COS426 Final Project Write-Up: Branching Simulation with Infinigen

Suhas Kanamarlapudi & Quinn Russell

Abstract

Our project aimed to simulate branching in Infinigen. We took initial inspiration from tree branches, and so modeled our simulation after that. We were able to accomplish a basic branching simulation, which - though rudimentary - does resemble a tree. We also simulated leaves, creating a final product that looks like a cherry blossom tree. We modified the ground of our simulation as well, and created several parameters involving the tree branches, leaves, and ground so that the tree's appearance changes, while staying within range enough to still resemble a tree.

Introduction

In this project, we tried to create a branching simulation that represents a cherry blossom tree on a bumpy, texturized ground using Python and Blender. We believe that our creation will provide an asset for scenes that desire a computationally simple, yet visually appealing scene. This saves time for other 3D designers, who do not have to re-create their own version of a cherry blossom tree. Additionally, those designers could use the parameters we incorporated to fine-tune the asset to suit their desired visuals. Other related work include other branching simulations, such as those of lightning and roots. Additionally, more computationally expensive trees exist with more detailed textures. These previous approaches can be more than necessary for many applications, and create longer render times and require more processing power. On the other hand, when more detailed and life-like assets are necessary, the existing more complicated assets would look better. To create the tree and ground, we used mathematics and geometry to create a purely procedural asset. By programmatically combining individual objects and moving them in calculated ways, we assemble a mesh piece by piece. Then we apply simple mathematical textures instead of using external sources or manual painting. This approach works well for simple, high-performance applications, as there are fewer calculations that need to be done. Additionally, by generating the detail mathematically, we do not have to rely on external references that need to be loaded into memory. Furthermore, our asset works well when the camera is taking photos straight at the mesh, as it has enough detail and complexity to look good straight on.

Methodology

In order to implement our approach, we needed to create and place the components of the tree, create and place the ground, modify the texture and geometry of the tree, and modify the texture and geometry of the ground. For the creation of the tree components, we decided to build up the tree piece by piece by placing cylinders on top of others and rotating them. Another possible implementation was to separate the cylinder mesh, and modify the geometry from there, or to create vertices first and connect them with cylinders. Our method was simpler and

computationally performant, as it exclusively utilized primitive mesh placement and trigonometry to accomplish the branching effect, along with recursion. We began with a single cylinder as the trunk, then used a recursive function to add branches to each level, using a cylinder generated in the previous level as a new “main branch”. Furthermore we used trigonometric equations to determine the difference in position for the ends of the cylinder in order to offset them when placing new cylinders in order to ensure connectivity. Modern hardware often performs trigonometric operations with high performance, so this assisted in the speed of our implementation. However, with our implementation it was difficult to control branch overlap and to consistently ensure connectivity between the branches as they were each separate cylinders. Separating the mesh instead could have ensured connectivity, but would have required more performance and operations as new vertices would have to continuously be added after every mesh split. The vertex method would have ensured connectivity as well as control branch overlap, but it is difficult to procedurally create cylinders that connect vertices using the bpy library. We chose our implementation because it was within the scope of our programming abilities and utilized trigonometry to make an appealing result. The other implementations would have been beyond our skill-set and require significantly more time to implement. For the creation of the ground and the adding of texture, we began with a plane, and moved it to the desired location. We then use Blender’s subdivision utilities to subdivide the mesh and create more vertices. Then we procedurally added noise to the vertex positions as well as color to create the final ground output. This implementation was simple, within the scope of our skillset, and performant. The other implementation would have been to place individual meshes representing texture objects onto the base plane. This would allow for more detail in the ground’s noise, but it would come at the cost of loading more meshes into the scene. For the texturing and colors of the tree, we used modified spheres as simple blossoms. Then we procedurally added them to the branches, using density as a parameter. We did not add a texture to the leaves, as there were too many of them for performance to be maintained. In order to provide roughness to the branches in order to imitate bark, we used a noise texture, similar to the process for the ground.

Results

Since our branching simulation was roughly modeled off of/inspired by tree branches, we studied properties of actual tree branches in order to determine the qualities that our simulation should have in order to be successful. Some of the most defining tree branch features that we felt needed to be represented in our simulation were as follows: the branches should all be connected; they should split off from each other; the branches should be at different angles from each other; the branches should be different lengths; the branches should have varying thicknesses, and the branches should all stem (either directly or indirectly) from the same initial branch (i.e., the tree trunk), which should be thicker than the other branches. In order to test these qualities, we changed many of the parameters that we were using to create the branches. For example, we played around with the radius of the branches, raising it and lowering it until we found a radius size we liked. We then experimented further to find a range of radii that would

look acceptable for the branches, so that we could make the radius one of the changing parameters. We chose a total of six parameters that could be changed: the branch lengths, the branch angles, the radii of the branches, the number of branches, the amount of noise in the ground texture, and the density of the leaves on the branches. For each of these parameters, we played around with the values until we found ranges that we liked, and then clamped the parameters to these respective ranges.

Our resulting images show that we were successful in creating a branching simulation where these parameters were changing and where the tree also looked semi-realistic. That being said, it is clear from the images that our simulation leaves something to be desired in terms of *how* realistic it is. Our simulation does not look super realistic, so in the sense of achieving a completely photorealistic branching simulation, one could say that we were not successful. However, we did not expect to be able to make a simulation that was entirely photorealistic, and our primary goal was to make a branching simulation - which we did.

Discussion

Having now completed the simulation, we don't believe that the approach we took is promising. We believe that we squeezed as much out of the rudimentary method we used as we possibly could. So if anyone else wanted to attempt their own branching simulation, we would recommend that they try other methods. There are a couple of upsides to our method, though; it is certainly beginner-friendly, and because of its simplicity it is less computationally intensive than some other simulations might be. Infinigen simulations tend to become slow and expensive quite easily, so our method would work well for someone who doesn't have much computing power. It would also be helpful to anyone who is just starting out with Blender and Infinigen, as its simplicity helps you get a feel for the software and existing code base.

There are a few more promising approach ideas to branching simulations. One such approach would involve splitting the branch geometry itself. Our approach, rather than splitting the geometry in the fashion of "true" branching, creates individual branches, rotates them, and then creates branches from the ends of those branches and repeats the process. So, each branch is its own geometry. A better approach would be to start with one branch and actually split that branch into multiple new branches, and recursively split all of the new branches. Another way to approach a branching simulation would be to use Bezier curves to model the branches. We attempted to do this in the process of creating our tree, and found that it looked more realistic. However, it was much more difficult to add branches using the Bezier curve method, enough so that we were unable to figure out how to move ahead with it. This pushed us to revert back to our original, simpler method. Somebody with more knowledge of Blender, Infinigen, and Bezier curves would probably be able to get a lot further with this method, though. It would likely produce a much better (i.e., more photorealistic) branching simulation.

To follow up on this, we would recommend trying one of the aforementioned implementations. As we noted, our approach to the branching simulation was quite simple, and the limit to how realistic it can get is not very high. It could probably be improved further, but it

would likely never reach the point of looking truly photorealistic. So, to continue modeling branching, the best thing to do to achieve a better-looking result would be to take a different approach (i.e., Bezier curves, real branching, or something else).

Even though our simulation wasn't the most realistic thing in the world, we certainly learned a lot during the process of creating it. Neither one of us had ever worked with Blender before, and we hadn't even heard of Infinigen, so there was definitely a learning curve there when working to create our simulation. We still aren't experts in either topic by any means, but we went from knowing absolutely nothing about either topic to now having a basic understanding of both and being able to create things using the combination of the two. We also learned a lot just by merit of this being our first truly creative computer graphics assignment. Neither one of us had worked with computer graphics prior to taking this class, so everything we know and have done with graphics has been through this class. Our assignments have helped us learn a lot about graphics, but it is quite different to actually create our own project, rather than following more rigid assignment guidelines to implement projects that have already been done and that we have extensive references for. With this project, we had the opportunity to get creative with our implementation, and change it up as we went along. We tried simulating lightning, grass, leaves, snowflakes, cracks in ice, and lots more, which was fun for us and also allowed us to find something to simulate that we were more confident in and comfortable with, while also pushing ourselves to try new things. It was definitely daunting, especially given how little documentation there is on Infinigen, but it was a great learning experience for both of us.

Conclusion

Our original goal with this project was simply to create a branching simulation. We did not expect to be able to create something that was completely photorealistic, so that wasn't a goal that we set for ourselves. We did do our best to make it as realistic as possible, though, spending hours tweaking textures, colors, and parameters to get our branches to look as tree-like as we could. In that sense, since our goal was quite simple, we believe we accomplished it. We did, in fact, create a branching simulation; the branches are there, and they have the properties that we intended for them to have. There were some flaws still, but our original goal was accomplished and we made a better-looking simulation than we thought we would, so we're quite happy with the result.

The next steps in this simulation would be to make it more complex and thus realistic, ideally using one of the methods mentioned above. We were limited both by time and by our abilities with this project; but in a perfect world we'd have no time crunch and would be able to experiment with Blender and Infinigen much more. We'd be able to learn a lot more about the software in that case, and would have a much better understanding of the Blender interface, existing Infinigen assets, and how we could best use them together. Hopefully, we'd then be able to create a more realistic branching simulation.

There are a number of flaws with our current simulation. For one, the branches often go through each other, which would not happen in an actual tree. If this branching simulation was

meant to emulate cracks in ice, or something of that nature, then this would be fine, but since it is modeled after a tree, that would need to be fixed. The shape of the branches is also not entirely realistic, since we used cylinders to model them, so if we revisited this simulation we would try to modify the branches to have a less uniform and more natural shape. Additionally, we would have liked to spend more time modifying the branches' texture, as the one we ended up creating makes the branches look a bit like churros (which is not what real trees look like). One other thing we'd like to revisit would be the angles between the branches and the lengths of the branches. In our simulation, there isn't a ton of variation between the branches' lengths, which makes the tree look less natural. The angles between the branches are also somewhat restrictive currently, which leads to more collisions between branches and more of a bush-like shape than a tree-like shape. These are the main issues we believe should be addressed; though, if implementing the branching simulation using one of the other methods, these issues might be resolved.

Contributions

Suhas primarily coded the math behind the simulation (i.e., branch placement, branch rotation, branch connection), while Quinn worked on the visuals (i.e., textures, colors, leaves, object generation). We worked in tandem to determine what looked good and what didn't, tweaking values in the math when necessary. Overall, it was a very collaborative project, and the majority of it was done in a pair programming style.

Works Cited

We used the bpy (https://pypi.org/project/bpy/) and infinigen (https://github.com/princeton-vl/infinigen) python packages to create our asset. We also leveraged the tutorials provided in the Infinigen Github and the Youtube Blender tutorial linked in the specification document (https://www.youtube.com/playlist?list=PLjEaoINr3zgFX8ZsChQVQsuDSjEqdWMAD). We did not install any other external libraries for this project.