


## Article

# Integration of a System Dynamics Model and 3D Tree Rendering—VISmaF Part II: Model Development, Results and Potential Agronomic Applications

Mariano Crimaldi , Fabrizio Cartenì, Giuliano Bonanomi and Francesco Giannino 

Department of Agricultural Sciences, University of Naples Federico II, 80055 Portici, Italy

\* Correspondence: mariano.crimaldi@unina.it

**Abstract:** Biological–mathematical models of trees can be exploited for a wide range of agronomic applications including crop management, visualization of ecosystem changes over time, in-field phenotyping, crop load effects, testing of plant functions, biomechanics, and many others. Some models propose a 3D output of tree that, in addition to having functionality to visualize the result, offers an additional tool for the evaluation of some parameters of the model itself (interception and amount of light, temperature, obstacles, physical competition between multiple trees). The present study introduces a biological–mathematical model of tree growth with a 3D output of its structure in a realtime 3D rendering environment (*Unity*®). Thanks to the virtual environment created in *Unity*®, it was possible to obtain variable environmental parameters (amount of light, temperature) used as inputs to the mathematical simulation of growth. The model is based on ordinary differential equations (ODEs) that compute the growth of each single internode in length (primary growth) and width (secondary growth) and the accumulation of growth inhibitors regulating the seasonal cyclicity of the tree. Virtual experiments were conducted varying environmental conditions (amount of light and temperature), and the species-specific characteristics of the simulated tree (number of buds, branching angle). The results have been analyzed showing also how the model can be adapted for the creation of different tree species and discussing the potential agronomic applications of model.

**Keywords:** FSPM; ordinary differential equations; system dynamics; 3D tree rendering; *Unity*®



**Citation:** Crimaldi, M.; Cartenì, F.; Bonanomi, G.; Giannino, F. Integration of a System Dynamics Model and 3D Tree Rendering—VISmaF Part II: Model Development, Results and Potential Agronomic Applications. *Agronomy* **2023**, *13*, 218. <https://doi.org/10.3390/agronomy13010218>

Academic Editor: Yanbo Huang

Received: 14 December 2022

Revised: 3 January 2023

Accepted: 5 January 2023

Published: 11 January 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Biological–mathematical tree models can be used for a wide range of agronomic applications including crop management, visualization of ecosystem changes over time, field phenotyping, crop loading effects, plant function testing, biomechanics, and many others [1–3]. Some models propose a 3D tree output that, in addition to result visualization, offers an additional tool for the evaluation of some parameters of the model itself (light interception and quantity, temperature, obstacles, and physical competition between multiple trees) [4]. Among the various biological–mathematical models, those called Functional–Structural Plant Models (FSPMs) in the last two decades have been developed by scientists to explore and integrate plant structure and its biological/functional processes [5]. In conjunction with this, the development of 3D rendering techniques has achieved a level of complexity and realism [6] that allows the 3D structure to be used not only as a simple visual output of FSPMs but also to characterize plant phenotypes [7] or as feedback to evaluate and calculate light partitioning [8]. The 3D plants can be part of several modeling systems, being major components of many digital representations of landscapes or natural sceneries [9]. A 3D tree can be used in several scientific applications, such as synthetic forestry [10], generic digital representation of the real world [11], flow dynamics [12–16] and, as mentioned, in botany to determine light distribution and physiological parameters [17,18]. In addition, geometric modeling of plants allows researchers to visually validate biological processes, such as how plants interact with light and the

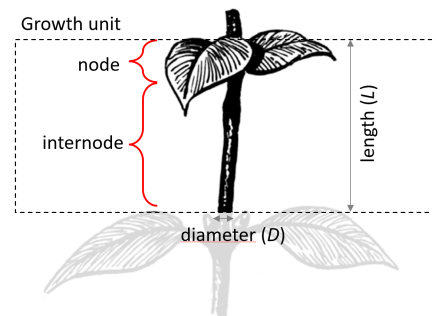
environment. An extensive review of different modeling approaches of tree generation using FSPM that have 3D output and the state-of-the-art of current modeling approaches is provided in the companion paper, part I of this research, by the authors [19]. Among the biological–mathematical models that provide 3D output, a biological–mathematical model of a tree with a 3D output of its structure in a real-time 3D rendering (*Unity*©) environment is presented in this paper as expansion and further development of a proof of concept presented by authors in Part I [19]. The goal of this work is to create a bridge between biological physiological process and mathematical system dynamics model to a real-time 3D rendering engine (*Unity*©). To create this connection, the authors used the object-oriented programming paradigm to make the various modules of the model independent from the 3D rendering while being functionally connected to each other. The biological basis of the model is based on a system of growth units (defined as a set of nodes and internodes), photosynthetic potential, growth rate of growth units, xylematic water transport efficiency and internal accumulation of growth inhibiting metabolites. The biological model is explained in more detail in Sections 2.1 and 2.2. The model proposed is also able to generate different architectural types of plants (globe shaped trees, columnar trees, etc.) depending on environmental variables (temperature, amount of light), species-specific genetic characteristics (number of buds, branching angle, maximum tree height, etc.) as well as biotic factors such as competition with other plants. How environmental inputs were modeled and how they affect the growth model are explained in Section 2.3. How the biological/mathematical model was linked to the 3D rendering environment is explained in Section 2.4 with references to the dumb code shown in the Appendix A. Finally, the various setups of different simulations are presented in Section 2.5, with the respective results shown in Section 3. In conclusion, in Section 4, the authors discuss the results, the planned future updates and enhancements to the model, the possible practical applications in agronomy and highlight the limitations of the study.

## 2. Materials and Methods

### 2.1. Model Overview

The proposed model is based on a hybrid approach with IBM (Individual-Based Model) elements and ODE elements interacting with each other [20] of the epigeal part of a tree species. The plant is simulated as a set of nodes and internodes whose internal dynamics of each internode are described by ODE, while the production of new internodes and interactions with the external environment are managed by the IBM part. This model (Section 2.2) is linked to *Unity*© [21] rendering engine in order to provide a real-time, procedural 3D output of model calculations and interactions with the environment (specifically, we only considered light and temperature as the main growth driving factors). The link can operate in both ways, i.e., by rendering a tree using parameters from ODE solutions, and/or by considering geometrical 3D data (e.g., the amount of light from the virtual scene) as input to solve the ODEs. *Unity*© has been chosen as 3D rendering engine because it offers a programming flexibility given by the use of scripts in C# language and a series of tools for the management of geometries and their interactions (collisions, physics, hierarchy, etc.). *Unity*© is also a widely used environment for many applications, mainly for the production and development of videogames. Since it was designed as a videogame development environment, it has been necessary to program from scratch some features necessary for the correct functionality of the model, such as a custom shader for the calculation of the virtual environmental light amount, and more importantly, an algorithm for the numerical resolution of the ODEs involved. In the following subsections is shown how the model has been developed from a system of ODE equations simulating growth units (Figure 1), then all transferred in *Unity*©, developing the modules in C# according to the Object-Oriented Programming (OOP) paradigm (Section 2.4). Being a real-time 3D rendering engine, through the development in *Unity*© it has been possible to implement the variation of the virtual environmental parameters during the run-time: it is not necessary to stop the simulation in order to change these parameters and then to start over but it is

possible to modify them in real-time observing instantly how the model adapts to new input parameters and computing the modified tree architecture.



**Figure 1.** Growth unit (node and internode) elements and simulated dimensions (ODE state variables).

## 2.2. Branch Dynamics

The mathematical model representing the growth dynamics of the virtual plant's branches has been developed as a set of ordinary differential equations (ODEs). Each branch is divided into several growth units representing a single node/internode couple (Figure 1). The first two equations represent primary and secondary growth, i.e., the dynamics in time of length  $L$  and width  $D$  of each growth unit (Figure 1).

In general terms, we assume that both processes follow a logistic dynamic whose growth rate is proportional to the photosynthetic potential of the plant which, for simplicity, is considered as a simple function of the two main environmental parameters temperature and light. Primary growth of each internode is limited by two factors, its position on the tree and internal metabolic factors. The first limiting factor is assumed to represent the water transport limitation as the height increases, in other terms internodes which are higher above the ground grow shorter than internodes closer to the ground. Each modelled tree is assumed to have a fixed species-specific maximum height representing the potential xylematic water transport efficiency (e.g., [22]). On the other hand, seasonal cycles of growth arrest and resumption are assumed to be regulated by the internal accumulation of growth inhibiting metabolites within the meristematic tissues [23,24]. Specifically, the primary growth arrest is caused by the internal accumulation of the inhibitors which leads to buds' formation. We then assume a gradual decay of such molecules leading to vegetative buds endodormancy break and the possibility of growth resumption if the environmental conditions are favourable. Restarting the growth, the system will generate one or more branches depending on a species-specific parameter (number of children). Similarly, secondary growth is assumed to start after the cessation of primary growth with a rate proportional to the environmental conditions. The following set of equations represents the biological processes described above.

Specifically, the differential equation modeling the  $i$ -th internode length over time  $L_i$  is defined as follows:

$$\frac{dL_i}{dt} = k_i * \alpha * L_i * \left(1 - \frac{L_i}{L_{max}}\right) \quad (1)$$

where  $k_i$  is a given parameter as in Equation (5) and:

$$\alpha = c_L * envTemp * inputLight \quad (2)$$

with  $c_L$  representing the optimal growth coefficient;  $envTemp$  as temperature given by virtual environment and  $inputLight$  as the amount of light calculated into virtual environment.  $L_{max}$  represents the maximum length of the internode, calculated as:

$$L_{max} = 1 - \frac{\sum L_i}{H_{max}} - 0.2 * \frac{I}{I_{max}} \quad (3)$$

where  $\sum L_i$  is the actual tree height,  $H_{max}$  is the tree potential max height as species-specific parameter,  $I$  is the inhibitor concentration as calculated in Equation (4),  $I_{max}$  is the set maximum value of inhibitor concentration. The differential equation modeling inhibitor concentration is defined as follows:

$$\frac{dI}{dt} = \sum_i c_I * \alpha * L_i * \left(1 - \frac{L_i}{L_{max}}\right) - k_i * d_I * I \quad (4)$$

where  $c_I$  is the optimal inhibitor coefficient,  $k_i$  is a given parameter that changes if the tree is in seasonal stop or not. The parameter  $k_i$  changes its value if the inhibitor concentration reaches the  $I_{max}$  value or the  $I_{min}$  value, making the tree enter and exit season stop, as follows:

$$\begin{cases} I \geq 0.95 * I_{max} & \rightarrow k_i = 1 \rightarrow \text{season stop} \\ I \leq I_{min} & \rightarrow k_i = 0 \rightarrow \text{restart growth} \end{cases} \quad (5)$$

The differential equation modeling secondary growth is defined as follows:

$$\frac{dD_i}{dt} = d_{actual} * D_i * k_i \quad (6)$$

where  $d_{actual}$  is a parameter function of virtual environment temperature calculated as:

$$d_{actual} = c_D * envTemp * inputLight \quad (7)$$

with  $c_D$  as optimal secondary growth coefficient and  $k_i$  is a given parameter as in Equation (5). An overview of all variables involved, their definitions and assigned values are shown in Table 1. This set of differential equations cannot be solved directly in *Unity*© because there is no built-in differential equation solver. A C# script as abstract class has been written, allowing the above differential equations to be solved via the fourth-order Runge–Kutta numerical method. This script has been associated with the internode element for integration of its length and secondary growth, and with the branch element for calculation of the inhibitor concentration. For example, the internode element processes its length at each time step by numerically solving the associated differential equation (Equation (1)) at each time step thanks to the integration script created.

**Table 1.** Variables involved in tree growth ODEs system.

Variable	Definition	Assigned Value
$L$	Length of internode	$0.1(t = 0)$
$I$	Concentration of inhibitor	$0.1(t = 0)$
$D$	Width of internode (secondary growth)	$0.1(t = 0)$
$t$	Simulation time	0
$c_L$	Optimal growth coefficient	0.5
$c_I$	Optimal inhibitor coefficient	0.2
$c_D$	Optimal secondary growth coefficient	0.01
$d_I$	Inhibitor degradation parameter	0.1
$envTemp$	Virtual environmental temperature	as set by user
$inputLight$	Light amount	-
$H_{max}$	Max tree height	species-specific
$I_{max}$	Max inhibitor concentration	1
$I_{min}$	Minimum inhibitor concentration	0.1
$k_i$	Season stop parameter	0 or 1

### 2.3. Environmental Inputs

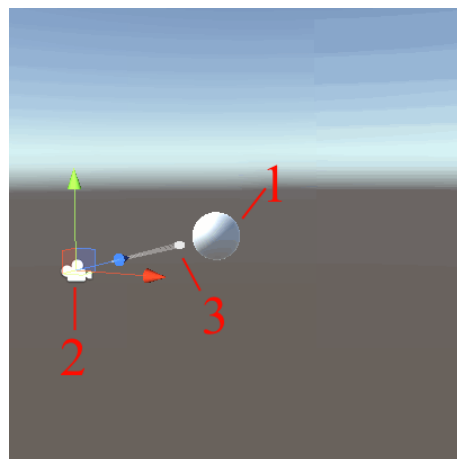
#### 2.3.1. Light Amount Calculation

A key part of the model development is the calculation the amount of light coming from the virtual environment. Several shader solutions have been tested to find the optimal one in terms of computational performance and flexibility. A shader is a program that runs

in the graphics pipeline and tells the computer how to render each pixel. These programs are called shaders because they are often used to control lighting and shading effects [25]. Standard lighting shaders are unable to compute indirect light (such as diffusive light). The shaders tested have been Global Illumination (GI), Radiosity and Ray-tracing, each one showing pros and cons. The Ray-tracing is the most accurate but more computationally heavy, Radiosity and GI are computationally lighter but less accurate and not suitable for the purpose of this work because they failed to calculate diffusive and indirect light in the quality and precision desired for this particular type of application. The alternative solution, evaluating all the solutions previously shown, has been to write a custom shader specialized in calculating the amount of light. A “light meter” sphere *GameObject* (a fundamental object in *Unity*®) is placed in the scene with a camera pointed on the object (Figure 2). The position of the camera changes each frame to cover the entire surface. The camera renders a very narrow texture and records the RGB values into an array. Then, a conversion from RGB values to brightness is made thanks to Equation (8) from ITU-R BT.709 standard [26] as shown in Listing A5 as dumb code in Appendix A. The final value of brightness will be the sum of all arrays calculated at each camera positions.

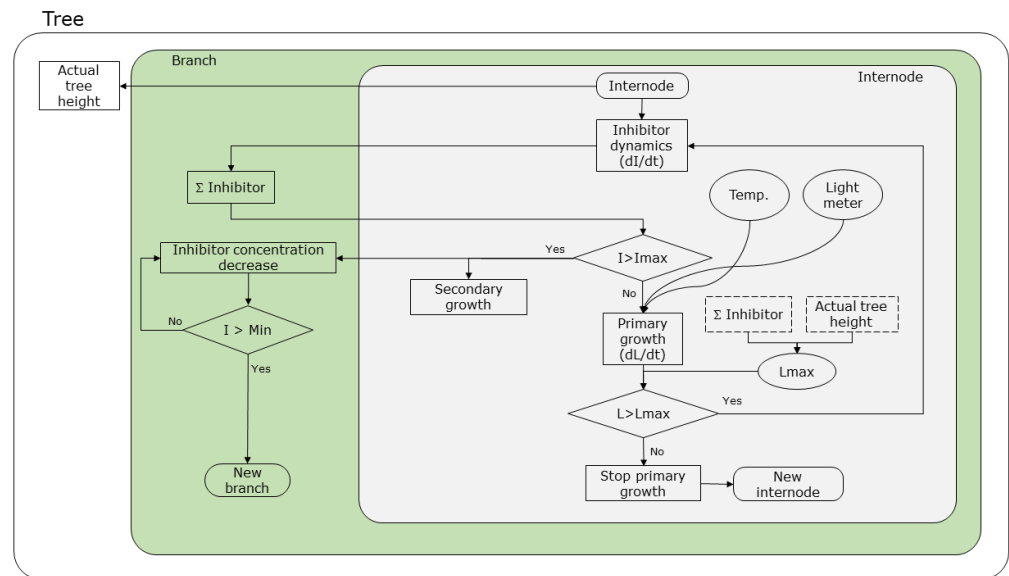
$$\text{Brightness} = 0.2126 * R + 0.7152 * G + 0.0722 * B \quad (8)$$

This way it is possible to calculate direct and indirect light, even if the object is partially in shadow, in real-time and with a little weight on performance (if the number of cameras is kept low).



**Figure 2.** Light meter *GameObject* (1—pictured as a sphere) with texture rendering camera in one position (2). The camera will change its position to calculate an array of RGB values from a very narrow texture (3), then converted to brightness. The light meter is placed at the end of each awake internode in the simulation.

As shown in Equations (1) and (2) as well as in the growing scheme in Figure 3, the light input affects internode growth with a standardized calculated scheme value between 0 and 1. When the internode is growing in full light, with a value close to 1, its growth will be the maximum or very close to the maximum. The light meter has been placed at the top of the growing internode, and then removed whenever the internode stops growing, as it is no longer necessary to calculate the amount of light. By doing so, it has been possible to keep a relatively low number of light meters simultaneously active during growth, managing to have a certain balance between computational performance and precision. In addition, by placing the light meter at the apex of the growing internode, it has been possible to vary the growth angle  $\varphi_{AAV}$  shown in the Section 2.4, which will deviate from the ideal one as a function of the amount of light calculated.



**Figure 3.** Proposed tree growth model scheme and relationships between modules. The blocks in the schematic represent the various functions of the modules, the rhombus blocks represent the function activation conditions (if statements in C# code).

### 2.3.2. Light and Temperature Implementation

The *Unity*© environment allows the use of virtual environmental parameters as model inputs. In the previous paragraph it has been shown how it is possible to calculate the amount of light that is a part of the environmental parameters. *Unity*© allows the programming and modeling of other parameters that can be classified as environmental. As shown in the model diagram in Figure 3, another environmental parameter that has been chosen as an input to the model, in addition to the amount of light, is temperature. Obviously, in a virtual environment it is not possible to record a real temperature but it is possible to model its trend and use the simulated numerical data as model input. In this work, the temperature data is an input that influences the coefficient  $\alpha$  in the growth of the internode (Equations (1) and (2)). In combination with the amount of light, the temperature will affect the amount of internode growth. In *Unity*© is possible to set a function to simulate the non-constant temperature trend (temperature changes during seasonal and daily cycles) as well as for any parameter, value, variable in the model. In this study, the temperature has been managed by a curve that represent the variation of temperature value along a time range: the temperature changes over a period of time from 1 to 365 (as days, simulating a year) starting from the minimum value of 15 (assumed as °C) reaching the peak at 25 (assumed as °C) and then decreasing again. Once the 365th step has been passed, the cycle starts again from 1 simulating another year. The temperature value will be different at each time-step which, in the model proposed, corresponds to a day for each second of simulation (1 s of simulation = 1 day of tree growth). It is possible to create more complex curves to better simulate the real variability of temperature over time. For example, to simulate a real temperature trend over a year, it is possible to set a variation curve given by real yearly data record. In this work, the simple temperature variation curve explained above is used as generic example of temperature variation application. Moreover, this curve function can also be used to manage the variability of other environmental parameters such as the already mentioned ambient light in order to simulate the day/night cycle.

### 2.4. Model Implementation

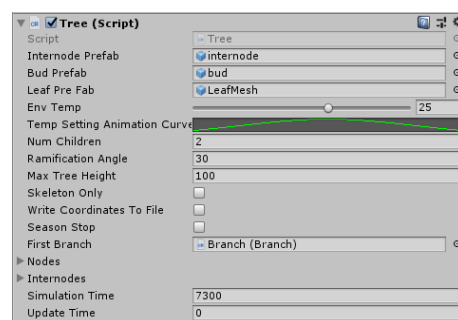
The model shown in Figure 3, a concise explanation of the model developed in *Simile* [27] that translates in stock and flow representation a system of ordinary differential equations that resolve numerically time variable dimensions, describes as flow-chart the separation of working modules and their mutual interactions. As discussed in the work



by Crimaldi et al. [19], *Unity*® has been the choice for a real-time 3D engine having useful features for the purpose. *Unity*® uses the C# language to code scripts associated with geometric elements (*prefabs*) in the working environment. Each element (or module) has a linked C# script that defines its behavior. The C# language complies with the object-oriented programming (OOP) paradigm: each element can inherit all common features, methods, functions and parameters from another. *Unity*® also allows the management of the hierarchy between elements enabling to have elements as father generating children: a situation that is well suited to model a tree where there will be child internodes descending from common father (primary branches from a common main trunk, secondary branches from primary branches, and so on). Moreover, thanks to the use of *Unity*® as a real-time rendering engine, it is possible to change some parameters during run-time: parameters such as the virtual environment temperature, light quantity, direction and characteristics, allowing the modeling of some typical situations of a tree growth like the day/night cycle or the seasonal cycle.

As shown in Figure 3 the tree has a number of parameters that allow the model to run. Some are species-specific, others are process parameters that change step by step in real time as the simulation runs. The species-specific parameters are set at the beginning of the simulation and do not change during the simulation. They are parameters proper of the tree species that is intended to simulate. In the model proposed in this work it has been chosen to simulate as species-specific characteristics the branching angle (defined as the angle of the new branch created in relation to the parent), the number of children (defined as the number of new branches created after the end of the seasonal stop of the tree, Figure 3), the maximum height of the tree (defined as the maximum height to which the tree will tend asymptotically, Figure 3) and the simulation time (defined as how long the simulation will run, 1 s in simulation equals to 1 day of tree life). These parameters have been coded in the C# script shown in Listing A1 in Appendix A as dumb code of *Tree* class.

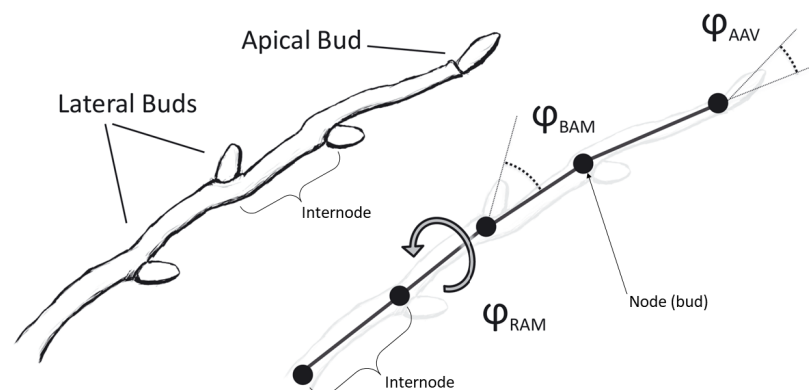
These settings are possible in *Unity*® modifying the variables of the script via the inspector (a built-in *Unity*® editor), Figure 4, or via scripting. According to the object-oriented programming paradigm, the other modules and sub-modules will descend from the *tree* script, inheriting the common characteristics. Each module that inherits from *tree* will have the same characteristics. For example, it will inherit the maximum height of the tree as a parameter. The *tree* module holds also lists of all branches, internodes and buds (nodes) of tree to process the tree structure hierarchy.



**Figure 4.** The species-specific parameters (number of children, ramification angle, max tree height) modifiable via *Unity*® inspector panel. The “simulation time” parameter sets the temporal length of the simulation (1 s = 1 day).

In *Unity*® engine, the whole structure of the tree is based on a hierarchy between the elements described above. Thanks to C# scripts and the object-oriented programming paradigm, all elements are interconnected structurally but also behaviorally. Any single element can affect all other elements and vice-versa. The single elements in *Unity*® that have a shape and a behavior are called *GameObject*. The hierarchy of a tree in the very first phase of growth is simple and evolves in a much more complex form during growth. From

the *GameObject Tree* all the other *GameObject(s)* descend hierarchically, starting from the first bud called *seed* and then moving on to subsequent branches, internodes and buds. As explained above, the *GameObject "end"* is the one positioned in the space where it should be the end of the length of the growing internode calculated step by step by solving the ODE (Equation (1)). By doing so, although it seems an overcomplicated procedure, it is possible to separate and make independent the calculation of the length of the internode (e.g., the position of the *GameObject "end"*) and the generation of the mesh or geometry of the same internode. The *GameObject* in charge of calculating the amount of light from the external virtual environment is the *LightMeter* described in detail in Section 2.3. An example of hierarchy is provided in the Appendix A, Figure A1. The tree has a final structure formed by a series of nodes (lateral and apical buds) and consequential internodes according to a hierarchy (Figure 5). The main trunk is the collection of main internodes father of all subsequent child internodes that have a lower hierarchical scale, up to the apical internodes that are the internodes having lower hierarchical scale. Branches are the entities that contain the entire sequence of nodes and internodes and that manage, thanks to their code scripts, the functions of generating new branches at certain branching angles and the concentration of inhibitor that will determine the seasonal stop of the tree. At the end of the seasonal stop, as shown in Figure 3, new branches are generated according to the number of children set in the species-specific characteristics of the tree and according to the branching angle. The actual branching angle deviates from the ideal branching angle set as a species-specific parameter based on the amount of light calculated from the virtual environment as explained in Section 2.3.1.



**Figure 5.** Nodes-internodes scheme of a branch with branching angles.  $\varphi_{RAM}$  is the bud rotation angle around internode axis;  $\varphi_{BAM}$  is the bud angle to internode axis;  $\varphi_{AAV}$  is the apical bud angle to internode axis.

The *Internode C#* class manages the behavior of internode(s) as explained above. As shown in the dumb code in Listing A2 in Appendix A, the internode can have 3 states: awake, asleep or dead. The three states can be changed thanks to external events or by other modules of the tree (i.e., by branch module at the end of the seasonal stop). The internode only grows when it is in the awake state, is paused when it is in sleep state (it can resume growth), and can no longer grow if it is in the dead state. At tree initiation, the initial numerical conditions of the differential equations for calculating length and width (secondary growth) are set as well as the creation of a *GameObject* (a fundamental object in *Unity*®) "*end*" representing the end point of the internode whose position is the numerical solution of the ODE that calculates the length (see method `public override void Init(Tree t)` in Listing A2). This approach has been chosen because there is no need to create a 3D shape each time and scale it according to the calculated length, but only the position of the end point is calculated, reducing the computational load according to the principle of procedural rendering as proposed by Weber and Penn [28]. More specific code explanation is provided in Appendix A.



The *Branch* C# class manages the behavior of branch(es) as explained before (Equation (4)). As shown in the dumb code in Listing A3 in Appendix A, the branch can have three states: awake, asleep or dead like the internode. The branch calculates the inhibitor concentration numerically resolving the ODE (Equation (4)) setting the numeric initial conditions, then calculates the numeric solution of the ODE at each time step. When the inhibitor concentration reaches its maximum, the tree (and the branches) enters in seasonal stop. When the inhibitor concentration reaches its minimum, the seasonal stop ends and the tree restarts the growth, as well as the calculation of inhibitor concentration. At the end of seasonal stop, new branches are also created as the number of children set. More specific code explanation is provided in Appendix A.

The *Bud* C# class manages the behavior of buds, both apical and lateral ones (Figure 5). As for internodes and branches, buds can have three states: awake, asleep and dead. In this case, as shown in Listing A4 in Appendix A, the script will check at each update if an external event affects the buds, changing their state from asleep to awake or dead and vice versa. If the bud is awakened by an external event, it will produce new branches that will be rotated at certain angle function of light amount. At the end of internode growth, when the internode reaches its maximum length, a method is called to generate a new internode child of the previous one with a rotation angle function of the amount of light ( $\varphi_{AAV}$  in Figure 5). Another important function handled by the *Bud* class is leaf placement. In this work we decided not to model the leaves as an organ of the plant with its own ODEs but only as an object in the 3D environment with its own mesh, orientation and rotation. This is because, during preliminary tests, we found an exponential drop of computational performance as the number of leaves increases. The leaves, therefore, having their own shape and position, will contribute to the calculation of light, influencing tree own shadow and consequently its growth. The parameters of leaves angles and positions are either fixed as hard-coded value or depend on the position of the bud. A particular bud is the one named *seed* in the hierarchy: it is considered as the first bud from which the main trunk grows and generates the first branch as the main trunk. More specific code explanation is provided in Appendix A.

## 2.5. Simulations Setup

In order to test the potential of the model and its flexibility and adaptability to different conditions and parameter settings, experiments have been conducted generating growing trees in the virtual environment created in *Unity*© with different combinations of both species-specific and environmental parameters. Specifically, simulations have been first carried out with fixed environmental parameters: with a constant temperature (set to 25 °C) and amount of light (set to 1, corresponding to a tree growing in full light) and only changing the species-specific parameters such as number of buds and branching angle. Specifically, tests were carried out with:

- number of buds: 2–3–4;
- branching angle: 30°–40°–55°–60°.

Tests with a number of buds equal to 1 have already been done in the preliminary tests showed in the previous companion paper published by authors [19]. Tests made by changing environmental parameters have been done by keeping fixed the number of buds and varying the temperature, the quantity and the direction/origin of the light source. The temperature regimes adopted in these simulations have been as follows:

- fixed temperature at three different values: 15 °C–25 °C–35 °C;
- variable temperature over time as explained in the previous paragraph (Section 2.3.2).

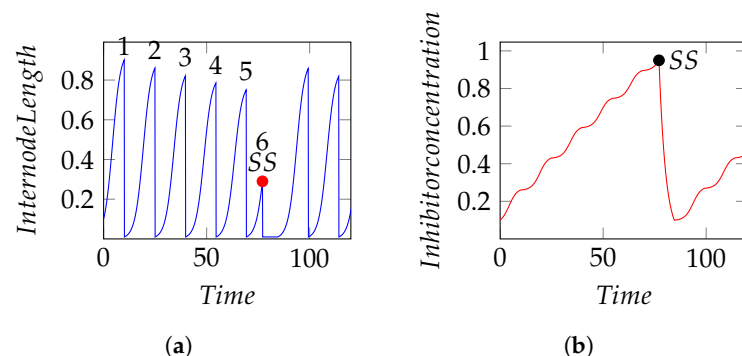
Competition tests were carried out by growing 2 and 4 trees with the same species-specific parameters and temperature at a variable distance (1 m, 2 m). Tree competition is limited to light: no module has been modeled that simulates resources and thus competition for resources (e.g., nutrients, water). This module will be developed in forthcoming work. Finally, tests have been performed by varying species-specific parameters to simulate the

growth of different species of trees; in this work, columnar-shaped trees (such as a *Populus nigra*) and conical-shaped trees (such as a *Picea abies*) have been modeled. All tests were carried out for a growth duration of 5 years (1826.25 s of simulation). In the following section, the results with the different combinations of parameters discussed in the previous paragraph are shown. The results of the simulations in *Unity*© are shown without the presence of the leaves even though the position of the leaves have been computed and they have been considered in the calculation of tree self and mutual shadow, to analyze and evaluate the architecture of the resulting trees with their species-specific characteristics.

### 3. Results

#### 3.1. Species-Specific Parameter Changes—Fixed Environmental Parameters

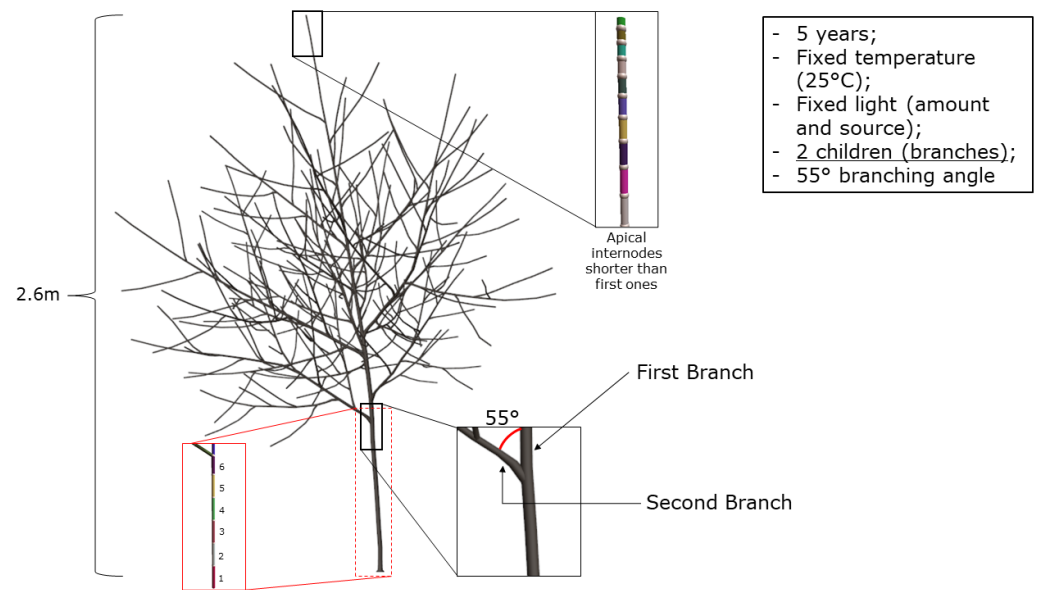
The expected behavior of the branch dynamics, with fixed environmental parameters, is the one shown by the simulations in Figure 6. The tree branch grows by increasing internode length, until it reaches the maximum internode length calculated as in Equations (1) and (3). At the end of growth, the internode stops growing, the apical bud generates a new child internode that will grow to its maximum length. Simultaneously, the inhibitor concentration increases until it reaches its maximum. When the maximum concentration value is reached, according to Equation (4), the tree enters the seasonal stop and secondary growth starts. Inhibitor concentration begins to decrease until it reaches its minimum, the tree starts growing again, secondary growth stops, inhibitor concentration begins to increase again, as does internode growth. At the end of the seasonal stop, new branches of the same number set in the tree species-specific parameters will also be generated. The expected behavior of the simulation is, therefore, to have internodes that decrease their maximum length as they approach the maximum tree height. The result will be that the first internodes (those in the main trunk) will be longer than the apical internodes.



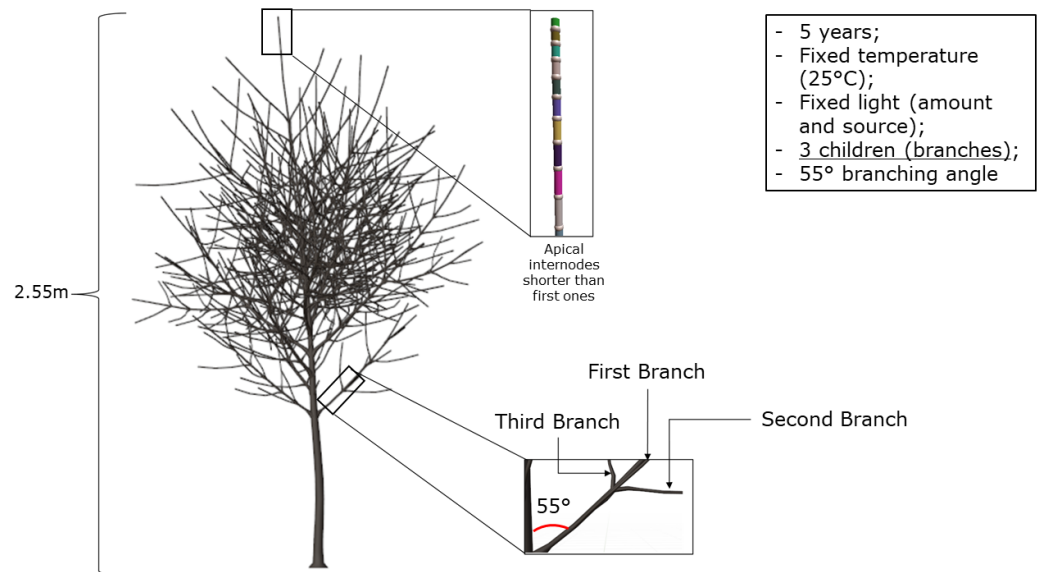
**Figure 6.** Internode length and Inhibitor concentration simulations in *Simile* as expected behaviour with fixed environmental parameters. SS = Season Stop. Numbers are the count of internodes. (a) Internode Length; (b) Inhibitor Concentration.

The result obtained in *Unity*© with fixed environmental parameters, 55° branching angle and two children (branches) is shown in Figure 7. The tree reached after 5 years (1826.25 s) of simulation the height of 2.6 m. It can be noticed that the number of internodes in the main trunk is the same as in Figure 6. It is possible to see the set branching angle and how the apical internodes are shorter than those in the main trunk.

By increasing the number of children, while leaving the environmental parameters constant, the tree appears more dense due to higher new branches generated. The apical internodes remain shorter than those in the main trunk, as can be seen in Figure 8 where the simulation generated a tree 2.55 m in height.



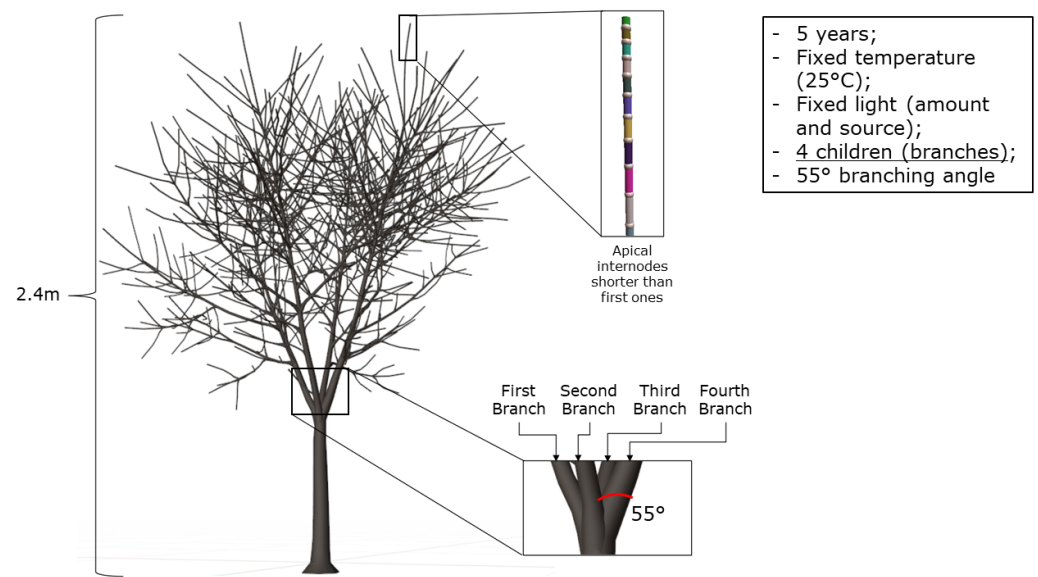
**Figure 7.** Tree growth with fixed temperature (25 °C), fixed light source (top) and amount, 55° branching angle and 2 children branches. The apical internodes are displayed with different colors for showing purpose.



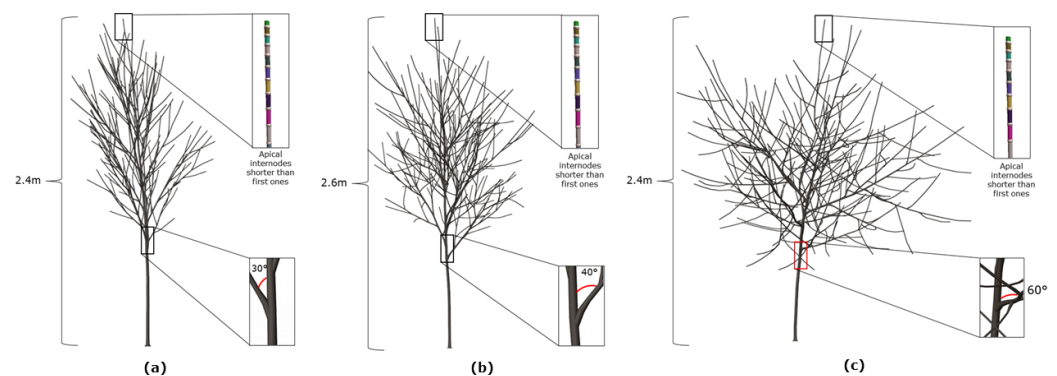
**Figure 8.** Tree growth with fixed temperature (25 °C), fixed light source (top) and amount, 55° branching angle and 3 children branches. The apical internodes are displayed with different colors for showing purpose.

It is possible to notice the same effect generating a tree with 4 children, as shown in Figure 9, where it was generated a tree 2.4 m high.

By changing the branching angle instead, the results obtained are shown in Figure 10. It can be seen that by changing the branching angle from 30° (Figure 10a) to 40° (Figure 10b) and 60° (Figure 10c), the tree widens the canopy and takes up more space. The heights are also different, respectively, 2.4 m for the trees with 30° and 60° branching angles, and 2.6 m for the tree with 40° branching angle.



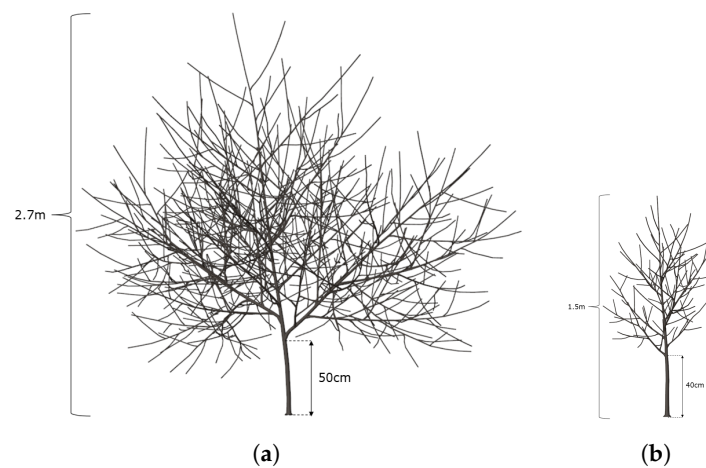
**Figure 9.** Tree growth with fixed temperature (25 °C), fixed light source (top) and amount, 55° branching angle and 4 children branches. The apical internodes are displayed with different colors for showing purpose.



**Figure 10.** Tree growth with fixed temperature (25 °C), fixed light source (top) and amount and 2 children branches. The branching angle showed is (a) 30°, (b) 40° and (c) 60°. The apical internodes are displayed with different colors for showing purpose.

### 3.2. Virtual Environment Parameter Changes

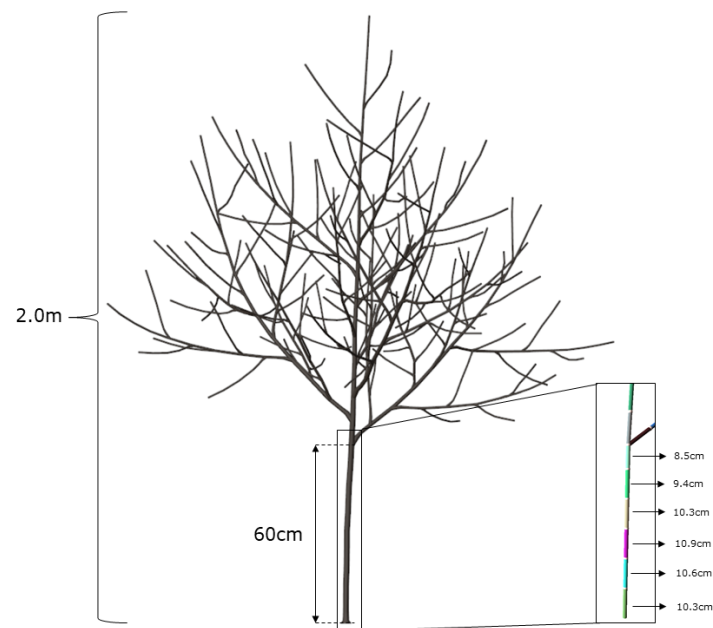
The expected behavior, varying the virtual environmental temperature, is that the behaviors of the internode length and inhibitor concentration curves change. By lowering the temperature, internode growth is slower. The variation in inhibitor concentration also changes, slowing it down as well. At lower temperature, therefore, there will be the effect of seeing a tree grow, for the same simulation time, lower in height and less branched because it takes longer to reach the value of maximum internode length and maximum inhibitor concentration, which causes the tree to enter seasonal stop late, generating new branches at lower heights. The result of growing a tree at a low temperature is shown in Figure 11b where it also possible to see that the first branching is lower in height than the tree growing at higher temperature in Figure 11a. By contrast, increasing temperature will cause internode length and inhibitor concentration to increase faster, reaching their maximum values sooner. There will be the effect of seeing a tree, for the same simulation time, higher and more branched because it reaches the value of maximum internode length and maximum inhibitor concentration quickly, which causes the tree to enter seasonal stop soon, generating at higher heights the new branches, as shown in Figure 11a.



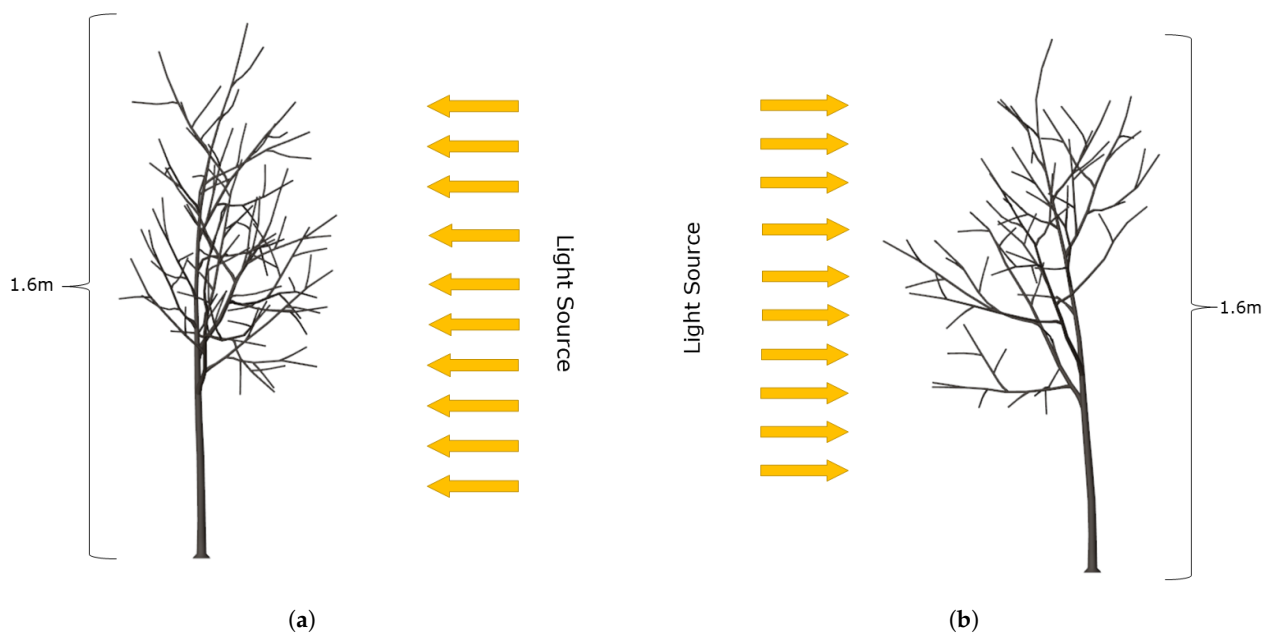
**Figure 11.** Tree growth with changing environmental temperature, fixed light source (top) and amount, 55° branching angle and 2 children branches. (a) 35 °C environment temperature; (b) 15 °C environment temperature.

As described in Section 2.3.2, the temperature can also be changed dynamically over time, following a variation curve. In this study, the variation curve manages the temperature variation, starting at second 1 (day 1) from a value of 15 °C, reaching a maximum at second 182 (day 182) of 25 °C and then dropping back down to a value of 15 °C at second 365 (day 365) and then start again from day 1. The variation curve can have values and curvature at will, in this work it was set up this way to demonstrate how the growth function, and thus the resulting tree, adapts to changes in virtual environmental parameters in real time. Due to low starting temperature in the early days (seconds of simulations), the internode length growth is slower. As the temperature dynamically returns to the “standard” value of 25 °C, the growing rate approaches the ideal conditions. After reaching the changing temperature curve apex, the temperature itself starts to decrease again slowing down the growth of the internode. Inhibitor concentration follows the same trend: slow increase in the first part, ideal situation in the middle part, slowed down in the second part where the temperature has a descending trend. The resulting tree with dynamically time changing temperature is shown in Figure 12.

It is possible to notice in Figure 12 how the dynamic variation of the temperature makes the internodes grow in different lengths depending on the period in which the simulation is along the curve of variation of the temperature. In addition, the tree is shorter in height (2.0 m) than those grown under ideal conditions as growth is slowed at the periods when the temperature deviates from the ideal, increasing (first phase of the temperature change curve) and decreasing (second phase of the temperature change curve). As shown in the model diagram in Figure 3 and elaborated in the internode growth Equations (1) and (2), the amount of virtual ambient light affects the rate of internode growth. The effect, as the amount of light changes, is quite similar to that showed with changing temperature. In addition, in this case, if the amount of light is reduced, the growth rate of the internode will also be reduced, i.e., for the same simulation time the internode will grow more slowly and less. In addition to the amount of light, as explained in Section 2.4, the direction of the light source will also affect growth. Specifically, internodes will grow by changing the angle  $\varphi_{AAV}$  (apical bud angle to internode axis, Figure 5) to turn towards the light source. This effect can be clearly seen by placing a light source to the side, creating parts of the tree in full light and others in shadow, as shown in Figure 13. The shadowed parts of the tree have internodes with a low growth rate due to low light, no growth at all if the light amount is equal to 0. The brighter parts of the tree have higher internode growth rates and they are directed towards the light source. The not equally distributed light over the entire tree as in the previous results will also lead to a lower overall tree growth rate, producing a smaller tree than ideal conditions.



**Figure 12.** Tree growth with dynamically time changing environmental temperature, fixed light source (top) and amount, 55° branching angle and 2 children branches.



**Figure 13.** Tree growth with 55° branching angle, 2 children branches and different light source directions. (a) Tree growth with light source from right; (b) Tree growth with light source from left.

### 3.3. Competition between Neighboring Trees

The model's ability to adapt to the amount and location of the light source allows it to simulate competing groups of nearby trees. Figure 14 shows the growth of two trees 1 m away from each other while Figure 15 shows two trees 2 m away from each other. Figure 16 shows four trees grown in two parallel rows at 1 m between trees and between rows. Lastly, Figure 17 shows the growth of four trees arranged in 2 rows at a distance of 2 m between trees and 2 m between rows.

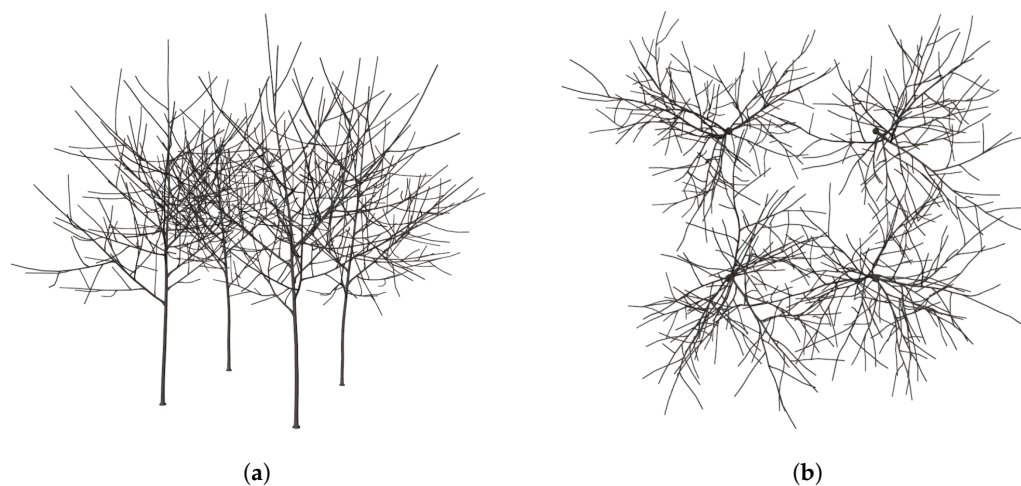




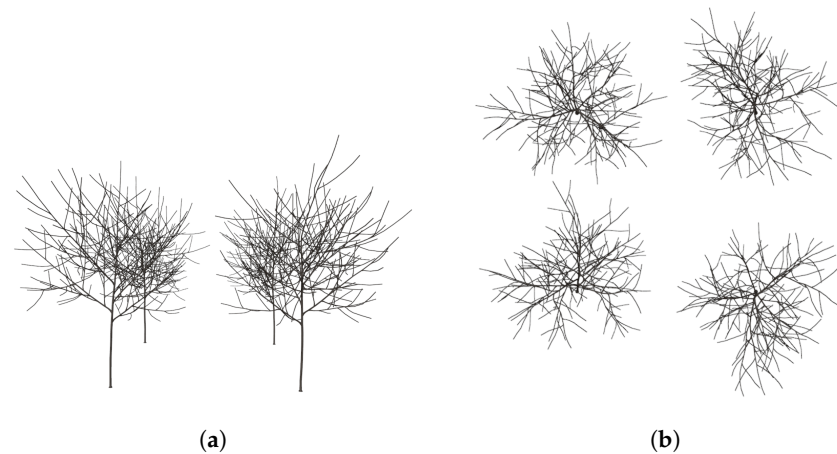
**Figure 14.** Group of 2 trees growing at 1 m distance. (a) Side view of 2 trees growing at 1 m distance; (b) Top view of 2 trees growing at 1 m distance.



**Figure 15.** Group of 2 trees growing at 2 m distance. (a) Side view of 2 trees growing at 2 m distance; (b) Top view of 2 trees growing at 2 m distance.



**Figure 16.** Group of 4 trees growing at 1 m distance. (a) Side view of 4 trees growing at 1 m distance; (b) Top view of 4 trees growing at 1 m distance.

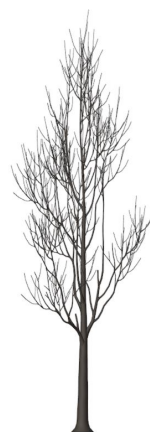


**Figure 17.** Group of 4 trees growing at 2 m distance. (a) Side view of 4 trees growing at 2 m distance; (b) Top view of 4 trees growing at 2 m distance.

The greatest effects of competition for light can be seen in simulations with groups of trees at shorter distances from each other (1 m) where the proximity of trees increases mutual shadow, decreasing the rate of growth of internodes closer to the nearby tree and therefore more in shade. At higher distances the amount of mutual shadow between trees is much less, causing neighboring trees to grow in near ideal conditions.

### 3.4. Model Adaptation to Different Tree Shapes

The tree models shown in the previous subsections all represent cluttered globe-shaped trees. By changing the model parameters as described in the previous sections, it is possible to model different types of tree shapes. In the examples proposed in Figure 18, it is possible to see how by setting a branching angle of  $45^\circ$  and a number of children equal to 2, it has been possible to recreate a tree with a columnar shape, like a tree of the species *Populus nigra* (Figure 18a, with a comparison with a real photo of *Populus nigra* in Figure 18b). Figure 18c shows a tree with a conical shape, with a number of children equal to 5 and a branching angle of  $90^\circ$  for the primary branches (those directly generated by the main trunk) and  $55^\circ$  for the secondary and tertiary branches (those generated from the primary or subsequent branches). In the case shown in Figure 18c, it is possible to compare this type of tree with a real photo of a conifer with the same characteristics, such as a *Picea abies* in Figure 18d.

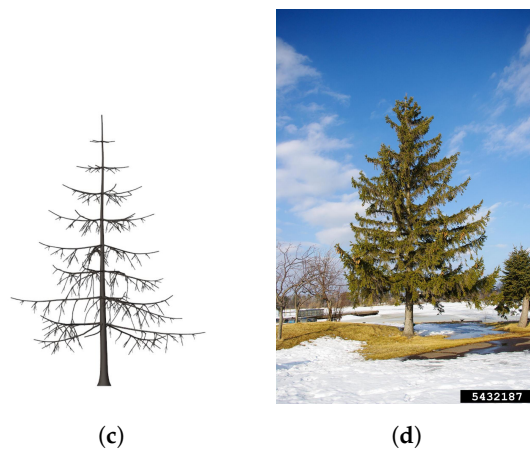


(a)



(b)

**Figure 18.** Cont.



**Figure 18.** Visual comparison for different shape trees. (a,b) are columnar shaped trees; (c,d) are conical shaped trees. (a) Columnar shaped tree with branching angle equal to  $45^\circ$  and number of children set equal to 2; (b) Real photo of a *Populus nigra* for comparison. Photo courtesy of T. Davis Sydnor, The Ohio State University, Bugwood.org; (c) Conical shaped tree with branching angle of  $90^\circ$  for the primary branches,  $55^\circ$  for the secondary and tertiary branches and number of children equal to 5; (d) Real photo of a *Picea abies* for comparison. Photo courtesy of Judy Slater, Bugwood.org.

#### 4. Discussion

The results presented in Section 3 show how the model is adaptable to different specifications, species-specific parameters, and environmental conditions. The flexibility given by the OOP paradigm allows a separation of modules (and their code) while remaining interconnected at the functional level. It is possible to change at any time any ODE that rules a specific module without having to modify the 3D rendering part nor affecting other parts of the code. Any change made can be immediately displayed in the generated output structure. In addition to ODEs, the results show that changes are also possible to the parameters that regulate growth, showing different outputs depending on the modifications. Modifying species-specific parameters, it is possible to model potentially any tree species with different numbers of children, different maximum heights, and different branching angles. Being a real-time rendering environment and procedurally computing the solutions of the ODEs at each frame and rendering the results, it is also possible to modify the inputs of virtual environment parameters such as temperature and light. For the temperature, the parameter is only numerical not being related to a physical component of the virtual environment and its variation modifies the amount of growth of the internode and therefore of the tree, as well as the concentration of inhibitor. As for light, in the virtual environment its value and position are linked to a physical component (light source). Changing the amount, position, and direction of the light source, the internode growth rate changes, modifying the direction of growth by simulating phototropism. Despite being a compromise between the precision of *ray-tracing* (but too heavy computationally) and the computational lightness of the GI (but not very accurate), the system adopted for the calculation of the amount of light turns out to be satisfactory for the aim of the work proving to be adaptable to the desired growth conditions. Although not shown in the results, the presence of the leaves contributes to the calculation of the self and mutual shadow and influences the amount of light measured. The leaves themselves were not modeled as an organ and do not have a specific module, the authors will implement this module in future work. Thanks to the calculation of the amount of light in real time, it has been also possible to model small groups of trees competing with each other for light, showing how the proximity and therefore the shadow generated by another tree, the growth of the internodes (and of the tree) is affected by modified light amount detected. Competition for resources and their implementation is another module, as the one for leaves as organs, that the authors plan to incorporate in future work. Despite some limitations given by design and modeling choices, the authors believe that the present work can be extended and refined in future

research. However, at this stage, the proposed model can already be used in some practical and scientific agronomic applications as suggested in the following section.

#### *Potential Agronomic Applications of the Model*

The possible agronomic applications of the model presented in this work may be various. As shown in previous companion publication [19], the potential of FSPMs with 3D output can be numerous. In the case of the model presented in this work, the authors count on being able to use it in applications to study the effects on the architecture of planting density, to calibrate allometric models, or even for quantitative analysis of branching patterns [29–31]. The model proposed in this paper can be used to estimate the Tree Row Volume (TRV) index to optimize fungicide and insecticide delivery by automated and precision spray systems [32–37]. Another potential use of the presented model, being in real-time, is to implement it in an immersive visualization system to train agricultural operators in the practice of pruning or more simply to evaluate the effects of pruning on tree structure. The authors are already working on a system that allows the “slicing” in real time of the meshes that are part of the tree structure to visualize the pruning operation and its effects. In addition, the virtual pruned tree model can be used as a tool to optimize breeding form or in a scenario where automatic pruning systems are instructed to implement automatic operations [38–43]. The modularity of the proposed system can also be used to implement effects of plant pathologies and visualize their consequences on tree structure and its development in time. In the less agronomic context, the virtual trees generated by the proposed model can be used in the creation of synthetic datasets for the training of artificial intelligence in automatic species recognition [44–46]. More generally, the model presented can be applied in all applications that require a *digital-twin* [47], in this case of a tree, to perform destructive testing or operations without materially compromising a real tree.

## 5. Conclusions

The use of FSPM models has been well established in the modeling community for two decades [5,18]. Many models now have 3D output that allows rapid visualization of results as well as being an integral part of the model itself as a provider of input parameters, as described in the review by Crimaldi et al. [19]. The use of a real-time 3D rendering engine allowed the authors to create a link between a mathematical biological model of tree growth and the procedural creation of its architecture. This solution differs from other proposals of integration between models and 3D rendering in the capability of having as input the parameters of the virtual environment in real time (amount of light and temperature), calculating at each variation (at each simulation time step) the solutions to the ODEs that regulate the processes showing immediately the result as a modification of the structure. While showing a flexibility that is well suited to the purpose, thanks to the implementation of interconnected modules adhering to the OOP paradigm specialized in individual components of the tree, the model has some margin for improvement such as the implementation of a module for the modeling of leaves as organs, or the integration of competition for resources in neighboring trees. Further studies are underway by the authors to implement the above features, as well as direct applications of physical changes to the generated structure in the 3D environment such as a pruning operation.

**Author Contributions:** Conceptualization, F.G.; methodology, M.C. and F.G.; investigation, M.C.; model development, F.G., F.C. and M.C.; system implementation, M.C.; resources, M.C.; writing—original draft preparation, M.C.; writing—review and editing, M.C. and F.G.; supervision, F.G.; funding acquisition, F.G.; agronomic applications, G.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported and funded by the P.O.R. Campania FSE 2014/2020 “*Dottorati di ricerca con caratterizzazione industriale*” grant. The work of F. Cartenì was funded by the project AIM1850344 of the AIM (Attraction and International Mobility) Program, financed by the Italian Ministry of Education, University and Research (MIUR).

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

FSPM	Functional-Structural Plant Model
OOP	Object-Oriented Programming
ODE	Ordinary Differential Equation
GI	Global Illumination
IBM	Individual-Based Model
TRV	Tree Row Volume

## Appendix A

### Appendix A.1. Code Listings

In this Appendix all dumb codes mentioned in the main text are shown as listing.

The *Tree* class (Listing A1) sets the main specie-specific parameters of the tree as well as the simulation time. The class stores a list of all Buds, Internodes and Branches.

**Listing A1.** Tree script with species-specific parameters settings: NumChildren is the number of new ramifications at the end of internode growth; RamAngle is the ramification angle as in Figure 5; maxTreeHeight is the maximum tree height set as specie-specific parameter.

```
public class Tree : MonoBehaviour {
    // Number of children
    public int NumChildren = 2;

    // Ramification angle
    public float RamAngle = 30f;

    // Max tree height
    public int maxTreeHeight = 100;

    // Lists of Branches, Internodes and Nodes (buds)
    public List<TreeNode> nodes = new List<TreeNode>();
    public List<Internode> internodes = new List<Internode>();
    public List<Bud> buds = new List<Bud>();

    void Start() {
        // Start tree with species-specific parameters;
    }

    void Update() {
        // Update tree each time step until simulation time has reached;
    }
}
```

The *Internode* (Listing A2) class provides the calculation of internode length, the internode state and the secondary growth. The length of internode, e.g., the position of “end” *GameObject*, is calculated in method `public void UpdateGrowth()` as well as the max internode length (as in Equation (3)) as shown in Listing A2. The *Internode* class takes also as input parameter the light amount (Section 2.3.1) and temperature from virtual environment. The `private IEnumerator UpdateSecGrowth()` is a peculiar class called *coroutine*. The *coroutine* is used in *Unity*® as a method that can pause execution and return control to *Unity*® but then continue where it left off on the following update. This means that the action that takes place within a *coroutine* does not happen within a single update like in



standard methods. The use of *coroutine* is well suited to the computation of secondary growth that must be updated during the season stop of the tree and one step at a time.

**Listing A2.** Internode script with all methods performing length calculation and other variables and parameters.

```
public class Internode : Tree{
    // Internode state
    public enum InternodeState {
        Asleep, Awake, Dead
    }

    private IEnumerator UpdateSecGrowth() {
        // Secondary growth ODE solution calculation via Integrator script
    }

    public override void Init(Tree t) {
        // Create ``end'' object
        // Set internode length initial conditions for ODE integration
        // Set secondary growth initial conditions for ODE integration
    }

    public void InitInternode(Tree t, Branch b) {
        // Create Light Meter instance at internode creation
    }

    public override void InitGrow() {
        // Start internode growth
    }

    public Bud InitBud() {
        // Start and put bud in place
    }

    public void UpdateGrowth() {
        // Calculate internode length at each same step as ODE solution
        // Calculate max internode length in function of actual length and max tree height
        // Take as input the light amount and temperature
        // Create a new bud at the end of growth and destroy light meter instance
    }
}
```

The *Branch* class showed in Listing A3 provides the calculation of inhibitor concentration and controls the tree seasonal stop.

With method `public void BranchInit(Tree t, Bud b)` the class sets the numeric initial conditions, then calculates the numeric solution of the ODE at each time step performing the method `private void InhibitorUpdate()`. This method stops when the inhibitor concentration reaches its maximum and puts the tree (and the branch) in seasonal stop. When the inhibitor concentration reaches its minimum, the method ends the seasonal stop of the tree and restarts inhibitor concentration calculations, as well as re-start growing the tree. While in seasonal stop, a *coroutine* method is called (`private IEnumerator ChangeSeason()`) that waits until the seasonal stop ends, then creates new branches as the number of children set.

The *Bud* class is showed in Listing A4.

Thanks to the update method `public void Update()`, the script will check at each update if an external event affects the buds, changing their state from asleep to awake or dead and vice versa. If the bud is awakened by an external event, it will produce new branches calling the method `public Branch CreateBranch()`.

The method `public Internode CreateInternode()` is called when the internode stops growing having reached its maximum length. In the *Bud* class the leaf position is managed by the `public override void Init(Tree t)` method, which manages the angle and position on the bud itself.

The *Bud* class will call the `public override void InitGrow()` method if the bud is the first (seed) and generate the first branch as the main trunk.



**Listing A3.** Branch script with all methods performing inhibitor concentration calculation and other variables and parameters.

```
public class Branch : Tree {
    public enum BranchState {
        Asleep, Awake, Dead
    }

    public void BranchInit(Tree t, Bud b) {
        // Set branch inhibitor initial conditions for ODE integration
        // Add branch in tree branches list
    }

    private IEnumerator ChangeSeason() {
        while (seasonStop)
        {
            // Stop while season stop
        }
        for (int i = 1; i < tree.NumChildren; i++)
        {
            // Create new branch(es) for every bud in the tree
        }

        // Ends season stop and restart growing
    }

    public override void BranchUpdate() {
        // Update branch(es) and inhibitor concentration
    }

    private void InhibitorUpdate() {
        // Calculate inhibitor concentration at each same step as ODE solution

        if (!seasonStop && (annualInib >= 0.95 * inhibitorIntegrator.IMax))
        {
            // Activate season stop when inhibitor concentration reaches max
        }

        if (seasonStop && (annualInib <= inhibitorIntegrator.IMin))
        {
            // Restart growing when inhibitor concentration reaches min
        }
    }
}
```

The code of *LightMeter* class is showed in Listing A5. This class provides the calculation of light amount via a custom shader that converts an RGB array of values in brightness value.

**Listing A4.** Bud script with all methods performing internode creation, branch creation, leaf placement and other variables and parameters.

```
public class Bud : Tree {
    public enum BudState {
        Asleep, Awake, Dead
    }

    public void Update() {
        if (externalConditionOccured && state == BudState.Asleep)
        {
            // Bud state changes to awake if it was set as asleep
            state = BudState.Awake;
            // Create a new branch from awoken bud
        }
    }

    public Internode CreateInternode() {
        // Create a new internode from bud after the father internode has stopped growing
        var angle = // Function of light amount
        // The new internode created is rotated as the angle function of light amount
        return internode;
    }

    public Branch CreateBranch() {
        // Create the number of new branches set
        var angle = // Function of light amount
        // New branches created are rotated at certain angle of ramification function
        // of light amount
        return branch;
    }

    public override void Init(Tree t) {
        base.Init(t);
        float leafAngle = // Depends on bud position or hard coded

        // Leaf Placing
        if (!geometry && !tree.skeletonOnly)
        {
            // Place leaf prefab on bud at position
        }
    }

    public override void InitGrow() {
        if (Seed || tree.NumChildren == 1)
        {
            // Create first branch (main trunk) from first bud (seed)
        }
    }
}
```

**Listing A5.** Light meter script to calculate brightness from RGB array of values.

```
public class LightMeter : MonoBehaviour
{
    void Start()
    {
        // Define camera positions: front, back, left, right, up, down
    }

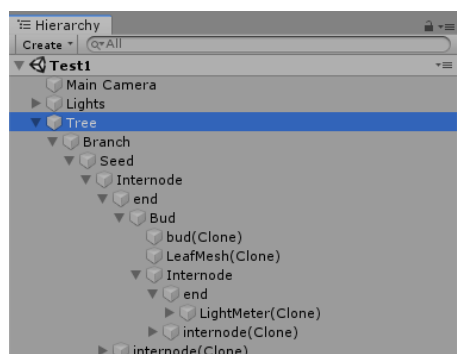
    void Update()
    {
        // Take pixel array values and calculate brightness
    }

    private void GetPix() {
        // Render temporary texture at i-th position of camera

        // RGB to brightness conversion of temporary texture
    }

    public void SumPix() {
        Output = // Final brightness as sum of all i-th temporary textures brightness values
    }
}
```

A simple, at early growth stage, hierarchy showing main tree *GameObjects* is shown in Figure A1.



**Figure A1.** Hierarchy in Unity© showing a simple early tree structure: one branch, two internodes, two buds.

## References

1. Anderson, N.T.; Walsh, K.B.; Wulfsohn, D. Technologies for Forecasting Tree Fruit Load and Harvest Timing—From Ground, Sky and Time. *Agronomy* **2021**, *11*, 1409. [\[CrossRef\]](#)
2. Magistri, F.; Chebrolu, N.; Behley, J.; Stachniss, C. Towards In-Field Phenotyping Exploiting Differentiable Rendering with Self-Consistency Loss. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 13960–13966. [\[CrossRef\]](#)
3. Pradal, C.; Dufour-Kowalski, S.; Boudon, F.; Fournier, C.; Godin, C. OpenAlea: A visual programming and component-based software platform for plant modelling. *Funct. Plant Biol.* **2008**, *35*, 751–760. [\[CrossRef\]](#)
4. Pirk, S.; Stava, O.; Kratt, J.; Said, M.A.M.; Neubert, B.; Mëch, R.; Benes, B.; Deussen, O. Plastic trees: Interactive self-adapting botanical tree models. *ACM Trans. Graph. (TOG)* **2012**, *31*, 1–10. [\[CrossRef\]](#)
5. Louarn, G.; Song, Y. Two decades of functional–structural plant modelling: Now addressing fundamental questions in systems biology and predictive ecology. *Ann. Bot.* **2020**, *126*, 501–509. [\[CrossRef\]](#)
6. Deussen, O.; Lintermann, B. *Digital Design of Nature: Computer Generated Plants and Organics*; Springer Science & Business Media: Heidelberg, Germany, 2006.
7. Zhu, B.; Liu, F.; Xie, Z.; Guo, Y.; Li, B.; Ma, Y. Quantification of light interception within image-based 3-D reconstruction of sole and intercropped canopies over the entire growth season. *Ann. Bot.* **2020**, *126*, 701–712. [\[CrossRef\]](#) [\[PubMed\]](#)
8. Che, Y.; Wang, Q.; Xie, Z.; Zhou, L.; Li, S.; Hui, F.; Wang, X.; Li, B.; Ma, Y. Estimation of maize plant height and leaf area index dynamics using an unmanned aerial vehicle with oblique and nadir photography. *Ann. Bot.* **2020**, *126*, 765–773. [\[CrossRef\]](#) [\[PubMed\]](#)
9. Lama, G.F.C.; Crimaldi, M.; Pasquino, V.; Padulano, R.; Chirico, G.B. Bulk Drag Predictions of Riparian *Arundo donax* Stands through UAV-Acquired Multispectral Images. *Water* **2021**, *13*, 1333. [\[CrossRef\]](#)
10. Makowski, M.; Hädrich, T.; Scheffczyk, J.; Michels, D.L.; Pirk, S.; Pałubicki, W. Synthetic silviculture: Multi-scale modeling of plant ecosystems. *ACM Trans. Graph.* **2019**, *38*, 131. [\[CrossRef\]](#)
11. Magnor, M.A.; Grau, O.; Sorkine-Hornung, O.; Theobalt, C. *Digital Representations of the Real World: How to Capture, Model, and Render Visual Reality*; AK Peters/CRC Press: Boca Raton, FL, USA, 2015.
12. Lama, G.F.C.; Errico, A.; Francalanci, S.; Solari, L.; Preti, F.; Chirico, G.B. Evaluation of Flow Resistance Models Based on Field Experiments in a Partly Vegetated Reclamation Channel. *Geosciences* **2020**, *10*, 47. [\[CrossRef\]](#)
13. Lama, G.F.C.; Crimaldi, M.; De Vivo, A.; Chirico, G.B.; Sarghini, F. Eco-hydrodynamic characterization of vegetated flows derived by UAV-based imagery. In Proceedings of the 2021 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor), Trento-Bolzano, Italy, 3–5 November 2021; pp. 273–278. [\[CrossRef\]](#)
14. Crimaldi, M.; Lama, G.F.C. Impact of Riparian Plants Biomass Assessed by UAV-Acquired Multispectral Images on the Hydrodynamics of Vegetated Streams. In Proceedings of the European Biomass Conference and Exhibition Proceedings, 29th EUBCE, Online, 26–29 April 2021; pp. 1157–1161. [\[CrossRef\]](#)
15. Lama, G.F.C.; Crimaldi, M. Assessing the Role of Gap Fraction on the Leaf Area Index (LAI) Estimations of Riparian Vegetation Based on Fisheye Lenses. In Proceedings of the European Biomass Conference and Exhibition Proceedings, 29th EUBCE, Online, 26–29 April 2021; pp. 1172–1176. [\[CrossRef\]](#)
16. Khan, M.A.; Sharma, N.; Lama, G.F.C.; Hasan, M.; Garg, R.; Busico, G.; Alharbi, R.S. Three-Dimensional Hole Size (3DHS) Approach for Water Flow Turbulence Analysis over Emerging Sand Bars: Flume-Scale Experiments. *Water* **2022**, *14*, 1889. [\[CrossRef\]](#)
17. Millar, A.J.; Urquiza, U.; Freeman, P.L.; Hume, A.; Plotkin, G.D.; Sorokina, O.; Zardilis, A.; Zielinski, T. Practical steps to digital organism models, from laboratory model species to ‘Crops In Silico’. *J. Exp. Bot.* **2019**, *70*, 2403–2418. [\[CrossRef\]](#) [\[PubMed\]](#)

18. Henke, M.; Kurth, W.; Buck-Sorlin, G. FSPM-P: Towards a general functional-structural plant model for robust and comprehensive model development. *Front. Comput. Sci.* **2016**, *10*, 1103–1117. [\[CrossRef\]](#)
19. Crimaldi, M.; Carteni, F.; Giannino, F. VISmaF: Synthetic Tree for Immersive Virtual Visualization in Smart Farming. Part I: Scientific Background Review and Model Proposal. *Agronomy* **2021**, *11*, 2458. [\[CrossRef\]](#)
20. Vincenot, C.E.; Giannino, F.; Rietkerk, M.; Moriya, K.; Mazzoleni, S. Theoretical considerations on the combined use of System Dynamics and individual-based modeling in ecology. *Ecol. Model.* **2011**, *222*, 210–218. [\[CrossRef\]](#)
21. Unity Technologies. *Unity® Game Engine*. 2021. Available online: <https://www.unity3d.com/> (accessed on 7 April 2021).
22. Russo, S.E.; Jenkins, K.L.; Wiser, S.K.; Uriarte, M.; Duncan, R.P.; Coomes, D.A. Interspecific relationships among growth, mortality and xylem traits of woody species from New Zealand. *Funct. Ecol.* **2010**, *24*, 253–262. [\[CrossRef\]](#)
23. Yang, Q.; Gao, Y.; Wu, X.; Moriguchi, T.; Bai, S.; Teng, Y. Bud endodormancy in deciduous fruit trees: Advances and prospects. *Hortic. Res.* **2021**, *8*, 139. [\[CrossRef\]](#) [\[PubMed\]](#)
24. Mazzoleni, S.; Landi, C.; Carteni, F.; de Alteriis, E.; Giannino, F.; Paciello, L.; Parascandola, P. A novel process-based model of microbial growth: Self-inhibition in *Saccharomyces cerevisiae* aerobic fed-batch cultures. *Microb. Cell Factories* **2015**, *14*, 109. [\[CrossRef\]](#)
25. Akenine-Moller, T.; Haines, E.; Hoffman, N. *Real-Time Rendering*; AK Peters/CRC Press: Boca Raton, FL, USA, 2019.
26. ITU-R. BT.709: Parameter Values for the HDTV Standards for Production and International Programme Exchange. 2015. Available online: <https://www.itu.int/rec/R-REC-BT.709-6-201506-1/en> (accessed on 15 April 2021).
27. Simulistics. *Simile*. 2021. Available online: <https://www.simulistics.com/> (accessed on 2 April 2021).
28. Weber, J.; Penn, J. Creation and Rendering of Realistic Trees. In Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '95, Los Angeles, CA, USA, 6–11 August 1995; Association for Computing Machinery: New York, NY, USA, 1995; pp. 119–128. [\[CrossRef\]](#)
29. Leroy, C.; Sabatier, S.; Wahyuni, N.S.; Barczi, J.F.; Dauzat, J.; Laurans, M.; Auclair, D. Virtual trees and light capture: A method for optimizing agroforestry stand design. *Agrofor. Syst.* **2009**, *77*, 37–47. [\[CrossRef\]](#)
30. Xia, N.; Li, A.; Lin, W. Simulation and Quantitative Analysis of Branching Patterns of the Plum Tree. *J. Comput. Sci. Technol. Updat.* **2014**, *1*, 9–18. [\[CrossRef\]](#)
31. Salter, W.T.; Shrestha, A.; Barbour, M.M. Open source 3D phenotyping of chickpea plant architecture across plant development. *Plant Methods* **2021**, *17*, 95. [\[CrossRef\]](#) [\[PubMed\]](#)
32. Sarghini, F.; Visacki, V.; Sedlar, A.; Crimaldi, M.; Cristiano, V.; Vivo, A.D. First measurements of spray deposition obtained from UAV spray application technique. In Proceedings of the 2019 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor), Portici, Italy, 24–26 October 2019; pp. 58–61. [\[CrossRef\]](#)
33. Behlau, F.; Lanza, F.E.; da Silva Scapin, M.; Scandellai, L.H.M.; Silva Junior, G.J. Spray Volume and Rate Based on the Tree Row Volume for a Sustainable Use of Copper in the Control of Citrus Canker. *Plant Dis.* **2021**, *105*, 183–192. [\[CrossRef\]](#)
34. Miguez, M.; Deleón, R.; Vicente, G.; Zoppolo, R. Real Time Tree Row Volume Estimation for Efficient Application of Phytosanitary Products in Fruit Trees. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 26–29 May 2019; pp. 1–3. [\[CrossRef\]](#)
35. Xun, L.; Garcia-Ruiz, F.; Fabregas, F.X.; Gil, E. Pesticide dose based on canopy characteristics in apple trees: Reducing environmental risk by reducing the amount of pesticide while maintaining pest and disease control efficacy. *Sci. Total Environ.* **2022**, *826*, 154204. [\[CrossRef\]](#) [\[PubMed\]](#)
36. Miranda, M.P.; Scapin, M.d.S.; Vizoni, M.C.; Zanardi, O.Z.; Eduardo, W.I.; Volpe, H.X.L. Spray volumes and frequencies of insecticide applications for suppressing *Diaphorina citri* populations in orchards. *Crop Prot.* **2021**, *140*, 105406. [\[CrossRef\]](#)
37. Rathnayake, A.P.; Sahni, R.K.; Khot, L.R.; Hoheisel, G.A.; Zhu, H. Intelligent Sprayer Spray Rates Optimization to Efficiently Apply Chemicals in Modern Apple Orchards. *J. Agric. Saf. Health* **2022**, *65*, 1411–1420. [\[CrossRef\]](#)
38. You, A.; Sukkar, F.; Fitch, R.; Karkee, M.; Davidson, J.R. An Efficient Planning and Control Framework for Pruning Fruit Trees. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 3930–3936. [\[CrossRef\]](#)
39. He, L.; Schupp, J. Sensing and Automation in Pruning of Apple Trees: A Review. *Agronomy* **2018**, *8*, 211. [\[CrossRef\]](#)
40. Di Gennaro, S.F.; Nati, C.; Dainelli, R.; Pastonchi, L.; Berton, A.; Toscano, P.; Matese, A. An Automatic UAV Based Segmentation Approach for Pruning Biomass Estimation in Irregularly Spaced Chestnut Orchards. *Forests* **2020**, *11*, 308. [\[CrossRef\]](#)
41. Zahid, A.; He, L.; Zeng, L.; Choi, D.; Schupp, J.; Heinemann, P. Development of a Robotic End-Effector for Apple Tree Pruning. *Trans. ASABE* **2020**, *63*, 847–856. [\[CrossRef\]](#)
42. Silwal, A.; Yandun, F.; Nellithimaru, A.; Bates, T.; Kantor, G. Bumblebee: A Path Towards Fully Autonomous Robotic Vine Pruning. *arXiv* **2021**. [\[CrossRef\]](#)
43. You, A.; Parayil, N.; Krishna, J.G.; Bhattarai, U.; Sapkota, R.; Ahmed, D.; Whiting, M.; Karkee, M.; Grimm, C.M.; Davidson, J.R. An autonomous robot for pruning modern, planar fruit trees. *arXiv* **2022**. [\[CrossRef\]](#)
44. Kar, A.; Prakash, A.; Liu, M.Y.; Cameracci, E.; Yuan, J.; Rusiniak, M.; Acuna, D.; Torralba, A.; Fidler, S. Meta-Sim: Learning to Generate Synthetic Datasets. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.
45. Lu, Y.; Young, S. A survey of public datasets for computer vision tasks in precision agriculture. *Comput. Electron. Agric.* **2020**, *178*, 105760. [\[CrossRef\]](#)

46. Lu, Y.; Chen, D.; Olaniyi, E.; Huang, Y. Generative adversarial networks (GANs) for image augmentation in agriculture: A systematic review. *Comput. Electron. Agric.* **2022**, *200*, 107208. [[CrossRef](#)]
47. Pylianidis, C.; Osinga, S.; Athanasiadis, I.N. Introducing digital twins to agriculture. *Comput. Electron. Agric.* **2021**, *184*, 105942. [[CrossRef](#)]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.