom feeder
chainLeft2	*agx.RigidBody	The chain body to the left bottom feeder
cogRight2	*agx.RigidBody	The distance cog body to the right bottom feeder
cogLeft2	*agx.RigidBody	The distance cog body to the left bottom feeder

A.3.1 createFeederArm

In this function one entire feeder mechanism is created. The catching wires must be created separately though beacuase they need both the right and left feeders to rout the wire. This function uses the functions attachingSprings, guidingCylinder and backFeederSuspension to create the feeder.

Parameters

name	type	description
cogRadius1	float	Radius of the front cog wheel, in the cut direction
cogRadius2	float	Radius of the rear cog wheel, in the cut direction
cogDistance	float	Distance between the two cog wheels

Input

name	type	description
root	*agxOSG.group	–
deviceMaterial	*agx.Material	The device material
feederMaterial	*agx.Material	The material of the cogs and chain
attachmentBody	*agx.RigidBody	The rigid body that the blade is attached to
data	*agx.Vec3	Depth, length, height of the frame that the arm chould be placed on
backLength	float	Length of the frames' "back"
barThick	float	Thickness of the bar in the frames
isRight	bool	True means right, in cut direction

Output

name	type	description
feederAssembly	agx.Assembly	The assembly containing all rigid bodies and constraints of the feeder
chainBody	agx.Rigidbody	The rigid body chain
distanceBody	agx.RigidBody	The distance cog body to the left top feeder

A.3.2 attachingSprings

Here the springs of the feeder are connected to it. Modelled as two distance joints.

Parameters

name	type	description
mechConstant	float	Spring constant of the regular spring
gasConstant	float	Constant of the gas spring
gasDamping	float	Damping for the gas spring

Input

name	type	description
chainBody	*agx.RigidBody	1st attachment of mechanical spring
cylBody	*agx.RigidBody	2nd attachment of mechanical spring, 1st attachment of gas spring
frameBody	*agx.RigidBody	2nd attachment of gas spring
frameLocalPos	*agx.Vec3	position in local coordinates to attach gas spring to frame
isRight	bool	true if right, false if left

Output

name	type	description
mechanicalSpring	agx.DistanceJoint	The mechanical spring
gasSpring	agx.DistanceJoint	The gas spring

A.3.3 guidingCylinder

This function just creates the cylinder in front of the feeder, helping the trees to me guided into the chains.

Parameters

name	type	description
cylLength	float	Length of the cylinder
cylRadius	float	Radius of the cylinder

Input

name	type	description
root	*agxOSG.group	–
material	*agx.Material	The material of the cylinder
cogPos	*agx.Vec3	The position of the front cog Cm
cogRadius	real	Radius of cog
data	*agx.Vec3	Depth, length, height of the frame that the arm could be placed on
barThick	real	The thickness of the bars
color	*agx.Vec4	Vector with the color for the
isRight	bool	True if right, false if left

Output

name	type	description
cylBody	agx.RigidBody	The cylinder rigid body
cylFramePos	agx.Vec3	Position in steel frame coordinates where cylinder should be attached
cylChainPos	agx.Vec3	Position in steel frame coordinates where cylinder should be attached to the chain

A.3.4 backFeederSuspension

With this function the back suspension of the feeder is created.

Parameters

name	type	description
length1	float	Length of the shorter bar
length2	float	Length of the longer bar
barThickness	float	Thickness of the bars

Input

name	type	description
root	*agxOSG.group	–
material	*agx.Material	The material of the suspension
framePos	*agx.Vec3	The position where the suspension attaches to the device frame
chainPos	*agx.Vec3	The position where the suspension attaches to the cogs and chain
color	*agx.Vec4	Vector with the color for the suspension
isRight	bool	True if right, false if left

Output

name	type	description
bar1	*agx.RigidBody	The shorter bar
bar2	*agx.RigidBody	The longer bar
hinge	*agx.Hinge	The Hinge connecting the bars

A.4 createCatchingWire

The wires that the trees are accumulated into are created and routed here. This function is called by the assembleDevice function but uses agx.RigidBody's that are created in the function createFeeder.

Parameters

name	type	description
wireRadius	float	The radius of the wire
wireLength	float	The length of the wire
wireResolution	float	Number of lumps per meter of the wire
forceRangeParam	float	The force that the winch can withstand

Input

name	type	description
attachBody	*agx.RigidBody	Body to attach the wire to
winchBody	*agx.RigidBody	Body to attach winch to
Y_displacement	realValue	Displacement in Y direction
isRight	bool	True if winch to the right, false if to the left

Output

name	type	description
wire	*agx.Wire	The created wire