Tree Growth Simulation based on Ray-Traced Lights Modelling

Bence Tamás Tóth

John von Neumann Faculty of Informatics
Óbuda University
Bécsi street 96/B, Budapest, H-1034, Hungary,
toth.bence@nik.uni-obuda.hu

Sándor Szénási

Faculty of Economics
J. Selye University
Bratislavská cesta 3322, 945 01, Komárno, Slovakia,
szenasis@ujs.sk

Abstract: In the fields of forestry and horticulture, it is necessary to have forecasts about the growth of trees. This process is affected by a lot of external factors like weather, light conditions, other objects, etc. There are several already existing methods for this purpose, but these can give only rough estimations. This paper presents a novel solution, based on the simulation of the growth of the tree. During this process, the application takes into account the environment of the tree and the properties of the species, which parameters are all easily configurable. The presented application can simulate not just one, but a group of trees parallel, estimating their effects to each other (shadows, etc.). The result of the simulation is a three dimensional model of the tree(s) at any time of the growth process. The distortion effects of these external factors are well visible on this model, giving a realistic estimation about the integration of the tree(s) into the given environment. The simulation has high computational demand; therefore, the most computationally intensive steps of the simulation are implemented on graphics accelerators using the CUDA framework.

Keywords: Computer Simulation; Computer Graphics; Ray-tracing; GPU programming

1 Introduction

In practice, experts of the fields of forestry and horticulture would like to see the state of a given tree in the next decades. This should be essential for gardeners to design the estimated landscape of an area and for foresters to maximise the profit from the trees. The simulation of tree growth based on the environmental effects

can help giving answers for further scientific questions related to climate change, air pollution [1], education [2], and other fields [3].

But the growth of trees is a very complex process because it is affected by several external factors (weather, light conditions, other objects, etc.). Although, every tree can be distinguished by its type traits, but in reality, these can be heavily altered by external forces. After a given tree is rooted in a given place, it has to evolve to withstand a wide range of environmental effects. The main guiding principle of the authors is that to simulate a natural phenomena, the best way is to follow the steps of nature. Nowadays, there are some popular algorithms following this approach, and the method presented in this paper is also based on this principle. It simulates the growth of a given tree from the first day. During this, it takes into account the properties of the given species and the surrounding environment [4].

In some cases, it is not enough to reflect on the static surrounding factors, because it should be also important to take into account the neighbouring trees. In the case of planting multiple trees at once, taking into consideration their effects on each other are also necessary. The presented algorithm is also suitable for this kind of simulations, the final result shows a good estimation of the future shape of all trees.

The main mechanism of the presented model consists of two separate steps:

- The determination of the shape of the tree without any environmental effect.
- Deformation of this shape based on the environmental factors.

It is necessary to repeat this process iteratively every year and recursively for every branch of the tree. After the basic growth step (given by the species properties), the deforming modifiers are applied one by one from a list which can be easily updated or expanded.

The presented model contains the following potential factors to configure the tree growing:

- Type traits
- Heliotropism
- Collisions
- (Self) shades

The main goal of this research is the development of an easily configurable model which creates visualisation of trees in a given environment. That can help professionals and hobbyists to plan their garden or even to choose the correct tree species for populating a new forest.

The rest of this paper is structured as follows: Section 2 contains a state-of-the-art overview of the already existing theoretical methods and implemented systems. The next section shows the methodology of how to simulate the growth of a given tree. Section 4 presents the experimental validity and runtime test results; and finally, the last section contains the conclusions, limitations and further plans.

2 Related Work

The simulated growth of trees is a main task in the field of systems biology and mathematical biology to reproduce plant morphology with computer simulations. There are several related attempts to create realistic tree models [5, 6, 7]. In the past, these were only ray-traced renders; but later, simulations also appear for this purpose. A significant difference should be made between the main approaches of the latter alternatives [8]:

- Developmental models start with a single root element and produce further elements by adding and dividing already existing items of the model. These are very accurate and well usable models with high computational demand. The real-time usage of these is not a real alternative.
- Non-developmental methods are based on prefabricated models. After the insertion of these, the model undergoes a series of necessary modifications and adaptations. This approach has the advantage of very fast response time, but the result is usually less realistic. Nowadays, it is common to use non-developmental models to simulate plants in animations and computer games.

The main objective of this paper is the generate as realistic models as possible (the runtime is less important); therefore, the non-development approach is not acceptable. The growth of a plant is a complex, continuous process altered by several external factors. There are already existing computer simulation based methods to replicate this behaviour with various models. The most commonly used one is the Lindenmayer System model (L-Systems) [9].

The earlier variants of this model handled only the branching structure of the trees in a non-developmental way [10, 11]. The newer variants use several environmental factors to modify this process and create highly detailed developmental models [12] [13].

Based on its recursive approach, it is well applicable for simulation of self-organising objects. The process is based on a simple starting state and a rule set like (1).

$$\begin{aligned} &S\{A,B\} \\ &\alpha\{A\} \rightarrow \{B,B\} \\ &\beta\{B\} \rightarrow \{A\} \end{aligned} \tag{1}$$

After that, the algorithm executes recursive steps applying the given rules in all iterations. Based on this model, it is possible to generate realistic plants.

The original L-Systems implementation contains the following modification factors:

- Effect of gravitation
- Heliotropism
- Geotropism

- Longitudinal and transverse growth
- Collisions
- Rotation of branches
- Angle of branches

Our presented method uses some of the recursive properties of the L-Systems, but makes the model applicable for simulating higher level plants.

The "Structural simulation of tree growth and response" research project [14] had very similar objectives. It presented a mathematical model taking into account the energy consumption of the growth, the weight of branches, and the energy input given by photosynthesis. The developed system gave a good overall estimation of the shape of the tree, but it was not able to handle environmental effects.

It is also worth mentioning the work of Jason Weber and Joseph Penn [5]. Their approach was not simulation based. The shape of the generated tree was described by a simple rule set (shape of the tree, number of branches, subdivision of branches, angle of branches, etc.). According to these rules, it was able to build a tree body made of pyramids and having leaves as surface items. This method mainly focused on visualisation; therefore, it did not take into consideration any environmental effects.

From the visualisation point of view, it is also worth mentioning the already existing advanced modelling and rendering methods, like SpeedTree [15]. These are not simulations, but modelling tools. The end user has to set all necessary properties, and the application can generate a model according to these. Trees generated by this approach are the key components of 3D animations, computer games, and augmented reality applications [16].

Applications from the field of forestry have a very different approach than above. These are detailed simulations supporting the estimation of cost/benefit of tree production. There are very accurate and take into consideration all available environmental factors, but does not have any graphical output.

3 Methodology

3.1 Type Rules

To simulate the most defining traits of a tree, information should be collected from the field of Dendrology. The problem is that this field can help identify trees but not to create them. Therefore, as a preliminary step, we had to find the traits defining the look of a tree. The following properties have been identified:

- Longitudinal growth
- Transverse growth
- Branching angle
- Branching rotation

- Branching type which can be “whorled”, “opposite”, “alternate”, and “spiral”

These properties are not constant in the whole lifetime of a tree. It is necessary to differentiate several life stages as follows:

- 0-1 years: seedling
- 1-5 years: sapling
- 5-15 years: young tree
- 15-70 years: mature tree
- 70-150 years: seed-growing tree

The set of used rules reflects on this by having different parameter values for every life stage. The life stages may vary with different tree types.

As an additional option, different branch levels can also have different parameter values. In this context, branch level refers to the distance from the root. The algorithm is able to handle an arbitrary number of levels, but in practice, it is enough to use three of them:

- 1st level: the trunk of the tree
- 2nd level: branches grown from directly the trunk
- 3rd level: further branches

It is possible to set different parameter values for each branch level.

3.2 Light Detection

Heliotropism is the most decisive modifier because light is the most important resource for every species. Trees can observe light in multiple ways. The simulation uses virtual sensors located in the branch tip buds detecting the direction and amount of light. This information significantly influences the grow direction of this branch. Sensors use Monte Carlo ray-tracing to gain the requested information.

Ray-tracing algorithms are heavily used in three-dimensional graphics. These are used for rendering realistic images by tracing the paths of light as pixels in an image plane. These are capable of producing realistic results, but at the cost of very great computational. Basically, ray-tracing starts beams from one of the pixels of the image (camera) and tries to follow its path. It is able to simulate several optical effects, such as reflection, refraction, scattering, and dispersion. Based on this path, it is possible to determine the colour and intensity of the given pixel. The presented implementation is different from this basic process as it doesn’t produce a flat image, but explores the directions where light beams arrive at the virtual light sensors.

The special Monte Carlo ray-tracing [17] algorithm is used to simulate this behaviour of trees. In general, Monte Carlo algorithms are randomised procedures to estimate the value of very time-consuming computations. These algorithms are based on repeated randomised sampling, and they produce an output which might be

incorrect with a certain probability. This probability can be reduced using a higher number of random samples.

As an optimisation step, the presented light detection algorithm first looks for a direct line to any light source. If any of these exists, the dominant direction of light is from this point. If there are no such direct lines, then it starts the path tracing. During this, a beam of ray is started into a random direction. If it hits a triangle (all objects in the model space are described by the triangles forming their surface) it continues to a direction according to the rules of reflection, and repeats this process for a given number of times or until it reaches a light source. After that, it is able to start further beams, and at the end of this process, it results in an array of vectors representing the directions of potential light sources.

Based on this array, the light source with maximum energy is selected (the dominant light source direction). This will modify the direction of the branch growth leading to that light source.

3.3 Collision Detection

The growth of a tree is significantly influenced by the objects in its environment. It is obvious that a branch cannot move across or into any solid obstacles. The simulation is executed in a three-dimensional model space; therefore, it is possible to create and place any additional objects into this. There are no limits to the shape and size of these objects. Furthermore, the neighbouring trees can also be considered as obstacles too.

The collision detection is another part of the iterative growing process. It is done for every branch after the altered growing direction and length is determined. The potential new branch interval is checked against every triangle in its neighbourhood. If there are no collisions, the potential branch becomes real. If it collides with any of the triangles, it is necessary to alter the direction of growth.

This new direction is based on the projection of the growth vector to the plane of the triangle. This projection gives a new growth direction if it does not cause further collisions. This method has some limitations, for example, if the growth vector is perpendicular to the surface of the triangle then the projection is a point which stops the growing process of the given branch.

3.4 Pruning

It is possible to dynamically modify the structure of the tree. Selected branches and their sub-branches can be removed from the model. This lets more light for other branches, modifying their behaviour. Using this technique, it is possible to give more realistic results.

There are also some automatic pruning mechanisms. In the case of some species, branches without enough light become inactive and withered. These are usually the lower/inner branches of a tree, especially in a multi-tree environment. It is possible to automatically remove these parts.

3.5 Simulating Multiple Trees

It is possible to simulate not just one but multiple trees parallel. These are simulated year by year, sequentially one after the another. This may cause some problems with the trees effecting to each other in an unnatural way. For example, trees at the beginning of the iteration can grow over the others blocking more light from them. The prevention of this phenomena is that shades are calculated and updated at the start of every iteration.

The partially separated simulations of trees have several benefits. There are no limits to the number, type and age of the trees in the same model space. It is possible to plant different species at different times and run the simulation. The result will show a good estimation of the whole group.

3.6 Acceleration with GPUs

3.6.1 GPU Acceleration of Ray-Tracing in General

The well-known disadvantage of ray-tracing methods is the very high computational demand compared to other three-dimensional rendering techniques [18]. Tracking the light beams from their source to the final destination needs several mathematical calculations like reflection angles, collision projections, etc. This is the reason, why the accepted view is that this method is not applicable for real-time purposes.

Using the Monte Carlo method makes it possible to significantly decrease the number of these tracings, but it is also worth considering that more rays usually gives more accurate results. This means thousands of rays from one sensor. In the first years, this is not an issue, but as the tree becomes older, the number of branches increases exponentially. On an average CPU, it becomes days to simulate the changes for further one-year iterations, which is unacceptable.

Fortunately, ray-tracing is a typical embarrassingly parallel algorithm. Thanks to this, there are several parallel implementations and it is obvious that it is suitable for data-parallel implementations. Because all pixels of the target image are independent of each other, it is possible to fully parallelise the process. GPU implementations are usually based on the idea that it is possible to assign each pixel (camera ray) to one thread of the GPU. This means thousands or millions of threads, but this is what the GPU for [19, 20], the GPU based implementations can achieve significant speedup over the serialised version.

3.6.2 GPU Acceleration of Tree Growth Simulation

The novel GPU accelerated tracing method works mostly the same as the presented regular implementation. The major difference is in the pluralisation phase. Ray-tracing algorithms are easy to parallelise because these are usually completely data parallel. This means that each photon paths are independently tracked from each other and the branch traces are also separate from each other. This helps the implementation, because it is possible to handle the branches in CUDA blocks where the paths are handled by independent threads within those blocks.

The other difference is that the CPU version calculates every branch sequentially, but the GPU version calculates these at once at the beginning of the iteration. This needs the following consecutive steps:

1. Before the calculation, all data required for the ray-tracing process is copied to the device from the host.
2. GPU kernels run and calculate the results.
3. The results should be copied back to the host from the GPU.

This data transfer is very time-consuming because it uses the standard PCI-e bus. It is a hardware limitation, therefore there is no way to significantly decrease the requested time.

This causes the behaviour presented in the evaluation part that in the first few years of the simulation, the GPU version is slower than the CPU because the amount of data copied is large compared to the number of calculations (where the GPU can offset this disadvantage).

4 Evaluation

4.1 Validation

4.1.1 Validation of Technical Sub-steps

It is always hard to evaluate and validate the results of a nature-inspired simulation, because there are no gold standards to compare with. Because of the several randomised factors, it is also not possible to give an exact expected state from a given initial state (environmental and growth parameters).

The best thing to to do is checking the validity of the presented technical sub-steps and do visual examination on the simulation results compared to similar real-world examples.

Fig. 1 shows the result of the validation of the growth sub-step. All presented sub-steps are validated in a similar way (not detailed in this paper).

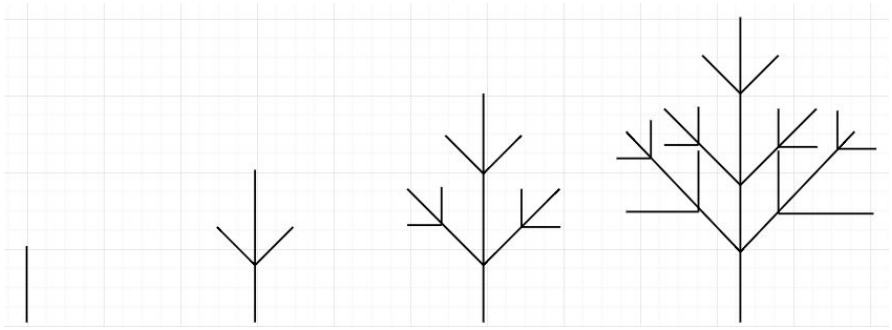


Figure 1
Validation of growing in the first 3 years.

4.1.2 Basic Tree without any Environmental Effects

Fig. 2 shows the result of a simulation of a basic tree without any environmental effects. The length and number of branches conform the given simulation parameters. As visible, it has the expected regular and symmetrical shape. It is also possible to

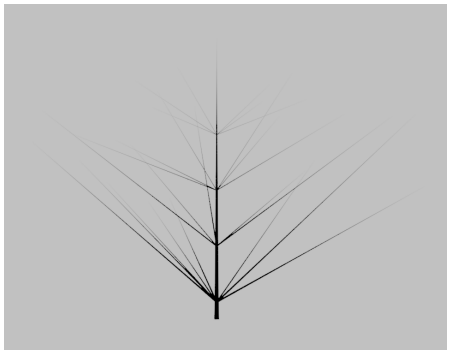


Figure 2
Tree without any environmental effects.

display the state of the tree at the end of every simulation year (Fig. 3).

Figure 3
8 years long simulation without environmental effects.

4.1.3 Light Detection

Adding some light sources to the model makes it possible to validate the effect of light tracking. Fig. 4 shows a tree grown with an external light source (from the upper left corner of the image). As expected, the tree is blended toward the light source.

4.1.4 Collision Detection

To validate the collision detector, it is possible to place an additional obstacle object into the model space. Fig. 5 shows some examples for simulations of trees grown near one or more external object(s). As visible, all trees were growing straight upwards in the first years (there were no external light sources to modify this behaviour). After that, some of their branches could not grow to the desired direction. According to the expectations, they had to find another direction to grow after the

Figure 4
Tree with light effect.

collision. Overcoming the obstacles, they continued the growing process straight upwards.

4.1.5 Complex Examples

To validate the cooperation of modifiers, Fig. 6 shows a more complex example. In the first few years, the tree cannot see any direct light source, therefore it started growing towards the left inner side of the box, where some light reflection comes. But when is becomes taller, it turns towards the direction of the light source. As visible, it took care about the collisions and found the hole in the top of the box.

To summarise the effects of these modifiers, Fig. 7a shows the result of a 5 year long simulation. The parameters of teak trees were used. To help the visual validation, Fig. 7b presents a real-world 5 years old teak tree. As expected, the main visual attributes of these are very similar.

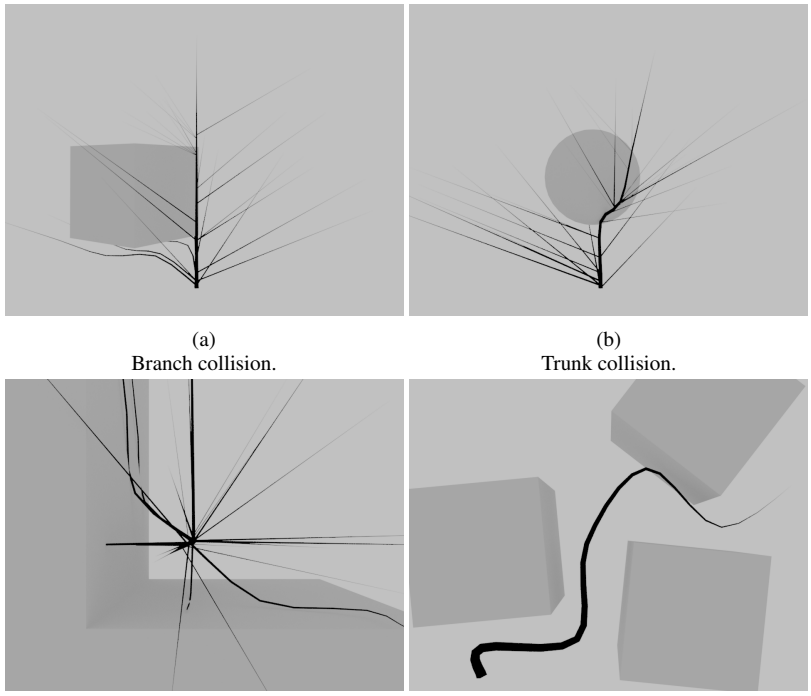
As a final test, Fig. 8 shows the result of a simulation on multiple trees. The rear one was deployed some years before the others, therefore it is larger.

4.2 Runtime

As mentioned, the tree growing simulation has very high computational demand. The most time-consuming part of the process is the ray-tracing step, which determines the direction and strength of dominant light in a given point. An efficient GPU based ray-tracing algorithm has been designed and implemented to speed-up this step.

The following hardware configurations were used for benchmarking:

- CPU configuration



(a)
Branch collision.

(b)
Trunk collision.

(c)
Upper view of collision.

(d)
Complex use case for multiple obstacles.

Figure 5
Validation of the collision detector.

  - CPU: Intel Core i5-2500K
  - Number of cores: 4
  - RAM: 8GB DDR3
  - TDP: 95W
- GPU configuration
  - GPU: NVIDIA GeForce GTX 1070
  - Number of CUDA cores: 1920
  - RAM: 8GB DDR5
  - MPC: 150W

Fig. 9 shows the runtime of the CPU and the GPU implementation of the ray-tracing algorithm for a one year simulation period. In the first few years, the tree is not enough complex to fully utilise all the available cores of the GPU. As a consequence, the execution time of the CPU implementation is smaller in the case of small (young) trees. But it is also visible, that the runtime increases faster in the

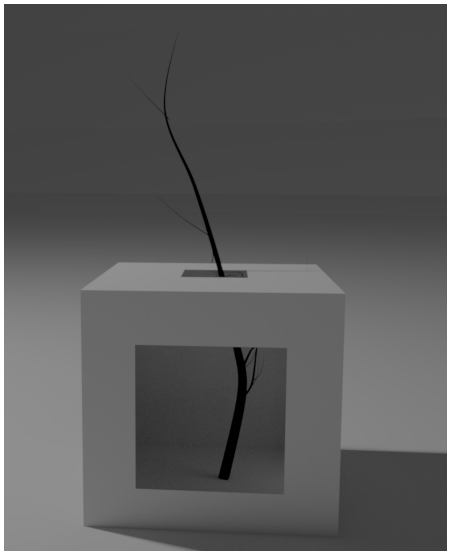


Figure 6
A complex example for light and collision detection.

case of the CPU. The measured runtimes are similar for 6 year old trees, and after that, the GPU clearly outperforms the CPU.

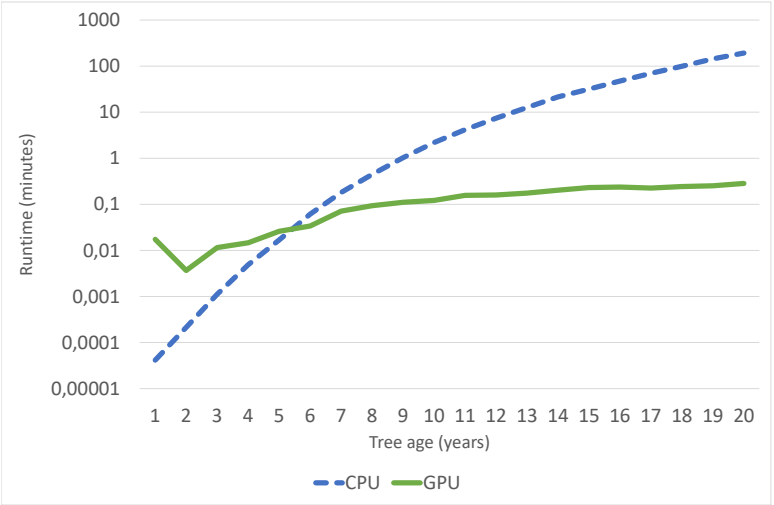


Figure 9
Year to year growth time

(a)
Real 5 year old teak tree.

(b)
Simulated 5 year old teak tree.

Figure 7
Comparison of simulated and real tree.

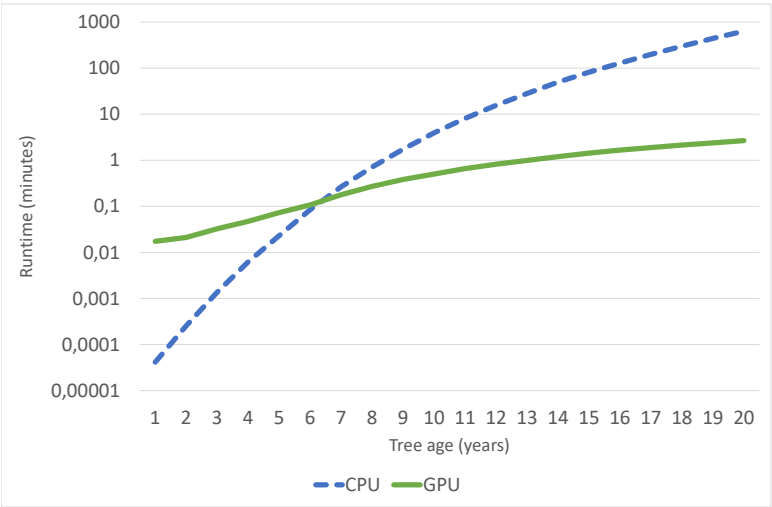


Figure 10
Total growth time

Figure 8
Simulating the growth of multiple trees.

The results of Fig. 10 are more interesting. These show the accumulated runtimes of the previous years, which are more important from the practical point of view. Obviously, the runtime of the CPU implementation is smaller for small trees. This changes after 7 years, when the overall runtime of the GPU becomes better. The runtime of both algorithms is exponential to the age of the tree (according to the exponential growth of branch count), but the runtime of the CPU implementation rises more steeply.

It was not necessary to compare the accuracy of the CPU and the GPU solution, because the underlying algorithms are the same. Therefore, the trees generated by the CPU and GPU are exactly the same. These share the same results regarding the validation steps.

In practice, only long-term simulations give valuable results. As a consequence, it is worth using the GPU implementation. It may worth considering to implement a hybrid approach which uses the CPU in the case of small trees and switch to the GPU for larger ones.

5 Conclusions

This paper presents a novel nature-inspired tree growth simulation algorithm to estimate the future states (shape, size, etc.) of a given tree according to the environmental parameters (light, obstacles, neighbouring trees, etc.), and its species characteristics (annual number of new branches; angle, length of branches, etc.).

It presents a novel method of simulating the growth of plants which is not entirely based on L-Systems. The main difference is that it takes a less direct approach as rule based generation. The environmental effects are not part of the growth rule

system but a separate subsystem which modifies the result of the growth model. This leads to a dynamically and easily extendable model.

The validation section shows that this novel method can efficiently estimate the future characteristics of a given tree. It was able to give not just quantitative information about the tree, but also a complete three-dimensional model. It is possible to preview the states during the simulation at the end of each year.

Profiling showed that the most time-consuming part of the algorithm is the ray-tracing sub-process. To speed-up this part, a novel GPU based ray tracking algorithm was developed and implemented using the CUDA framework. Benchmarks show that this was slower than the CPU implementation in the case of small (young) trees, but it was significantly faster in the case of large (old) ones.

Unlike the already existing implementations, the presented system can simulate not just one but multiple trees at once (where each of these can be different species). Benchmarks show that the lack of hardware resources should be the main limit for this kind of simulations.

Thanks to the modular design of the application, the set of deforming factors of the simulation can be easily extended. As further plans, authors would like to implement the following modifiers:

- Regionalism by altitude
- Water consumption and need
- Soil quality
- Geotropism

Acknowledgements

The authors would like to say thanks for the help of the Hungarian National Talent Program (NTP-HHTDK-18-0031 and NTP-SZKOLL-18-0016). Sándor Szénási would like to thank EPAM Systems for their financial support. The authors would like to thank NVIDIA Corporation for providing graphics hardware through the CUDA Teaching Center program. The authors also wish to thank the members of the CUDA Teaching Center at Óbuda University for their constructive comments and suggestions. The content of the paper, however, does not necessarily express the views of these companies and persons, the authors take full responsibility for any errors or omissions.

References

[1] D. Stojcsics, Z. Domozi, and A. Molnar, "Air pollution localisation based on uav survey," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2018, pp. 2546–2551.

[2] K. Czakóová *et al.*, "Microworld environment of small language as, living laboratory" for developing educational games and applications," in *Conference*

proceedings of eLearning and Software for Education (eLSE), vol. 1, no. 01, 2017, pp. 285–291.

[3] D. Kiss, "A model of heat exchange and accumulation in small-sized bioreactors during ethanol fermentation," in *2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, 2017, pp. 000 313–000 316.

[4] D. Chamovitz, *What a plant knows: a field guide to the senses*. Scientific American/Farrar, Straus and Giroux, 2012.

[5] J. Weber and J. Penn, "Creation and rendering of realistic trees," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995, pp. 119–128.

[6] P. Horácek, "Introduction to tree statics & static assessment," in *Tree statics and dynamics seminar, interpreting the significance of factors affecting tree structure & health, Westonbirt, UK*, 2003.

[7] K. Onishi, S. Hasuike, Y. Kitamura, and F. Kishino, "Interactive modeling of trees by using growth simulation," in *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, 2003, pp. 66–72.

[8] P. L. Jaworski, "Using simulations and artificial life algorithms to grow elements of construction," Ph.D. dissertation, UCL (University College London), 2006.

[9] A. Lindenmayer, "Mathematical models for cellular interactions in development i. filaments with one-sided inputs," *Journal of Theoretical Biology*, vol. 18, no. 3, pp. 280 – 299, 1968. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022519368900799

[10] C. Jirasek and P. Prusinkiewicz, "A biomechanical model of branch shape in plants," in *Proceedings of the western computer graphics symposium, Whistler, Canada*. Citeseer, 1998, pp. 23–26.

[11] C. A. Jirasek, *A biomechanical model of branch shape in plants expressed using L-systems*. Calgary, 2000.

[12] J.-F. Barczi, H. Rey, Y. Caraglio, P. de Reffye, D. Barthélémy, Q. X. Dong, and T. Fourcaud, "A structural whole-plant simulator based on botanical knowledge and designed to host external functional models," *Annals of Botany*, vol. 101, pp. 1125–1138, 2008.

[13] F. Tian-shuanga, I. Yi-binga, and S. Dong-xu, "Tree modeling and dynamics simulation," *Physics Procedia*, vol. 33, pp. 1710–1716, 2012.

[14] J. C. Hart, B. Baker, and J. Michaelraj, "Structural simulation of tree growth and response," *The Visual Computer*, vol. 19, no. 2, pp. 151–163, 2003.

[15] (2018) speedtree main page. [Online]. Available: https://store.speedtree.com/

[16] G. Molnár, Z. Szűts, and K. Biró, "Use of augmented reality in learning," *Acta Polytechnica Hungarica*, vol. 15, no. 5, 2018.

[17] H. W. Jensen, J. Arvo, P. Dutre, A. Keller, A. Owen, M. Pharr, and P. Shirley, "Monte carlo ray tracing," in *ACM SIGGRAPH*, 2003, pp. 27–31.

[18] G. Kertész and Z. Vámossy, "Current challenges in multi-view computer vision," in *2015 IEEE 10th Jubilee International Symposium on Applied Computational Intelligence and Informatics*. IEEE, 2015, pp. 237–241.

[19] P. Szántó and B. Fehér, "Hierarchical histogram-based median filter for gpus," *Acta Polytechnica Hungarica*, vol. 15, no. 2, 2018.

[20] V. Marković and Z. Konjović, "A contribution to software development quality management," *Acta Polytechnica Hungarica*, vol. 14, no. 8, 2017.