

# Procedural Generation of Strawberry Plants



UNIVERSITY OF  
LINCOLN

Christopher Page

School of Computer Science

College of Science

University of Lincoln

Submitted in partial satisfaction of the requirements for the  
Degree of Msc by Research  
in Computer Science

*Supervisor:* Dr Grzegorz Cielniak

*Secondary Supervisor:* Dr Petra Bosilj

January 8, 2023

# Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Contributions . . . . .	7
1.2	Overview . . . . .	7
<b>2</b>	<b>Related Work</b>	<b>9</b>
2.1	Digital Models for Agriculture . . . . .	9
2.2	Procedural Generation of Plants . . . . .	10
2.3	Plant Modeling for Phenotyping . . . . .	11
2.4	Comparison to the state of the art . . . . .	12
<b>3</b>	<b>Strawberry Plant Structure</b>	<b>13</b>
3.1	Primary crown development . . . . .	14
3.2	Inflorescence Growth . . . . .	14
3.3	Stolon/runner Development . . . . .	15
3.4	Leaf Development . . . . .	15
3.5	Petiole Length . . . . .	16
3.6	Tropisms . . . . .	16
3.7	Growth Degree Days . . . . .	18
3.8	Summary . . . . .	18
<b>4</b>	<b>Software Frameworks for Plant Modeling</b>	<b>20</b>
4.1	Plant Modelling Frameworks . . . . .	20
4.1.1	SideFX: Houdini . . . . .	21
4.1.2	SpeedTree . . . . .	21
4.1.3	Lpy . . . . .	22
4.1.4	L-studio . . . . .	23
4.2	Framework comparison . . . . .	23
4.2.1	Example L-system Model . . . . .	23
4.2.2	Fruit and leaf generation . . . . .	25
4.3	Summary . . . . .	26
<b>5</b>	<b>Strawberry Plant Model</b>	<b>27</b>
5.1	Plant Components . . . . .	27
5.1.1	Primary Crown . . . . .	27
5.1.2	Leaf and Petiole . . . . .	28
5.1.3	Fruit . . . . .	29
5.1.4	Visual Appearance . . . . .	29
5.2	Growth Functions and Modeling . . . . .	31
5.2.1	Stochastic L-system . . . . .	32
5.2.2	Gravitropism . . . . .	34
5.2.3	Phototropism . . . . .	36
5.2.4	Collision Detection and Avoidance . . . . .	36
5.2.5	Growth . . . . .	37
<b>6</b>	<b>Evaluation of the Plant Model</b>	<b>39</b>
6.1	Data gathering . . . . .	39
6.1.1	Leaf Area . . . . .	40
6.1.2	Petiole Length . . . . .	41
6.1.3	Inflorescence . . . . .	41

6.2	Model Parameter Tuning . . . . .	42
6.2.1	Leaf Area . . . . .	43
6.2.2	Petiole Length . . . . .	44
6.2.3	Inflorescence . . . . .	47
6.3	Evaluation of the Functional Components . . . . .	48
6.3.1	Plant Growth . . . . .	51
6.4	Evaluation of the Implementation . . . . .	51
<b>7</b>	<b>Conclusions and Future Work</b>	<b>53</b>
7.1	Model Limitations . . . . .	53
7.2	Lpy framework suitability . . . . .	54
7.3	Future work . . . . .	54
<b>8</b>	<b>Appendix</b>	<b>56</b>

## List of Figures

1	Image of the procedurally generated strawberry plants . . . . .	6
2	Key strawberry plant components as per [1]. . . . .	13
3	Guttridge’s diagram of a strawberry crown recreated as a 3D digital model [2]. . . . .	13
4	The primary and branching crowns grown in a potted plant during observational studies undertaken in this project. . . . .	14
5	Example of a inflorescence with developing flowers and fruit in a potted plant during observational studies undertaken in this project. . . . .	15
6	Example of a strawberry plant trifoliolate leaf from the observed plants . . . . .	16
7	Example the petiole growing from the primary crown seen in the observed plants. . . . .	16
8	Weight of the developing fruit compared to the flower in the observed plants. . . . .	17
9	Example of the petiole bending towards the light during growth. . . . .	17
10	Different variety of strawberries and their base temperature from studies [3, 4, 5]. . . . .	18
11	The User interface for Arbaro software used to create trees. . . . .	20
12	A basic strawberry plant model made in SideFX: Houdini using the L-system framework. . . . .	21
13	A procedural tree generated in SpeedTree. . . . .	22
14	A strawberry plant generated in Lpy. . . . .	22
15	From the left Lstudio, SideFX and Lpy output. . . . .	23
16	The same rule set in different frameworks starting with Lstudio top, Lpy middle and Houdini SideFX bottom. . . . .	24
17	A branching structure model with additional visualisation objects (spheres) generated in Lpy (left) and SideFX: Houdini (right). . . . .	26
18	Code for the primary crown written in L-py. . . . .	27
19	Functions to grow crown and to create a spiral around the crown for leaves and fruit to grow. . . . .	27
20	Petiole and leaf generation component. . . . .	28
21	Strawberry generation and weight component . . . . .	29
22	Lpy colour map and the options to edit the colours. . . . .	30
23	Field of strawberry plants all individually randomly generated . . . . .	30
24	Editable parameters to affect the growth of the plant. . . . .	31
25	Original L-system starting with 6 iterations on the left then 7 and 8 . . . . .	31
26	The original L-system model (centre), at an angle of 10° (left) and at 40° angle (right). . . . .	32
27	Using stochastic L-system in Lpy with the original output on the left . . . . .	32
28	Adding random length and angle to the rules. . . . .	33
29	Using the “/” with a random value of 0-360 to create a 3D structure. . . . .	34
30	Tropism implemented with elasticity in L-py on the left and no gravity on the right. . . . .	35
31	Normal gravity with assumed weight on the left and Dry weight added on the right. . . . .	35
32	Basic phototropism of the plants bending towards the light. . . . .	36
33	Random Values for leaf generation causing clipping on the left image. Uniform generation with slight randomness reduces clipping and adds evenness. . . . .	37
34	Method for calculating Growth Degree Days (GDD). . . . .	37
35	Comparison of days passed 30, 60 and 90 days at a constant temperature of 15°C and 7°C. Starting with 30 days on the right. . . . .	38
36	The two growing environments used for data collection: a row at the Riseholme strawberry farm (left) and a homegrown strawberry setup (right) . . . . .	39
37	Measuring the upper lobe length of a strawberry plant leaf. . . . .	40

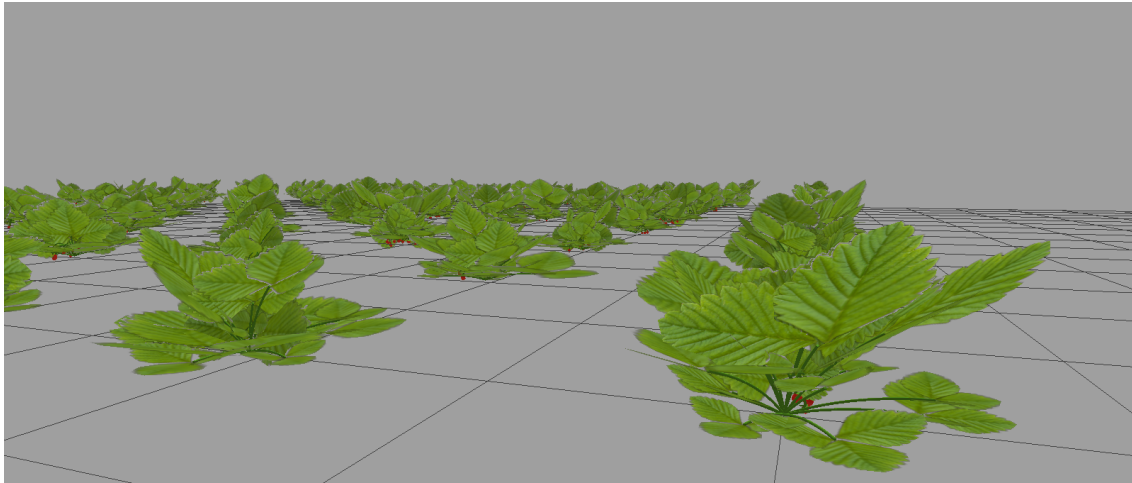


38	Distribution of the leaf area measurements taken at the Riseholme farm (left) and at home (right) measured in cm. . . . .	40
39	Measuring petiole with a ruler. . . . .	41
40	Distribution of the petiole length measurements taken at the Riseholme farm (left) and at home (right) farm measured in cm. . . . .	41
41	Inflorescence from a real plant. . . . .	42
42	Distribution of inflorescence measurements taken from the Riseholme farm measured in occurrences per stem. . . . .	42
43	Leaf area distribution models fitted to the data from the Riseholme farm (left) and home indoor plants (right). . . . .	44
44	Petiole length distribution models fitted to the data from the Riseholme farm (left) and home indoor plants (right). . . . .	45
45	Inflorescence distribution models fitted to the data from the Riseholme farm. . .	47
46	Effects of gravity of different strengths on the generated plant models when placed on a flat surface. . . . .	49
47	Effects of gravity of different strengths on the generated plant models when placed on a 45-degree angle surface. . . . .	49
48	Effects of light on the generated plants when placed on a flat surface. . . . .	50
49	Effects of light on the generated plants when placed on a 45-degree angle surface. .	50
50	Stages of plant development generated at every month for a period of six months with Earth gravity. . . . .	51
51	Generated strawberry plant with the improvements based on best fit for Riseholme distribution(left) and Home grown distribution(right) . . . . .	51

# Abstract

3D models of strawberry plants can be useful for simulations and predicting crop growth in certain environments. Simulating plant growth is a useful tool for phenotyping, predicting crop growth, and creating virtual environments. We have developed a strawberry plant model using an L-system framework called Lpy. The visual simulation is dependent on temperature and days of the month to determine its Growth Degree Days (GDD). Changing these values will affect the growth and stage of the plants' development. Data was gathered from outdoor and indoor-grown plants to tune model parameters such as leaf area, petiole length, and inflorescences. The results showed that L-systems make for an ideal approach for generating strawberry plants but require further work to improve their accuracy by gathering more data from real strawberry plants.

# 1 Introduction



**Figure 1:** *Image of the procedurally generated strawberry plants*

Simulating the growth of plants is an important functionality that finds use in many applications. This includes generating realistically looking plants or virtual environments [6] for use in media, film, and video games [7]. Simulated plants are also an important tool for biologists and plant scientists for studying the effects of external factors on the growth of the plants or for use in breeding programmes [8]. The simulated plants are also becoming increasingly important in creating realistic simulations for robotics applications as these enable pre-training of perception and control components prior to their deployment in the field. Some examples include the use of simulation for predicting the fruit behaviors for robotic picking applications [9], precision weeding [10], and safe navigation [11].

Strawberries are one of the most popular fruits in the world and are grown in many different areas due to the species adaptability [12]. This makes it all the more important to be able to simulate strawberry growth to help improve research and development in that field. For strawberries, however, there are no scientific models available to use - the lack of data and the highly dynamic structure of the plant makes it difficult to model and synthetically generate as compared to more rigid plants like trees or barley [13]. Strawberry plants have been studied and crossbred for many years with investigations including temperature, daylight hours, fruit yield, and their growth pattern. The temperature greatly influences how much the plant will grow and how many leaves it will develop which can be used to predict leaf growth [4]. The structure of a strawberry plant has been also been documented to showcase how the plant develops its components such as leaves, runners, and branching crowns which are used to develop the growth models of the plant [2].

The challenge with modeling plants is their complexity and uniqueness such as for example strawberries having the seeds of the fruit growing on the outside. Different varieties of plants are characterized by different sizes, and colours, and environmental factors like temperature or sunlight affect their growth in a unique way too [14]. To produce a model with high accuracy many of these factors are needed which makes the whole process very complex especially if done by hand. For simulation, a typically large number of individual plants with certain variations are needed which makes the manual approach to modelling completely prohibitive. Visually realistic modeling of strawberry plants is difficult because of the non-trivial appearance of its different parts including colours, shape, texture, and additional factors such as fine hair or imperfections. Surface imperfections, in particular, are difficult to implement as they can appear random but without them, a model can look unrealistically smooth [15].

Procedural Content Generation (PCG) techniques, can help to mitigate some of these challenges by allowing for the quick and large generation of unique plants, and using the data gathered from the real plants can help create more accurate models. Procedural generation is scalable and modifiable by adding the parameters that change the generated output such as the number of plants grown or values of environmental factors like temperature. Using real images of strawberry plants can also help to add some photo realism to the model. Using textures of real leaves and fruit is an easy method for adding realism compared to complex dynamically generated textures or relying on uniform colour.

For simulating and visualising the growth of a plant, one of the most common frameworks is the parametric L-system [16], a parallel rewriting system using grammar. Parameters are used to determine the structure and function of the plant determining how the plant will grow and can be extended to be affected by the environment. The framework has been used to generate models of plants such as barley [13] or trees from image inputs [17]. The L-systems framework is versatile and has been also used for other applications such as architecture [18]. Currently, there are no accurate procedurally generated strawberry plants available to use. The plant models that are available are of different plants and they may not be publicly available. The models need to be suitable for procedural generation allowing for changeable parameters and enhancements.

## 1.1 Contributions

The aim of this work was to develop a procedurally generated 3D model of a strawberry plant using the L-system framework and evaluate its visual and functional accuracy. To this end, the following contributions and outcomes were achieved:

- A working simulation framework, developed in Lpy, for procedurally generated 3D models of strawberry plants including their different components (e.g. leaves, fruit, crowns) and functionality (tropisms and growth).
- An observational study of 4 home grown strawberry plants involving their growing over the course of a single season under artificial light conditions and visual observation using time lapse techniques. The purpose of the study was to learn the appearance, structure, growth patterns and to perform initial data collection of the key plant traits including leaf area, petiole length and inflorescence.
- Tuning of the critical model parameters using trait measurements (i.e. leaf area, petiole length and inflorescence) data collected from the home grown plants and plants grown in real industrial polytunnels.
- The assessment of the suitability of the L-systems as a programming framework for procedural generation of the strawberry plants.
- Evaluation of the functionality of the simulator and visual fidelity of the generated 3D models.

## 1.2 Overview

The thesis will introduce first the related work on the importance of modeling strawberry plants and plants in general for robotics, simulations, and phenotyping as well as look at the current research being done in procedural content generation and the development of models of plants. The structure of a strawberry plant with its components, reproduction, and growth will be covered next so that it can be incorporated into a simulated model. The thesis explores also the various software frameworks for plant modeling and their comparison to identify the most suitable implementation framework followed by an overview of L-systems and their functionality.

The methodology sections cover the different aspects of strawberry plant modeling including structure and function. The evaluation described the collected datasets and the evaluation of the model parameters and the visual outputs from the simulator. Lastly, the thesis concludes with discussions on project outcomes, model strengths, and limitations leading to future work needed to improve the model further.

## 2 Related Work

This section presents the related work from three different areas. This includes digital models and computer vision for agriculture and how bringing synthetic data and digital models in can help improve research. Procedural generation of plants covers the current research and methods of procedural generation in video games and within scientific research. Lastly, plant modeling for phenotyping is covered which examines unique aspects of plants and the environmental factors that need to be considered when procedurally generating them in order to maintain the model's accuracy.

### 2.1 Digital Models for Agriculture

Computer vision is a discipline investigating how to gain a high-level understanding of videos or digital images. In agriculture, it has been used in the automated inspection of plants and fruits to replace manual inspection done by humans [19] with various degrees of effectiveness [20]. An example is a strawberry detection system that uses three characteristics of a strawberry to grade them: shape, size, and colour. The working process involved using a conveyor belt for the strawberries to be carried into the machine which would take photos of the strawberries to get graded at a constant speed [21]. Other fruits such as dates and citrus fruits have also been used for inspection [22, 23]. The benefit of using computer vision for the inspection of crops is the reduction of production cost, and improve accuracy as human perception is not always accurate enough at spotting errors in fruit [24]. The difficulty of fruit detection is the illumination, low resolution and sometimes the occlusion from other fruit can be heard to have an efficient fruit detection [25]. Another computer vision uses are weed detection. Weeds are often a problem when growing crops requiring farmers to invest significant resources for their removal. Weeds have distinct features which allow for their detection by computer vision algorithms such as colour, texture, and shape. These are required by traditional machine learning methods like random forest or Support Vector Machine [26] which are computationally efficient allowing for real-time use and automation [27, 28]. Other uses include phenotyping to understand how each crop is growing [29, 30].

All these computer vision techniques currently require equipment, crops, land, and datasets to conduct the study. It can also be destructive and labor-intensive to use real data unlike synthetic [31]. Using 3D models as synthetic data for computer vision could be useful for phenotyping, inspection, and crop/weed discrimination. Computer vision has been used to create 3D models for use in phenotyping [32, 33] or for investigating drought stress [34]. Creating synthetic data for use in computer vision can provide accurate results in combination with real data [35, 36]. Synthetic data can create datasets larger than real data providing more input for AI to train on [37]. Creating digital scenes with synthetic data is also a viable way to train AI in randomised scenarios as seen in the study creating gardens [38] and Sim4cv, a photo-realistic simulator for computer vision [39]. The simulated realistic models of environments can also help with building new agricultural robots as the cost and skill of building a robot and its components is complex and difficult. Robotic simulations are a useful method for developing and testing code and robot designs without the need to build a working robot. Gazebo is a popular robot simulator used in research and industry [40].

Using a simulation can reduce the cost and time required to conduct research in the aforementioned areas as equipment and crops are not required as they are being simulated. The disadvantage of using a simulation however is that it can be a poor representation of real-life especially if the data is inaccurate such as with physics engines where the accuracy may not be good enough for research [41]. The growth of plants is influenced by physics, such as gravity [42], but also many other different factors which can greatly affect how it grows and whether it will produce fruit or flowers. Aspects like less soil moisture can stunt or slow the growth

of a plant [43]. The amount of sunlight a plant can get will affect the growth [44] as well as more global factors such as climate change causing an increase in carbon concentration [45]. These are all factors that need to be considered when creating realistic simulated plant models and environments. Some of these have been implemented for crops like wheat to evaluate their growth [46] or crop response to factors such as soil nutrition, water, retention, and tillage [47].

Procedural Content Generation (PCG) is a method for creating data algorithmically instead of manually creating the data or using real data. Procedurally generated models were shown to improve robotic development by providing simulated scenarios accurate to real-life to lower costs and increase productivity. Crops In Silico uses multi-scale models to create an integrative platform to potentially increase crop yield, and sustainability and increase future food security [48]. The annotation of data is often expensive as well and requires manual labor to do while procedural generation can mitigate that [49, 50]. Machine learning methods using real data can create over-fitting as data provided may only represent a specific scenario - using procedurally randomised scenarios can lead to improved training of AI techniques in applications such as a robot traversing through various fields of crops or for use in autonomous driving [51, 52]. Procedural content can also be used to test a robot that has already been trained to find weaknesses before being deployed in real environments [53].

While procedural content generation has been shown to help with machine learning a lot of the current approaches have been done in controlled lab settings and are still far away from applications in real diverse environments [51]. Digital 3D models still have certain drawbacks as they can be a poor representation of real life due to errors in the rendering pipeline, photorealism, framerate drops, and the lighting [54]. The hardware required to generate large scenes may not be sufficient to assure sufficient quality and might require trade-offs affecting the final outcome [55]. While procedural generation has shown great potential, it is complex, difficult, and requires time to develop, which is especially challenging in applications such as agriculture requiring accurate plant models.

## 2.2 Procedural Generation of Plants

Procedurally generated content is an attractive alternative for use in research as it provides a way of creating data quickly and at a low cost. It is a popular method in video games and has been used for many years in titles such as *The Elder Scrolls II: Daggerfall* which used procedural generation to create the world and dungeons [56]. PCG can create an almost infinite environment with randomised variation with minimal input from an artist [57]. A large focus of these studies, however, is efficiency and real-time performance designed for games where scientific accuracy is not required. An example is a procedural forest that caches the data of the branches in order to save memory [58]. A recurring problem within procedural content generation is that it takes away the control of the artists and can produce undesired results [59]. A handmade piece of work from an artist which has been made intentionally allows for a tailored experience for the user with fewer glitches [60, 61]. The popular video game *Minecraft* has numerous problems with buildings being generated on water rather than on land which can create surprising visual appearances [62]. A large body of work in PCG focuses on generating the entire environments and placement of items rather than on individual objects as these are less important in large scenes [63, 64]. Well-researched PCG algorithms include Perlin noise and are well documented but designed for creating textures, height maps, and placement of objects [65] which has limited application in creating individual objects.

There are various methods for procedurally generating plants and crops such as using L-systems, space colonization algorithms, point cloud data, or image-based reconstruction. L-system is a type of formal grammar that consists of letters from the alphabet that represents a function for a production rule. An initial character known as the axiom is used to start the L-system [16]. Space colonization operates by simulating competition for space between growing veins, originally made to create leaf venation [66] but has also been expanded for tree generation

used in combination [67]. More recently, with the improvements in camera technology, 3d point clouds can also be used to generate tree structures using the data gathered. TreePartNet is a neural network that reconstructs trees using data from point clouds which are obtained from scanning real trees [68]. For image reconstruction, Disney Research used images of bushes to synthesize dense foliage [69]. An image is taken of a bush to capture its structure of it and an image of a leaf is taken to get the shape of the leaf for reconstruction. This approach can be quite limiting as it requires real-world images to do so which can make it difficult to reconstruct more exotic plants which may require difficult conditions to grow in certain countries such as the case with the US state of Minnesota where the sunlight is short and the intensity is not strong during winter [70]. L-systems is one of the most popular methods for procedural generation of plants like trees [71] as it has been applied, modified, and documented extensively over the course of the past decade and is still used in research with modern methods like deep learning [72]. It has also been used to experiment with ways of procedurally generating other content than plants, for example in architecture for generating buildings [73]. Space colonization and point cloud-based techniques are not as widely applied as L-systems in procedural generation because they are newer methods with fewer articles and can be hardware limited such as requiring scanning equipment for generating point clouds. They have been used it comes to procedural generation of plants but they are often used in conjunction with L-systems such as with point clouds to generate fine branches [74] or 3D reconstruction of fruit trees which used space colonization, L-systems and point clouds to build the geometric model of the tree [75].

Many authors have made use of L-systems in agriculture to procedurally generate crops in conjunction with their research [76, 77, 78]. Crops like tomatoes can be procedurally generated with an L-system because of their relatively simple straight stem and predictable growth [79]. In general, the original L-systems are rather predictable and not accurate to the way plants grow since it does not take into account environmental effects such as light, gravity, and nutrients. Point cloud does take the environment into account as the data can be gathered during the growth of the plant which can be used for plant phenotyping [80, 81]. Space Colonization grows within a space following points and does not take into account environmental factors [67]. L-systems have been expanded to include external factors to see how this may affect a plant's growth or fruit production over time. One example is to use hierarchical instantiation for radiosity to simulate accurate lighting and energy transfer to the surface [82]. This can allow the plant to react to the light and cause it to bend towards the light over time. Another example is using ray-tracing, a different rendering technique, to estimate plant photosynthesis rate under natural and artificial lights [83]. Ray-tracing is capable of diffusing light and therefore can be used to see how plants react within structures such as greenhouses [84].

One of the challenges of simulating plant growth procedurally is their dynamic nature and appearance which changes over time such as growing new buds and parts which are difficult to model for frameworks like L-system. The plant structure continues to change during its growth which has been a problem with recreations in 3D models for examination. Forward-backward analysis can be used to predict the budding or bifurcating process with 4D point cloud data. 4D point clouds are a relatively new way to create 3D models. Most commonly 3D laser scanners and light detection ranging are used to scan objects in the real world. A 4D point represents the spacial coordinates and colour information. The forward-backward analysis is used to predict where a new budding will take place which is compared to the 4D data set [69].

## 2.3 Plant Modeling for Phenotyping

Phenotyping is the process of determining and measuring an organism's observable traits. It is used to look for traits and to predict traits of a plant during growth which is useful in breeding and examining the plants' performance [85]. Being able to procedurally generate a plant and even predict how well it has performed based on its traits could be very useful for breeders to determine how effective a particular species is. Attempts at combining evolution and mutations



with L-systems have been done before by implementing various effects requiring the plant to adapt [86, 87, 88]. The resulting models, however, have drawbacks and do not represent real plants but instead random plants that can evolve based on some parameters. They also can show primitive shapes rather than accurate-looking plants that grow fruit [89]. Point clouds have been used for plant phenotyping as they can be used to create data for various plant organs by using 3D laser scans [90, 91]. Space colonization is not used in phenotyping and is more used for generating realistic-looking plant structures [67]. Various environmental factors can affect a plant's phenotype and change over a period of time and generations. Nutrients in the soil are one factor that can affect this [92]. Ecological modeling procedurally generated juniper and chamise shrubs with the L-system to use as live fuel modelling [93]. Water is another important aspect of plant growth and dryer and hotter seasons will affect the way crops grow and change. Using plant litter, dead plant material such as leaves that have fallen to the ground can help retain the moisture in plants [94].

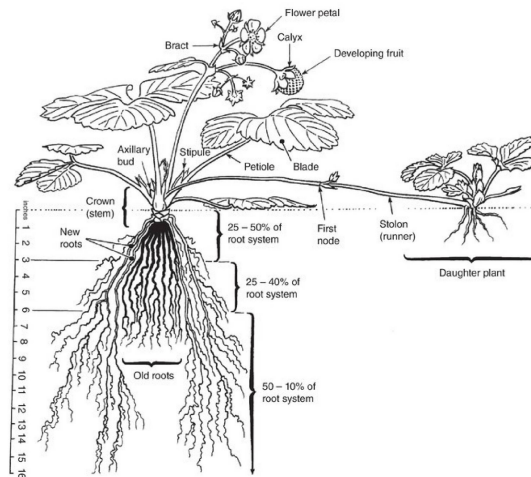
Various software and toolkits have been made to generate 3D plant structures. An example is PlantCAD, a C++ geometric toolkit to allow for easy generation of plant organs like leaves, fruits, branches, and petioles [95]. The VICA plant model is another example that creates a generic functional-structural plant model specifically for barley [13]. The model is formulated as hierarchically structured plant objects. Photosynthesis, nitrogen, and carbon are taken into account for the growth of the Barley to produce more accurate results. Both of these examples use the L-system to generate their plant organs. Another study generated maize with bi-directional communication with external modules to enhance the model further [96]. Generating models without these factors would not be accurate and differ from a real plant as many environmental aspects would be missing. It is also important to consider the unique aspects of plant phenotype and how they may interact with the environment.

## 2.4 Comparison to the state of the art

The work in this research has taken several aspects used in L-systems and combined them to create a procedurally generated strawberry plant within Lpy. Firstly, there are no scientific models of strawberry plants available to use creating a gap in research. Other plants such as tomatoes and wheat have been used to procedurally create models before in detail [13, 79, 97]. Following a similar style of research by incorporating dry matter, gravity, and to a degree sunlight most of which is seen in the tomato dry matter study [97]. Many aspects of the presented work rely on information from the original book about L-systems [16] for example by the way the system randomizes by changing the rules during its generation. Rather than being randomised completely various environmental effects change the way the plant grows instead. We have incorporated temperature using Growth Degree Days and time which affects the growth over time which is not a common feature in L-system due to its often static nature [16].

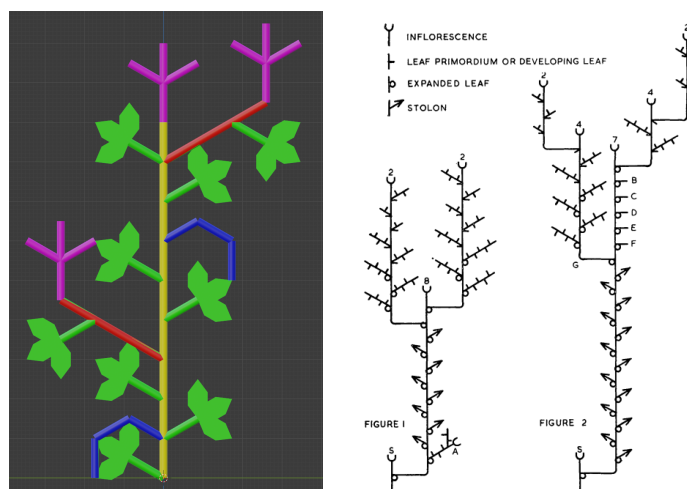
### 3 Strawberry Plant Structure

This section introduces the structure of a strawberry plant by looking into its components including the stem, leaves, flowers, crown, inflorescence and stolons/runners but also leaf development, petiole length and photo- and gravi- tropisms affecting its growth.



**Figure 2:** *Key strawberry plant components as per [1].*

Strawberries are a part of the genus *Fragaria*, a type of flowering rose plant (see Fig. 2). The key component is the primary crown, which is the main stem of the plant, where leaves and other crowns known as branching crowns can grow off of it. The leaves grow spirally around the crown and are typically pinnate (i.e. leaflets arranged on either side of the stem) or trifoliate (i.e. a leaf divided into three leaflets). Clusters of flowers also grow off the crown, known as inflorescence. Runners/stolons can also grow off of the crown which is thin stem that grows along the ground eventually growing roots and its own crown creating a separate plant. The first model of the strawberry plant was developed by Guttridge [2] together with further investigation into its growth and cultivation based on observational study of cultivated strawberry plants (see Fig. 3).



**Figure 3:** *Guttridge's diagram of a strawberry crown recreated as a 3D digital model [2].*

### 3.1 Primary crown development

The crown of the strawberry plant is small having an average diameter size of 6.6mm for bare root, 8.6mm for plug plant transplant during planting, 13.2mm for bare root and 15.2mm for plug plant transplant recorded after seven months later [98]. More branching crowns will appear during its growth, further increasing the number of leaves the plant can grow [98]. Larger crown diameters correlate with more leaves growing and the leaf primordium [99]. Branching crowns are able to develop consistently along the main axis. They are often formed when the primary crown is slowing down in growth which often occurs in late summer or mid-fall. After resting in winter, the growth in spring can also induce the growth of branching crowns. This also depends on the weather and cultural conditions [100].



**Figure 4:** *The primary and branching crowns grown in a potted plant during observational studies undertaken in this project.*

### 3.2 Inflorescence Growth

Inflorescence is simply a cluster of flowers arranged on a stem. Strawberry inflorescence vary slightly in number of branching structures that bear flowers per cultivated species. Over 48 different varieties produce an average of 1.6 crowns, 2.4 inferences and 23 flowers. For example, the variety 'Howard 25' produced the most inflorescences of 4.3 and produced the most flowers at 50.3 whilst the 'Lupton' variety produced the least amount of flowers of only 6.0 in a study presented in [101]. The temperature which the strawberries are grown can effect the size and growth rate of the fruit: higher temperatures result in smaller strawberries being produced [102].



**Figure 5:** *Example of a inflorescence with developing flowers and fruit in a potted plant during observational studies undertaken in this project.*

### 3.3 Stolon/runner Development

The runner/stolon is an auxiliary shoot which supports the plant until it is independent with its own roots grown. The daughter plant can be cut off from the mother plant once it is self sufficient and no longer needs to share resources [103]. The lifespan of an individual strawberry plant is about three years before it begins to lose vigor. Its daughter plants developed from the runners are exact clones which will not lose their vigour as compared to the primary plant. Over the years, the plant will continue to grow in size and produce more fruit in a linear or sigmoid pattern based on the season and plant part such as the flowers and all fruit growth follows a sigmoid pattern while while the number of leaves grows linearly [104]. In the same study the Leaf size and dry weight will go up with an increase in leaf surface area. The leaves take up the most dry matter of 50% with the roots taking up 25% and the flowers taking the remaining 25%.

### 3.4 Leaf Development

The crown of a strawberry plant has a number of leaves at different stages of its growth. The leaves are protected by stipules, A small leaf like appendage that grows to a leaf. Within a closing casing of the previous emerged leaf are five to six growing leaves which is consistent throughout the year. A Plastochron defined as the time interval between two successive recurring events during plant growth [105]. A plastochron index is used to measure the leaf primordium which is the cells that will grow a new leaf. Plastochron is able to measure the current developmental status of each leaf and the age of the shoot. Some leaves may be older in time but may be in a different morphological stage such as healing, growing [106]. Plastochron requires the plants to be genetically uniform and to be grown in controlled conditions and in vegetative growth. One plastochron is during the morphological change of the growing tip of the plant shoot known as the shoot apex. When a new leaf primordium is formed the growth continues and the shoot apex widens. A slight buldge grows from a leaf primordium known as a Leaf buttresses are made on the side of the apical meristem. The leaf buttress is the base of a leaf and a leaf primordium grows upward from the leaf buttress. Strawberry plants leaves have uniformity as the leaves grow around the primary crown like a stem with the sixth leaf being on top of the first [1].



**Figure 6:** *Example of a strawberry plant trifoliate leaf from the observed plants*

### 3.5 Petiole Length

The petiole is a stalk that joins a leaf from the stem. Strawberries have a very short crown but much longer petiole emerging from it. The length of the petiole grows similar to a sigmoid curve when treated to long days over 10 h starting from 5 cm and peaking at 25 cm over 8 weeks. When compared to short days the petiole length was greatly reduced only peaking at 15 cm [107]. An older study from Guttridge examined petiole length under far red light and high irradiance with some of the leaves reaching 30 cm in length [108].



**Figure 7:** *Example the petiole growing from the primary crown seen in the observed plants.*

### 3.6 Tropisms

Gravity and light greatly influence plant growth which has been well documented in Darwin's *The power of movement in plants* [109]. Gravity dictates the shoots to grow upwards while for the roots it dictates their downward growth into the ground [110]. Gravity also limits how high plants can grow causing difficulties of getting nutrients up the plant top and collapsing under its own weight [111]. The average weight of a strawberry fruit varies depending on the cultivar and growing conditions. A larger fruit could weigh 10 g while a smaller fruit can weigh less than 5 g [112]. This often causes the fruit to lie on the ground due to lack of strength in the

stem to keep the weight up. This can be seen in Fig. 8 where the fruit is laying on the grown while the flower is still upright because of the weight difference.



**Figure 8:** *Weight of the developing fruit compared to the flower in the observed plants.*

Phototropism is the growth response from light causing it to bend in a similar way that gravity does. A plant is continuously trying to get the maximum light while also trying to manage the structural stress [113]. Most plants look for light for photosynthesis which is sensed by blue light photoreceptors [114]. Fruit production benefit from longer light exposure during the day known as a photoperiod. The longest days of the year occur in the summer when fruit production is occurring. For strawberry plants, longer exposure to light tends to improve flower and fruit development compared to shorter periods of receiving light [115].



**Figure 9:** *Example of the petiole bending towards the light during growth.*

In flowering plants, the so called etiolation occurs if the plant receives partial or no light for a prolonged growth period. This is characterised by long weak stems, smaller and pale leaves. When exposed to sufficient and continuous light, de-etiolation occurs strengthening the stems and greening the leaves [116]. As strawberry plants are a flowering plant they can be etiolated during its growth, if they are not exposed to enough sunlight. Strawberries grown under polytunnels or in shaded areas experience an effect on etiolation. The colour of light also affects etiolation such as using red lights when growing under grow lamps [117].



### 3.7 Growth Degree Days

Growth Degree Days (GDD) estimates the growth and development of plants during its growing season:

$$GDD = \frac{T_{max} + T_{min}}{2} - T_{base} \quad (1)$$

This is done by recording the highest( $T_{max}$ ) and lowest( $T_{min}$ ) temperature of the day and acquiring a plants base( $T_{base}$ ) temperature which is the lowest temperature the plant will be capable of growth development. Temperatures above 30° are ignored as most plants do not grow faster above that temperature. The estimated base temperature of strawberry plants is 7° but this can depend on cultivar. For example, *Royal Sovereign* stops growth development below 5° and its fastest growth development was at 24° [5]. Some are capable of growing at even lower conditions such as 3c and are grown in colder climates [3]. Strawberries tend to develop a new leaf every  $63 \pm 3$  GDDs [4].

Cultivar.	Base Temperature
Festival	7c
Sugarbaby	7c
Rubygem	7c
Fortuna	7c
Winter Dawn	7c
Royal Sovereign	5c
Senga Sengana	3c
Chambly	3c
Clery	3c
Darselect	3c
Delia	3c
Harmonie	3c
Matis	3c
Sallybright	3c
Salsa	3c

**Figure 10:** Different variety of strawberries and their base temperature from studies [3, 4, 5].

### 3.8 Summary

The various structural elements of the strawberry plant present challenges in their accurate modeling. Different varieties of strawberry plants do not grow the same and have differences such as the size and even colour of the fruit. For example, the Pineberry variety produces white fruit with red seeds rather than the more common red fruit and yellow seeds. A specific variety of strawberry plant chosen for the simulation can help with creating a consistent model rather than trying to combine unique elements from other varieties.

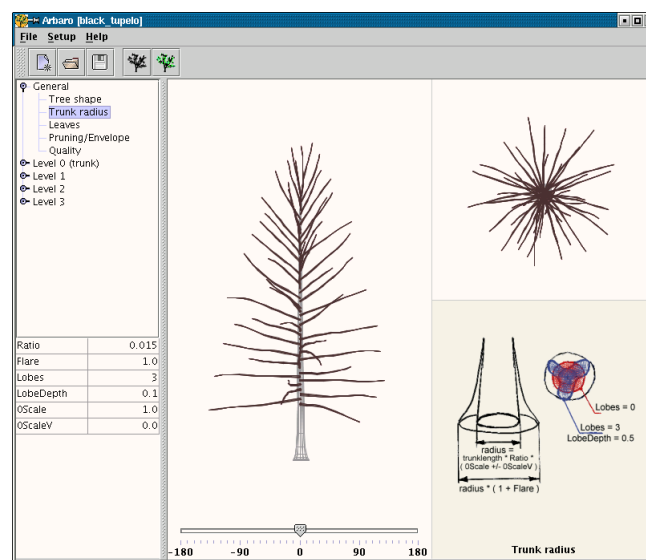
The plant's reaction to gravity and light is also challenging due to its complex structure. Different components such as fruit or leaves have different mass and shape. Different growing environments will also determine interaction of the plant with the gravity as plant components would hang if grown on raised beds or lie on the ground if they are not on raised beds. Different varieties will also affect the size and mass of the berries. It is also not trivial to determine how much sunlight is enough for plant growth. Other aspects of plant growth will also affect the simulated model. Soil fertility, quality, temperature and amount of water treated to the plant will affects its growth. These can also vary and change over time causing fluctuations in growth

or even death. On the other hand, temperature is relatively easy to measure and integrate with the plant model.



## 4 Software Frameworks for Plant Modeling

The most straightforward way of modelling synthetic plants is by manually designing them in 3D modelling software such as Blender or Autodesk Maya, which are both very popular amongst artists creating 3D assets for films, video games, and visual effects but also 3D printing. This method provides the artist with full control and can style a plant in any way they wish but can be a tedious and time-consuming process. Addons and extensions can also be used to improve the workflow for the designer such as the “Modular Tree” addon for Blender which allows for the easy creation of tree structures within the software. For that purpose, there are also standalone software programs that are designed specifically for the creation of digital plants and trees such as “Arbaro” which is specifically designed for creating realistic trees [118] which is usually combined with visualisation software such as POV-ray, an open source ray tracer [119]. They are useful for creating background scenery including plants foliage to allow designers to focus on other aspects of their work.



**Figure 11:** *The User interface for Arbaro software used to create trees.*

### 4.1 Plant Modelling Frameworks

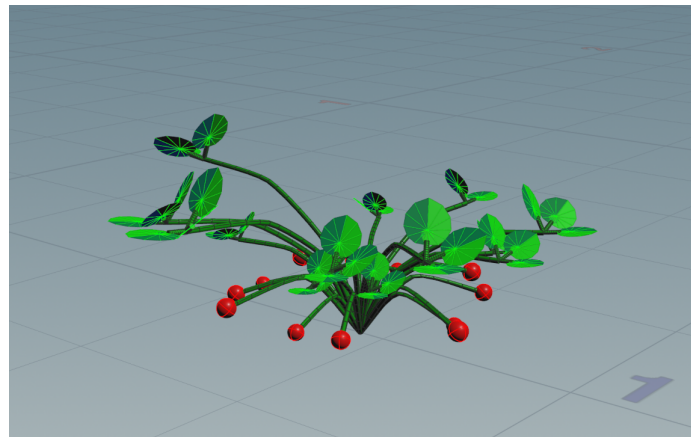
In this section, we present a summary of popular plant modelling frameworks which were considered for further technical developments including SideFX: Houdini, SpeedTree, Lpy, and L-studio. Smaller or incompatible frameworks were also investigated including TreeMagik G3, TreeGenerator, L-system4, Tree[d], Dryad, and MeshTree Studio but since they are no longer available or have OS compatibility problems they were not further considered. Also, other procedurally generated software which did not have elements of L-systems for user control were ignored [120].

Framework	Language	License
SideFX: Houdini	Python, C++	Non-Commercial
SpeedTree	C++	Paid-License
Lpy	Python, C++	Open-source
L-studio	C++	Open-Source

**Table 1:** *Popular plant modelling frameworks.*

#### 4.1.1 SideFX: Houdini

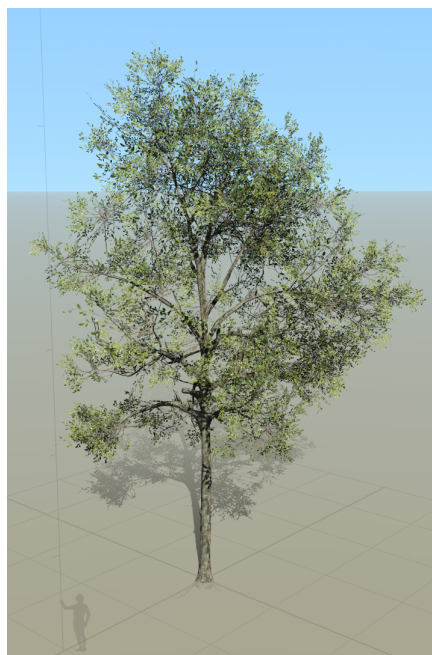
SideFX: Houdini is a procedural generation software tool commonly used in the film industry. It has a specific focus on procedural generation to distinguish it from other tools. It has been used in feature films such as Disney’s Zootropolis [121] or in Dreamworks Pictures film “The Time Machine” where L-systems were used to time-lapse a growth [122]. Its focus on procedural generation makes it useful for creating quick random content to be used in projects. Houdini has multiple versions including a learning edition that is free and used for non-commercial products. Indie, Artists, Education, and Studios licenses require a paid license or subscription to use which has various costs and features. Some studies have used SideFX for their research and for L-systems like converting 2D fractals into 3D [123]. Houdini is very well documented with tutorials and books as it is a large piece of software with industrial use.



**Figure 12:** *A basic strawberry plant model made in SideFX: Houdini using the L-system framework.*

#### 4.1.2 SpeedTree

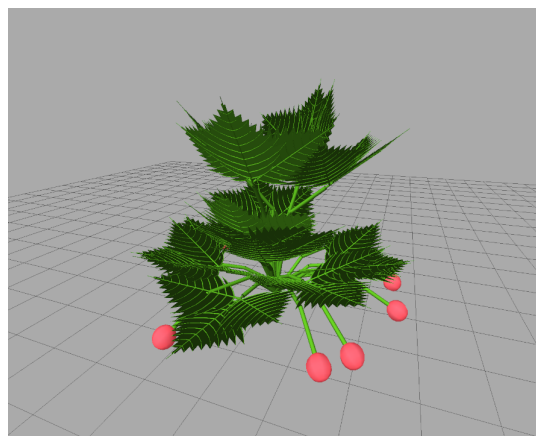
SpeedTree is vegetation programming and modeling software used to develop foliage for video games, films, and real-time simulation. It has been used in many productions like Disney’s Jungle Book where photo-realistic leaves, trees, and vines were needed for the scenes [124]. SpeedTree is known for its realism and customisation allowing to create high detail models like plum trees [125]. It can be used to create large scenes with high-fidelity models with reasonable performance [126]. The foliage is generated procedurally but still allows the user to customize the structure giving artists control to make more stylized content. The combination of quick procedural generation along with artist control allows developers to create high-quality models. SpeedTree is a subscription-based service but viewing SpeedTree models is free to use. It also has a library of pre-made models users can purchase to change and use.



**Figure 13:** *A procedural tree generated in SpeedTree.*

#### 4.1.3 Lpy

Lpy is an open-source Python framework built for the study of plant development. It allows for programming and the use of Python libraries to expand upon the framework and to provide more flexibility as compared to other frameworks which use synthetic overhead restricting its freedom [127]. Lpy uses the L-system rule set to generate plant-like structures onto a scene and can be exported into various file formats for further processing and visualisation in external programs. The framework stems from an OpenAlea library for plant modeling [128] which is used to create an IDE and GUI for users to create plant models without the need to do Python programming. The standard Lpy plant models are not as visually impressive and do not have the extensive features of SpeedTree and SideFX: Houdini such as path-traced lighting and high-resolution textures, however, its open-source nature and integration with Python allow developers to expand on the software and add their functionality. The framework is popular with the research community due to its open-source licensing, documentation and examples, and public code repository encouraging community contributions and support.



**Figure 14:** *A strawberry plant generated in Lpy.*

#### 4.1.4 L-studio

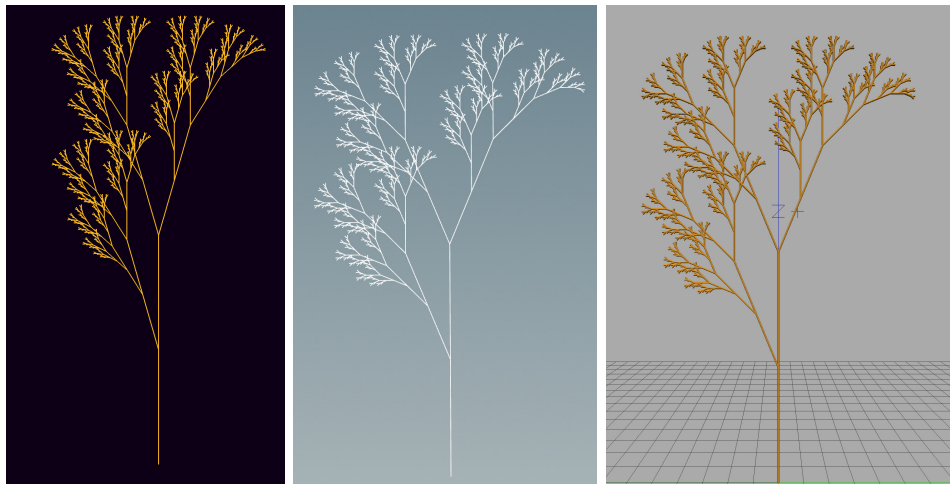
L-studio is a standalone program also designed for academic use [129]. It uses L-system to generate intermediate files (cpfg and lpfg) which are then used to simulate the models using L-system-specific constructs into the C++ programming language and OpenGL. L-studio provides a GUI and an editor for users to type in the rule sets rather than having to use C++ code making development easier.

## 4.2 Framework comparison

SideFX: Houdini, Lpy and L-studio all utilise L-systems with an exception of SpeedTree which does not provide precise detail on its simulation engine. SpeedTree has also paid licence and therefore was not selected for further comparisons. Although the selected frameworks use the same principles, the generated outputs from each system may differ, especially when it comes to visual appearance of the models.

### 4.2.1 Example L-system Model

As a model for comparisons. we used the same L-system premise (see Fig. 16) which corresponds to a branching tree structure. The generated outputs are quite similar besides the one generated by L-studio where the angle of rotation appears to be slightly different despite having the same value. This is a minor difference and does not have a negative effect on the results.

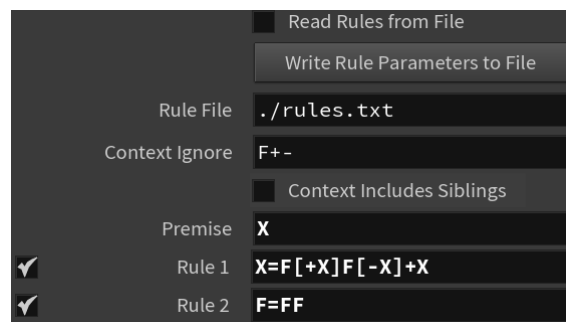


**Figure 15:** *From the left Lstudio, SideFX and Lpy output.*

```

1  #define STEPS 7
2  Lsystem: 1
3  derivation length: STEPS
4  Axiom: X
5  X --> F[+X]F[-X]+X
6  F --> FF
7  endlsystem
8
1  angle = 22.5
2
3  context().turtle.setAngleIncrement(angle)
4
5  Axiom: X
6
7  derivation length: 7
8  Production:
9  X --> F[+X]F[-X]+X
10 F --> FF
11 homomorphism:
12
13 F --> SetWidth(0.5) F
14
15 endlsystem
16

```



**Figure 16:** The same rule set in different frameworks starting with Lstudio top, Lpy middle and Houdini SideFX bottom.

The L-system framework uses a set of symbols each with meaning to create structures using turtle graphics-like syntax. For example, symbol “F” means to move forward which creates a line of a specific length. The most common symbols used in L-systems are presented in Table 2. An L-system starts with an initial string of symbols called an axiom. Changing the derivation length is the number of iterations the L-system will run. More steps will cause an exponential increase in strings and require more time to create and render. Rotation and length can also be edited to increase the height or to have straighter branches. Individual frameworks might support a subset of symbols, have additional unique ones, or require certain conditions, for example, symbol “/” requires a 3D environment to be supported. Various parameters can be edited to change the output and add randomness to the model. SideFX: Houdini has unique rules exclusive to its software and makes use of its node system. Lpy also has some unique characters for generating 3D primitives but a fine balance is needed as the 3D plant model can clip into each other and give us an undesired result.

Lpy uses Python programming for the developer to write L-system strings to generate 3D plant structures. This requires previous knowledge coded in Python as compared to other applications such as SideFX: Houdini which is more designer friendly. Lpy follows the L-system rules with the benefit of having additional Python functionality. A set of L-system rules can be made into a function to pass variables and normal Python code but normal L-system rules are

Character	Output
"F"	Move forward a certain amount
"f"	Move forward but do not draw
"_"	Turn right by a certain degree
"+"	Turn left by a certain degree
"["	Save current location
"]"	Return to last saved location
"__"	Reverse Direction
"#"	Increment the Line width
"!"	Decrement the Line width
">"	Multiply the line length by the lengths scale factor
"<"	Divide the line length by the length scale factory
"("	Decrement the turning angle
")"	Increment the turning angle
"\"	Roll clockwise
"/"	Roll counter-clockwise

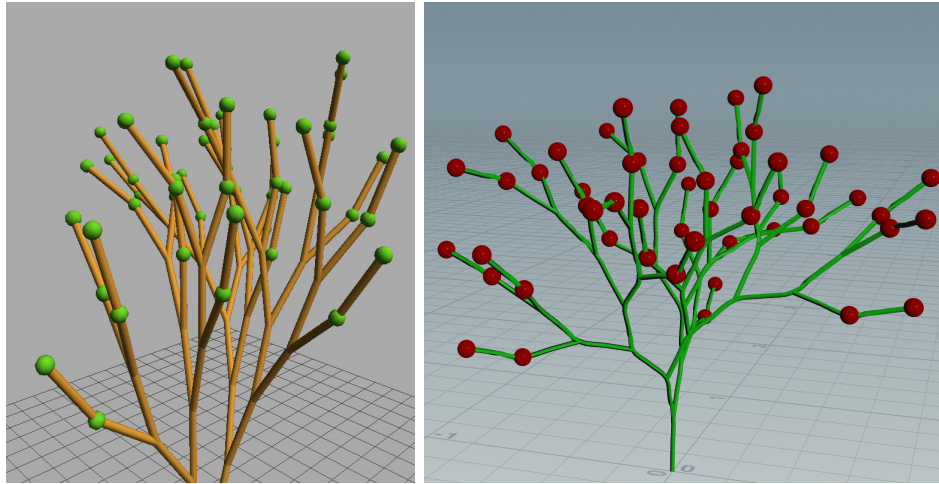
**Table 2:** *L-system characters and their meaning*

still usable. It uses PlantGL, an open-source geometric library for 3D plant modeling. PlantGL allows for exporting the models into various formats that can be used in other frameworks such as Blender. L-studio is simple and uses its custom config files (cpfg/lpfg) to interpret the code and then render output. Lpy uses a similar workflow so the two frameworks are similar in style but with Lpy offering more features [127].

In SideFX: Houdini, an L-system feature is available for users to create L-system strings to generate structures. It provides an L-system feature to create 3D or 2D structures. The L-system support in this framework is limited when compared with Lpy but it offers many other tools and features not associated with L-systems. SideFX: Houdini offers new and unique rules to the L-system such as 'j' which is used for a leaf input in the node system. It is widely supported by other frameworks such as the Unreal Engine which has a plug-in for easy integration between the two frameworks. Since the software is not open-source additions and modifications to the software cannot be made.

#### 4.2.2 Fruit and leaf generation

The framework needs to be able to generate additional structures at the end of branches for the strawberry fruit/leaf generation. Both Houdini and L-system have additional rules to generate non-L-system objects. Houdini's node system allows for multiple different models and features to be added onto the L-system such as simple 3D structures. However this is limited as the L-system only supports three inputs. Lpy allows primitives such as spheres, cylinders and cubes to be generated on the L-system similar to Houdini. Since Lpy is written in Python the user can create their own functions and use other libraries to generate and create their own rules unlike Houdini.



**Figure 17:** A branching structure model with additional visualisation objects (spheres) generated in Lpy (left) and SideFX: Houdini (right).

### 4.3 Summary

Overall there are many frameworks that utilise procedural generation and L-system in particular. The larger applications such as SpeedTree and Houdini require commercial licenses and are aimed at video games and cinema. Since they are closed-source, it is unclear what methods are used for procedural generation. Open-source frameworks are tailored towards academics such as Lpy and allow for expandability. Lpy was the chosen framework because of its open-source nature, documentation, and expandability. It is still actively maintained on Github but only by a single developer so the updates are not frequent or substantial which puts it at a disadvantage when compared to the larger frameworks. It does lack the photorealism seen in SpeedTree and Houdini which would be a nice touch to add to the models, making the details much sharper. The cost barrier and the closed-source nature of the frameworks would make them difficult to work with. It would be difficult or impossible to extend the frameworks to include features that are not available. L-studio is still available to use but it has not been updated in a long time, Lpy uses a lot of the features in L-studio and has expanded on them making it feel like an advancement of L-studio.



## 5 Strawberry Plant Model

### 5.1 Plant Components

Lpy is used to create the procedural model of the plant using component based method. Each component represents a part of the plant and will be explained in detail to show how it is affected by other parts of the models such as tropisms.

#### 5.1.1 Primary Crown

```

1  #Primary Crown
2  PC(x):
3      #If more than 10 leaves are generated generate fruit
4      if (x >= 10):
5          for a in range(rd.randint(1, 3)):
6              #component to generate the fruit
7              nproduce [Rol()INF(0)]
8      for leaf in range(x):
9          #generate each leaf
10         nproduce [Rol()L(leaf)]
11

```

**Figure 18:** Code for the primary crown written in L-py.

The primary crown (see Fig. 18) is the first component generated in each plant. From the primary crown, all the leaves and fruit grow. Fruit can only grow if there is at least ten leaves generated and will generate 1-3 stems with the growing fruit.

```

12 #Spiral generation
13 Rol():
14     nproduce F(0.05)/(Spiral())
15
16 #Get Last roll and add new roll in
17 def Spiral():
18     global spiral
19     spiral = spiral + rd.randint(50, 70)
20     return spiral
21

```

**Figure 19:** Functions to grow crown and to create a spiral around the crown for leaves and fruit to grow.

The “Rol” function (see Fig. 19) grows the primary crown slightly and increments the spiral for the fruit and leaves to grow evenly around the crown. A small amount of randomness is added by choosing a value between 50 and 70° so the plant looks less uniform. This choice is motivated by the fact established by Guttridge in his study [2] who observed that the sixth leaf grows on top of the first grown leaf.



### 5.1.2 Leaf and Petiole

```

22  #Leaf polygon generation
23  L(x):
24      #generate petiole length
25      petiole = rng.uniform(ny.min(Petiole), ny.max(Petiole), 1)
26
27      #set the colour and scale the texture
28      nproduce SetColor(2) TextureScale(1)LF(round(petiole[0] / 3), x)
29
30  #Leaf generation
31  LF(x, leaf):
32      #The current leaf
33      T = leafs - leaf
34
35      #Generates leaves with rayleigh distribution based on the mode of the
36      real data.
37      LeafArea = rng.rayleigh(137, 1)
38      #loop to create new values in case the value generated was above the min
39      or max of the original data
40      while LeafArea < 30.41 or LeafArea > 323.64:
41          LeafArea = rng.rayleigh(137, 1)
42
43      LeafArea = ((LeafArea[0] / 3) / 3) / 3
44      #LeafArea = round(T / 5) + 1
45
46      #Get weight based on the leaf area
47      DryWeight = round(2000 / 1 + float(decimal.Decimal(LeafArea)/5))
48
49      #the older the leaf the bigger the angle
50      angle = T * rd.randint(1, 10)
51      #Cannot exceed 70 degrees
52      if (angle > 70):
53          nproduce +(70)
54      else:
55          nproduce +(angle)
56
57      for a in range(x):
58          #If hair is enabled generate
59          if (Hair == True):
60              nproduce Hr(100)
61              nproduce @Tp(LightLoc(Px, Lx), LightLoc(Py, Ly),-1) Elasticity(abs(
62                  tropism)/DryWeight) nF(1, 0.2)
63              nproduce /(90)P(LeafArea)

```

**Figure 20:** *Petiole and leaf generation component.*

Leaf generation takes up most of the 3D space of the model and requires parameters which can be measured from real observations (see Sec. 6). The petiole is the stem that grows from the primary crown with the growing leaves at the end. The length of the petiole is generated using the uniform distribution, which was established through the parameter fitting procedure presented in Sec. 6. The value is parsed onto the main leaf generation where the leaf area is generated by using the Rayleigh distribution which was also established through the parameter fitting procedure based on observation of real plants (see Sec. 6). It also checks to see if the value is within bounds based on max/min values established from real measurements. The age of the leaf effects the angle from which it grows from, with a small amount of randomisation to prevent it from being too uniform. The older the leaf, the greater the angle is, as it would be pushed down from newer leaves growing. It cannot exceed 70° however to prevent the plant

from growing into the ground. The gravity and weight are applied to the inbuilt tropism and elasticity function. Phototropism also effects the direction of the leaf growth. Strength of the tropism can be modified to increase or decrease the strength of gravity and is then divided by the leaf weight a heavier leaf will result in a stronger effect of tropism.

### 5.1.3 Fruit

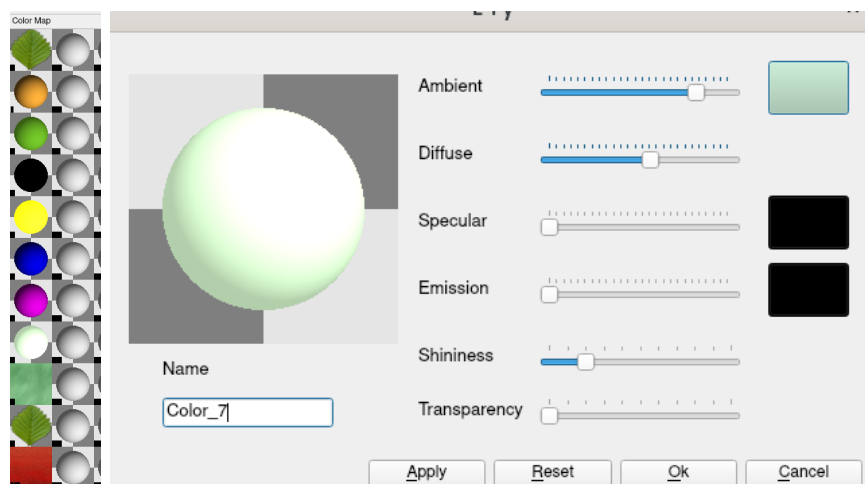
```
62  #Fruit
63  INF(x):
64      #Create number of inflorescence based on the mode of Riseholme data using
65      Rayleigh
66      s = rng.rayleigh(5, 1)
67
68      nproduce SetColor(2)/(5)+(15)@Tp(0,0,-1) Elasticity(abs(tropism)/1000.)nF
69      (1.5, 0.1)[nF(2, 0.1)Berries(int(s))]
70
71  BWeight(x):
72      Size = round(50 / x)
73      nproduce @Tp(0,0,-1) Elasticity(abs(tropism)/Size)
```

**Figure 21:** *Strawberry generation and weight component*

Fruit generation is similar to the leaf with it growing a stem before it splits to produce the strawberry at the end. An initial stem grows before splitting into other stem-like structures before finally growing the fruit at the end of it. The size of the strawberry uses the Rayleigh distribution, established from real observations (see Sec. 6). The weight uses the same tropism for the gravity but the elasticity is affected by the weight of the fruit leading to heavier and larger fruit causing more of the stem to bend.

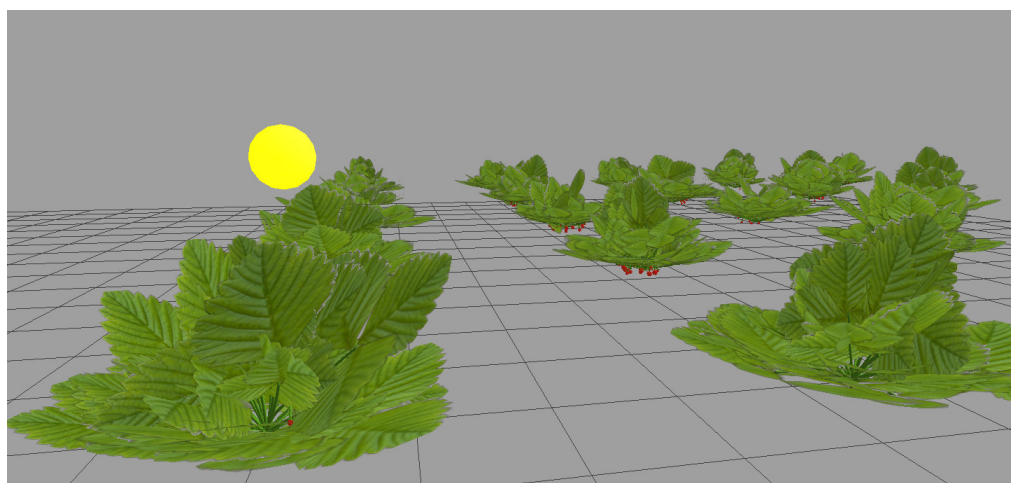
### 5.1.4 Visual Appearance

To make the generated models more visually appealing and realistic, the textures and different colours were applied (see Fig. 22). Lpy has some parameters to help improve visuals but not as many as other 3D rendering software like Blender which features a wide set of parameters and functions to change the look of an object. An example is a normal map which makes an object look more detailed by making the bumps look 3D in the texture [130]. It is ideal for objects that have imperfections such as the bumps on a strawberry. Lpy only has support for ambience, diffuse, specular, emission, shininess and transparency which for the developed model were left at default values.



**Figure 22:** *Lpy colour map and the options to edit the colours.*

The model was further enhanced by the use of textures for leaves and fruits rather than colour maps. This was a relatively straightforward procedure in Lpy (see Fig. 23).



**Figure 23:** *Field of strawberry plants all individually randomly generated*

Using Lpy, we have created a procedurally generated 3D model of a strawberry plant using L-systems. The user can create a field of strawberry plants by specifying how many columns and rows should be generated (see Sec. 23). Lpy only uses python and the execution time on a AMD Ryzen 9 5950x CPU is 0.12-0.2 milliseconds on average when generating the L-system string, just generating one plant is only 0.01 millisecond. The render time can take 1-2 seconds for 100 plants and for just one plants it takes 0.1-0.2 seconds.

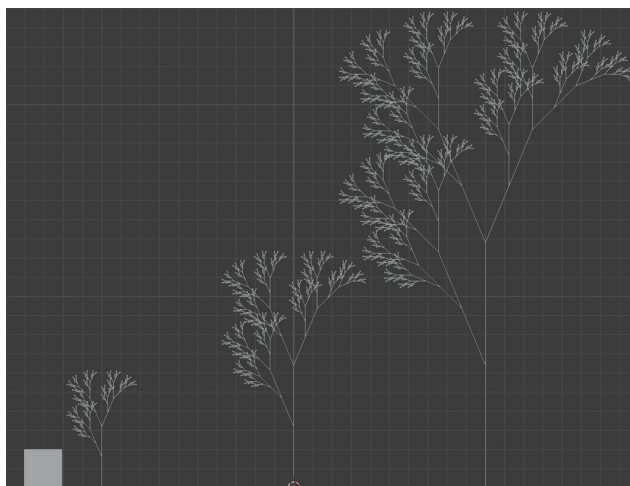
Parameter	Value
tropism	9.81
Hair	<input type="checkbox"/>
Point Light	<input type="checkbox"/>
Phototropism	<input type="checkbox"/>
Temperature	
High_temp	0.0
Low_temp	0.0
Days	
Days	140
SDLD	<input type="checkbox"/>
months	6
start	1

**Figure 24:** Editable parameters to affect the growth of the plant.

Lpy allows to create a set of configurable parameters that can be used in the code to change the output. They act as global variables which can be read but not changed during the running of the application. Within the proposed model, the select parameters included the strength of gravity, how many days or months of growth have passed and the temperature value bounds which influence the growth (see Fig. 23).

## 5.2 Growth Functions and Modeling

This part investigates the implementation of functional aspects of the plant growth. The models are made using Lpy and exported into Blender to showcase the size and to allow for comparison between models and their different parameters. A two metre cube, placed in the scene, is used for size comparison of the structures. Firstly, the amount of iterations that will go through the axiom can be changed (see Fig. 25). It results in an exponential increase of characters generated, making the model scale up in size.



**Figure 25:** Original L-system starting with 6 iterations on the left then 7 and 8

The angles at which the branches can grow can also be changed for a tighter or wider spread (see Fig. 26). A simple change of this value can greatly changed the output of the result. Angles are measured in degree turns the same as real life which is easier to implement real life plane angels to create plant models.



**Figure 26:** *The original L-system model (centre), at an angle of  $10^\circ$  (left) and at  $40^\circ$  angle (right).*

### 5.2.1 Stochastic L-system

```

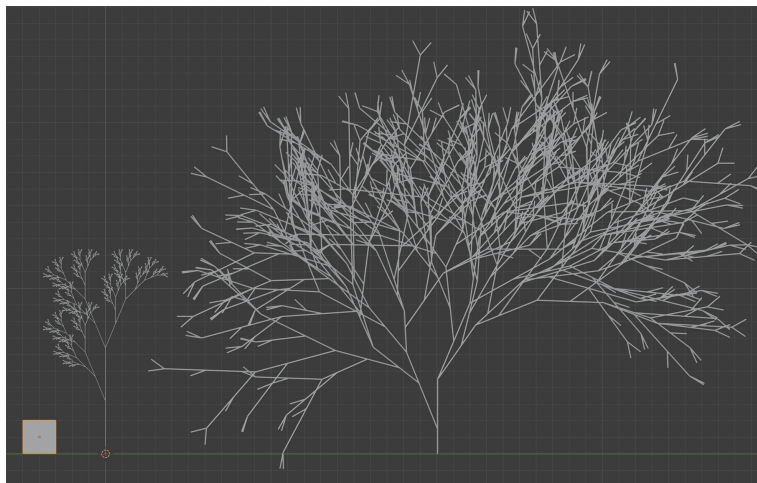
73  Axiom: X(rd.randint(1,3))
74
75  derivation length: 7
76  production:
77  X(a):
78      if a == 1:
79          nproduce F[+X(rd.randint(1,3))]F[-X(rd.randint(1,3))]X(rd.randint
80          (1,3))
81      elif a == 2:
82          nproduce F[-X(rd.randint(1,3))]F[-X(rd.randint(1,3))]X(rd.randint
83          (1,3))
84      else:
85          nproduce F[-X(rd.randint(1,3))]F[+X(rd.randint(1,3))]X(rd.randint
86          (1,3))
87  F --> FF
    
```



**Figure 27:** *Using stochastic L-system in Lpy with the original output on the left*

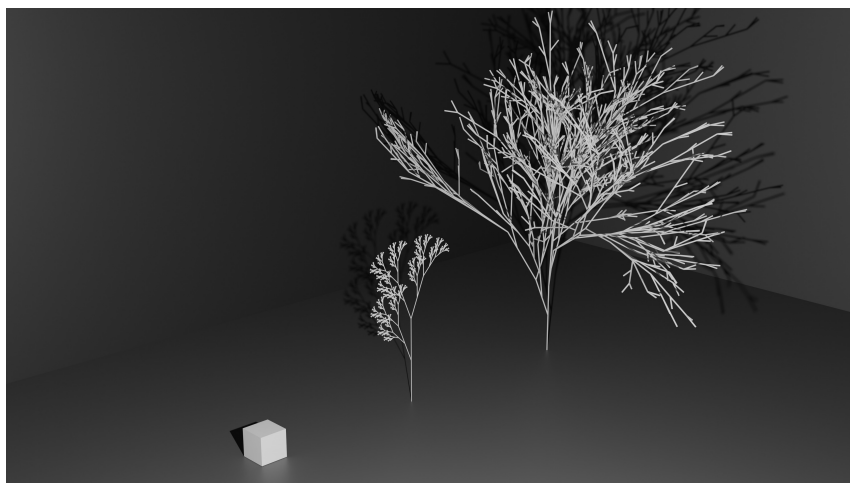
Stochastic L-system adds randomness by integrating variation of the rule with a chance for a symbol to output something different. Stochastic L-systems are documented in the original book “The Algorithmic Beauty of Plants” [16]. Fig. 27 shows three variations of the rule each

with a different arrangement of turns using “+” and “-” symbols. This creates a slight change each generation while retaining the model’s consistency. Adding more randomness can lead to deteriorating results as can be seen in Fig. 27 where the model to the right shows all the branches generating on the right side of the plant. This could be considered an undesired affect since one side of the plant has no growth on it.



**Figure 28:** *Adding random length and angle to the rules.*

More variations can be added or more rule variations such as adding an extra branch to the structure. Figure 28 showcases the combination of random angle and length with the previous example. This results in even more variation by creating a large tree with long branches. This can easily be done in Lpy because the framework can utilise random number generators provided by the Python libraries. Other software such as L-studio are unable to include or expand on the framework in the same way. Adding in new character rules can also greatly change the output. For example, adding “/” will change the direction of the growth into a 3D structure (see Fig. 29). Applying a random value of 0-360° will roll on a Z axis creating a 3D tree-like structure. The simple 2D fractal branch has been changed into a random 3D structure by adding and changing some of the values. This will create quite a chaotic result with branches colliding into each other and growing randomly. Measures like collision detection and restricting the randomisation can all help to mitigate such results. Methods like checking if the growth will be within a volume such as a box and stopping the growth of the plant [131]. Stochastic l-systems are used in my model to randomise the plant growth based on the original Lindemayers method [16] but with constraints that are based on the real plant such as new leaf formations grow in a uniform spiral rather than randomly around the crown [2].

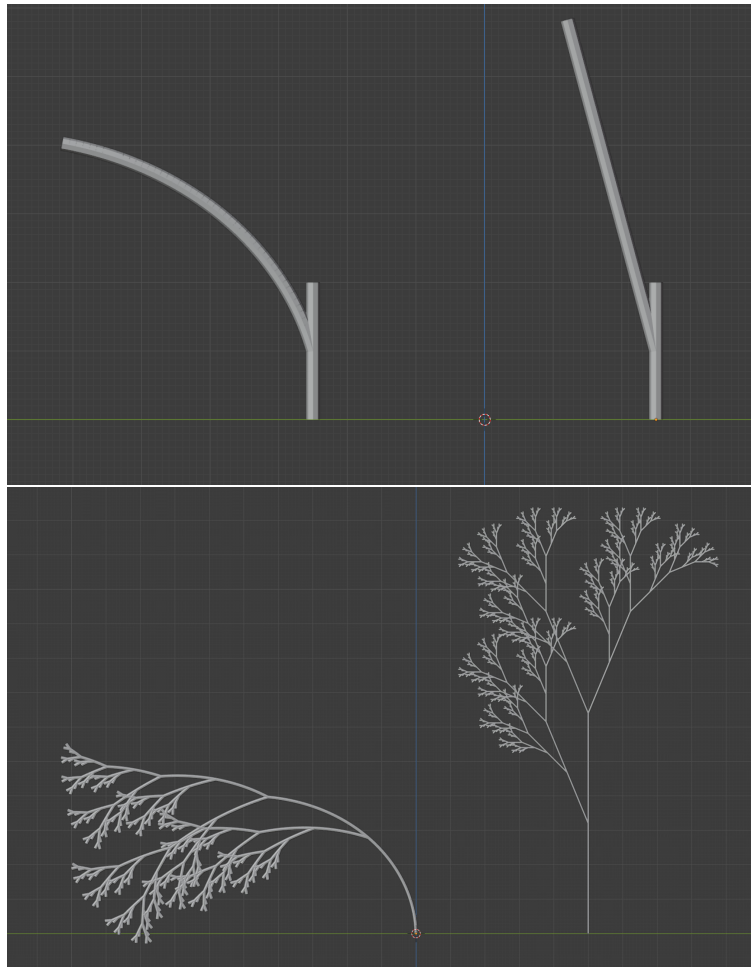


**Figure 29:** *Using the “/” with a random value of 0-360 to create a 3D structure.*

### 5.2.2 Gravitropism

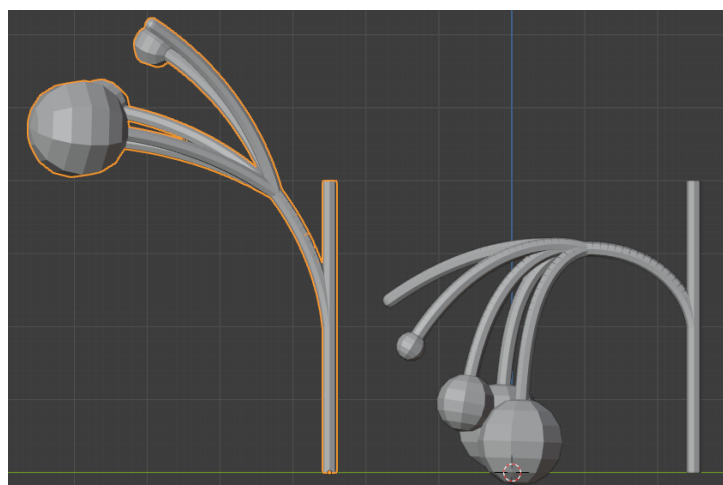
L-systems on their own do not acknowledge tropisms so the generated models are static and only represent the components (i.e. genetics) of a plant. Additional features are needed to be implemented to improve L-systems in this aspects. Both SideFX: Houdini and Lpy have a tropism feature. A method to implement gravity into L-systems is to take a bio-mechanical approach. A branch can be broken into internodes and one can assume the branch is growing straight without gravity. When applying gravity, we also assume that each node is an elastic joint. The joints have torque caused by gravity [132] which can be seen in Fig. 30 with visible bending branch/structure. In the top image, the gravity is applied just to the branch growing off the stem which leaves the main stem unaffected by gravity. In the image below, the gravity is applied to the whole of the plant causing it to bend to one side.





**Figure 30:** *Tropism implemented with elasticity in L-py on the left and no gravity on the right.*

Additional weight will effect the bend of a branch and even the growth of a plant. The support force of the branch is the amount of dry matter of the fruit, leaves and flowers [97]. Using the current bio-mechanical model for bending of the branches, editing the elasticity depending on the dry matter will cause the branch to bend more under the weight.

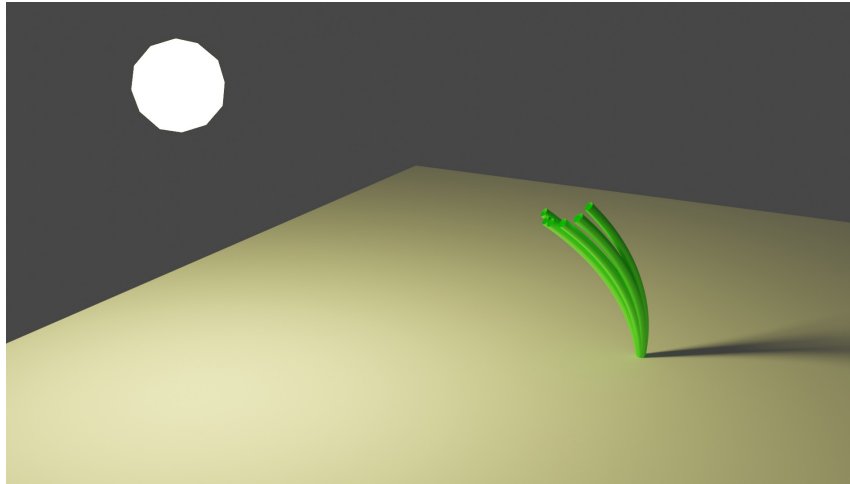


**Figure 31:** *Normal gravity with assumed weight on the left and Dry weight added on the right.*



### 5.2.3 Phototropism

In order to simulate phototropism, a light source is needed for the plant to bend toward. Software like Blender uses accurate lighting with path-tracing and can be used to calculate the lights position to bend towards [131]. Radiosity is an alternative algorithm used for lighting and operates by accounting for the light paths coming from the light. Its benefit over other algorithms is that it can be used to calculate heat/energy transfer from a light source [82].

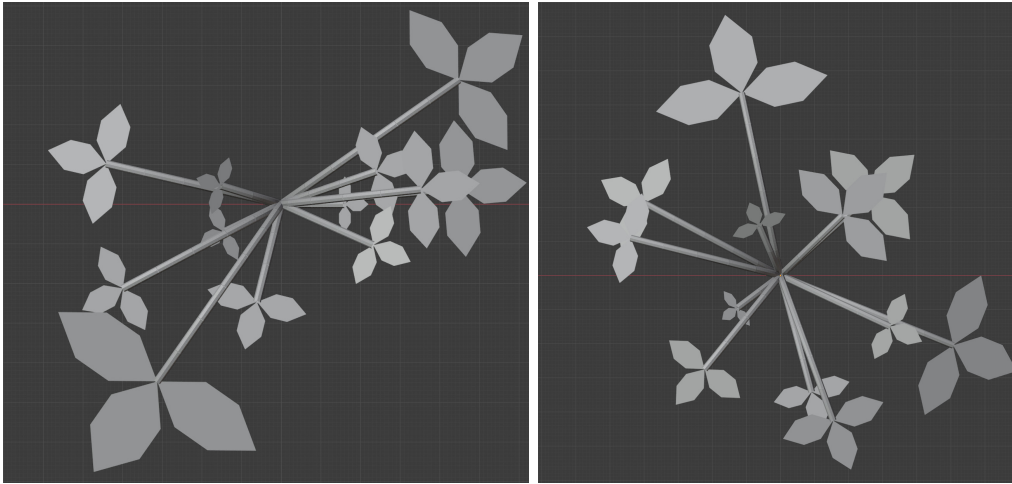


**Figure 32:** *Basic phototropism of the plants bending towards the light.*

A simple method of knowing where the light is, can be achieved by placing a point light in the scene. Depending on the light location, the plant will bend based on the x,y position of the light. If the light x position is greater than the plant's, then the tropism x will equal 1, if the light position of x is negative the tropism will be negative causing it to bend towards the light location.

### 5.2.4 Collision Detection and Avoidance

A risk of using procedural content generation is that the randomness of the generation can cause the components of the plant to collide and clip into each other. L-systems generate the rules and then create a 3D structure from them meaning the collision calculation needs to be done before the model is made. In the Blender's Lpy addon, the branches are simply cut off if an object is in the way creating unrealistic looking growth [131]. Rigid body and soft body physics is used in simulations for collision detection but this is computationally demanding and requires time to process [133]. This would also require more development time as the Lpy framework does not support real time simulations.



**Figure 33:** *Random Values for leaf generation causing clipping on the left image. Uniform generation with slight randomness reduces clipping and adds evenness.*

### 5.2.5 Growth

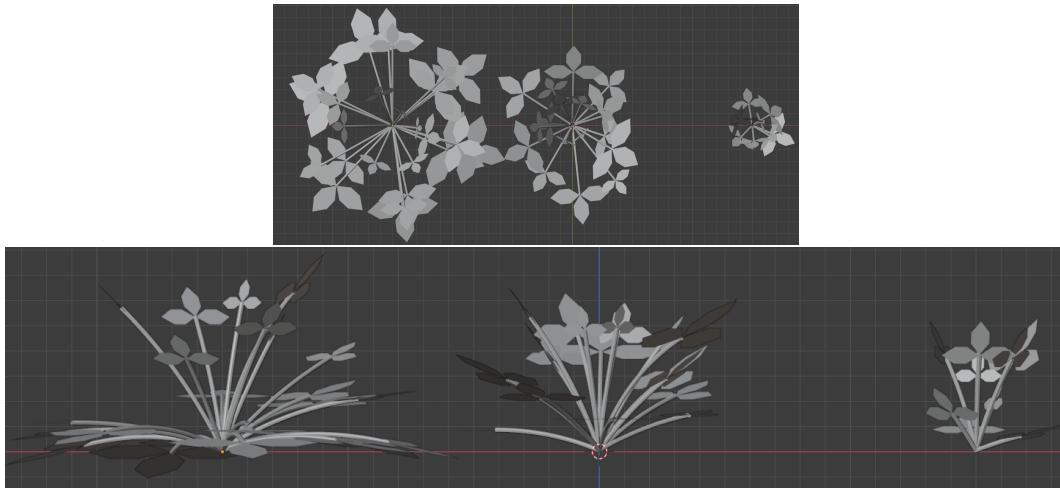
To simulate the plant's growth over a period of time, we can use the Growing Degree Days (GDD) value which is used by plant scientists to normalise the growth time with respect to temperature. We can use GDD to estimate how many leaves will have grown during that period of time rather than randomising the number. It is estimated that every  $63 \pm 3$  GDD a new leaf will emerge from the plant [4]. We can use the recorded temperature of the UK over the months from the Met office which provides the highest and lowest temperatures every year [134]. Also manually setting the temperature as well allows for more experimenting as seen in Fig. 35.

```

86  #Calculate the Growth Degree days
87  def CalGDD(high, low):
88      base = 7
89      if (low < 7):
90          if (high > 30):
91              return 30 + 7 / 2 - base
92          else:
93              return high + 7 / 2 - base
94      elif (low > 30):
95          return 30 + low / 2 - base
96      else:
97          return high + low / 2 - base
98

```

**Figure 34:** *Method for calculating Growth Degree Days (GDD).*



**Figure 35:** *Comparison of days passed 30, 60 and 90 days at a constant temperature of 15° C and 7° C. Starting with 30 days on the right.*

## 6 Evaluation of the Plant Model

The strawberry plant model had been developed and procedurally generated using the L-systems approach described in Sec. 5 using the L-py framework [127]. The key structure of the plant model and key parameters were derived from two sources of image and measurement data collected from 1) a research farm at Riseholme campus of the University of Lincoln, where strawberries are grown in polytunnels using standard industrial practices and from 2) an observational study of strawberry plants grown in pots under a grow light in indoor environment throughout the duration of the project conducted by the author of this thesis. The specific measurements collected include leaf area, petiole length and inflorescence including flowers and strawberry fruit per single stem. The two datasets were also used to evaluate the realism of the generated strawberry plant models by direct comparison to the measurements obtained from the simulations. Certain aspects of the functionality such as gravity and phototropism were evaluated in a qualitative manner due to complexity of the processes involved.

### 6.1 Data gathering



**Figure 36:** *The two growing environments used for data collection: a row at the Riseholme strawberry farm (left) and a homegrown strawberry setup (right)*

The research farm at Riseholme features strawberries grown in polytunnels elevated from the ground in growth bags with an automated watering system (see Fig. 36). A total of 1610 plants were planted and maintained in 10 rows during the 2021 season. Five random plants were chosen to collect the measurements of leaf area, petiole length, and inflorescence. The data was collected on the 4<sup>th</sup> of August 2021 using basic equipment including a ruler and paper notepad to record the data. Since, the farm is constantly maintained which includes weekly leaf trimmings, this can affect the actual measurements compared to other grown strawberry plants that are not being pruned. The two varieties from Riseholme were Amesti and Zhara.

The homegrown strawberries for the observational study were grown in 9 cm deep and 7 cm wide pots. Initially, 5 plants were grown but 4 of the plants died after a few months due to difficult growing conditions. 5 new plants of the Royal Sovereign variety were bought online as dormant runners to replace the dead crop. They were planted on the 12<sup>th</sup> of March 2021 with the data being gathered on the 8<sup>th</sup> of July 2021 allowing for a few months of growth. One of the new plants had withered away before July leaving 4 plants for data gathering. The grow light setup used was switched on for a period of 12 h every day within a dark room with no exposure to natural sunlight (see Fig. 36).

### 6.1.1 Leaf Area

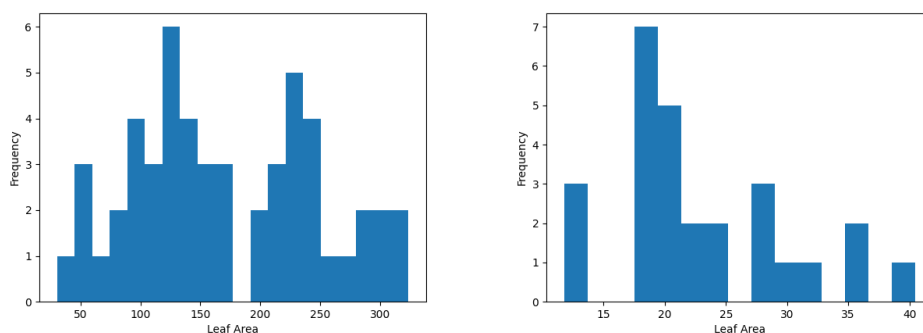


**Figure 37:** *Measuring the upper lobe length of a strawberry plant leaf.*

The leaf area can be calculated in a non-destructive way (see Fig. 37) using two measurements including upper leaf lobe length and left lobe width with appropriate weighting values [135]:

$$\text{leaf area} = 1.89 + 2.145 * \text{upper lobe length} * \text{left lobe width}. \quad (2)$$

A more accurate method of using a planimeter can be used to measure the leaf area but this requires leaves to be cut off. There are apps also available that can estimate a leaf area by taking a photo of a leaf on a clear background such as LeafArea by Skyberry or PocketLAI [136]. The user can manually draw over the leaf image and annotate them or use an automated procedure which detects the green colour of a leaf. This method is also destructive and hence not used in this work. A total of 52 leaves were measured at the Riseholme farm of different varying sizes and different stages of growth. 27 leaves of the home grown plants were measured, some older dried and dead leaves on the plant were not measured as they had changed in size or were damaged and were no longer growing.



**Figure 38:** *Distribution of the leaf area measurements taken at the Riseholme farm (left) and at home (right) measured in cm.*

When comparing the distributions of the two datasets, the homegrown plants demonstrate a smaller leaf area which is most likely due to the fact that the plants at home were grown in small pots limiting their root growth, therefore, reducing the overall size of the plants. Both datasets have a spike in the distributions, which is especially evident in the home data with the majority of the leaf area recorded between 16-21 cm<sup>2</sup>. The distribution of the data from

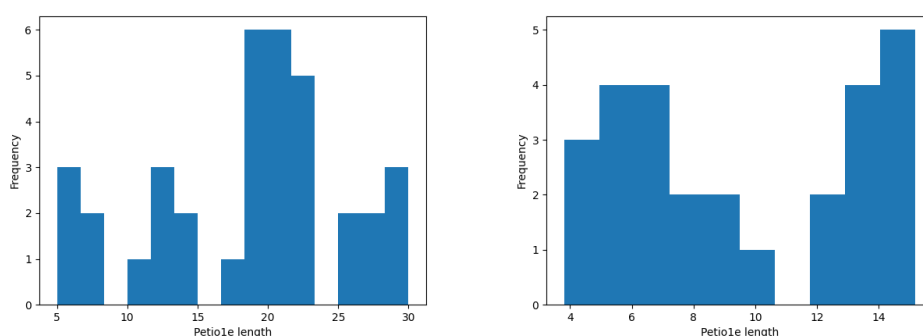
the Riseholme farm resembles a bell curve with increased frequency around the center of the histogram. There is a gap that appears in middle of this histogram caused by a small sample size and the selected histogram parameters which is likely to disappear if larger samples size was used.

### 6.1.2 Petiole Length

The same 5 plants from the Riseholme farm which were used to measure the leaf area, were also used to measure the petiole length. It can be relatively difficult to calculate the petiole length without damaging the plant due to some of the petioles not being straight and to overall high density of leaves, fruit and other plants making it hard to reach individual petioles. All these factors can affect the quality of measurements. The distribution of the collected petiole length (see Fig. 40) also resembles a normal curve with more frequent values appearing around 17-20 cm but there is also an increase of length frequency in 5-8 cm region most likely caused by new younger leaves growing from the plant. The home grown plants resembles a bimodal distribution with two peaks on each end of the graph. The length were shorter than the Riseholme plants with the longest petiole length being 16cm.



**Figure 39:** *Measuring petiole with a ruler.*



**Figure 40:** *Distribution of the petiole length measurements taken at the Riseholme farm (left) and at home (right) farm measured in cm.*

### 6.1.3 Inflorescence

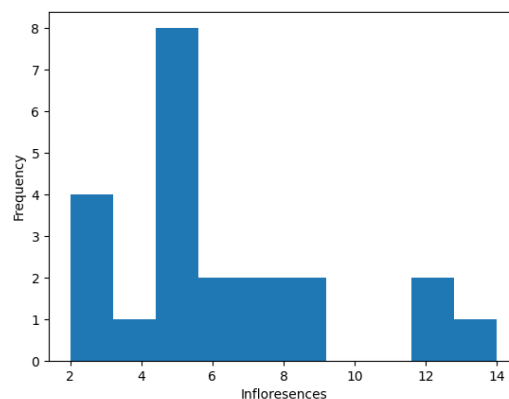
For inflorescence measurements, the flowers and fruit per stem were counted for plants from the Riseholme farm, to see how many fruit/flowers could grow on a single stem for a single plant.



Looking at the distribution of the flower/fruit counts there is a large spike in inflorescence values between 4-6 and a few large clusters of 12-14 possibly from larger plants capable of producing larger clusters of flowers and maintaining the fruit for longer. Since, the homegrown plants were cultivated in artificial conditions, there were no present inflorescence at the time of data collection and hence these measurements were omitted in this part.



**Figure 41:** *Inflorescence from a real plant.*



**Figure 42:** *Distribution of inflorescence measurements taken from the Riseholme farm measured in occurrences per stem.*

## 6.2 Model Parameter Tuning

The collected measurements from both scenarios are then used to estimate the best fitting distributions which consequently will be used for generation by the plant model. To that end, we select a number of popular distributions including Normal, Rayleigh, Weibull, Chi, Uniform and Exponential Power. These distributions are well known and are available in the numpy and SciPy python package making it easy to implement into the model [137]. The data was fitted using a Maximum Likelihood Estimation method implemented by the SciPy Python library [138] with a convenience wrapper called Fitter [139] allowing for automation of the process and visualisation of the results.

To evaluate the best fitting distribution, we use the sum of square error (SSE) describing the differences between the predicted and observed data points. The lower the error value the better the distribution fits the data points. Other information used to find the best fit is the Akaike information criterion (AIC) [140] and the Bayesian Information Criterion (BIC) [141]

Name	Formula
Normal	$p(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-(x-\mu)^2/2\sigma^2}$
Rayleigh	$p(x) = \frac{2xe^{-x^2/\alpha}}{\alpha} \quad x > 0$
Weibull	$p(x) = \frac{\alpha}{\lambda} \left(\frac{x}{\lambda}\right)^{\alpha-1} e^{-(x/\lambda)^\alpha}$
Chi	$p(x) = \frac{(1/2^{k/2})}{\Gamma(k/2)} x^{k/2-1} e^{-x/2}$
Uniform	$p(x) = \frac{1}{b-a}$
Exponential Power	$p(x) = \frac{1}{\beta} \exp(-\frac{x}{\beta})$

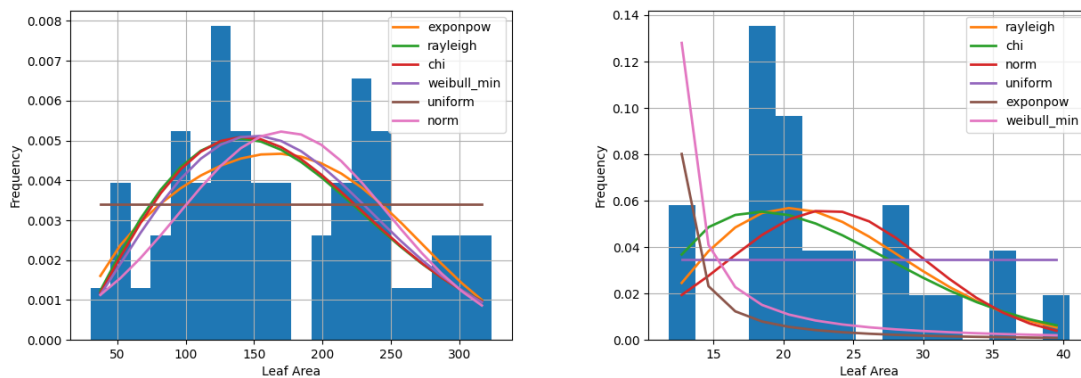
**Table 3:** *Selected distributions for fitting data measurements.*

which both indicate better fit of the model to the data with their lower values. It is worth noting that these selected metrics do not always agree in ranking different distributions and for the final choice we rely on the SSE metric. We also use an autofitter which finds the best fit out of all distributions for the data set using SSE, AIC, and BIC. We mainly use SSE to evaluate the best fit for each dataset as it looks at the spread of data with the predicted value therefore if there is a large spread of data when implementing the distribution to my model it will not be very accurate as the prediction will be poor.

### 6.2.1 Leaf Area

When comparing the plants grown at the Riseholme farm and at home, we can see significant differences in the selection of the optimal distributions so two distributions for each dataset is required to have a best fit rather than trying to choose a distribution to represent both. The Weibull and Exponential Power distributions indicate exponential left-biased curves in the homegrown plants unlike the fitted distributions for the Riseholme farm plants where all have a consistent bell curve seen in Figure 43. The homegrown plants have significantly smaller leaves with a few larger examples causing the left-side skew. This is likely to be caused by the fact that these plants were only grown in small pots rather than in ground limiting their access to nutrition and it is likely that bigger plants may have shown more consistency. The homegrown plants have also far fewer samples which might have caused further issues with distribution fitting process.





**Figure 43:** Leaf area distribution models fitted to the data from the Riseholme farm (left) and home indoor plants (right).

The Rayleigh distribution feature highly in rankings for both datasets hence this is a good candidate for the optimal model to be used in the simulations followed by the Chi distribution. The Rayleigh distribution has ability to deal with skewed distributions characteristic for the indoor data and was the second best fit for the outdoor dataset with the differences very small especially with the SSE which had a 0.000001 difference to the highest scoring distribution. The second best choice is the Exponential Power distribution featuring the best SSE, AIC and BIC values for the outdoor data (see Table 4). The scarcity of data from the home indoor plants and their generally low leaf area values resulted in different distribution fits with the Rayleigh distribution leading the ranks (see Table 5).

The Chi distribution for both datasets provided a good fit with the same SSE values as Rayleigh for the outdoor Riseholme plants but its AIC and BIC had higher values than the Rayleigh and Exponential Power distributions. For the homegrown plants the Chi distribution was the second best - it can also deal well with the skewed distributions of the homegrown plants. The Weibull distribution for the Riseholme outdoor plants was a relatively good fit only having 0.000003 difference to the best fit (Exponential Power). Its AIC and BIC values were similar to Chi but characterised by higher SSE values. For the homegrown plants it had the worse fit with a high SSE of 0.059848. It also creates an odd curve in the graph which is similar to the Exponential Power fit.

The Uniform distribution proved to be relatively poor choice for fitting both datasets. It had the lowest AIC score of 231.237 for the Riseholme data but its SSE was higher than for other distributions making it the second worse result in the outdoor case. The Normal distribution had the worse fit with the highest SSE values and a poor AIC value of 238 but its BIC value was the lowest of all six distributions with the Riseholme data. This may be because of the gap in the distribution of data that appears in the middle and which affected the overall fit of all distributions and the Normal one in particular. Compared to the homegrown dataset the Normal distribution had a better fit but the small size of this dataset led to all the distributions having a high error value.

Both the distributions for home and Riseholme data are similar with some exceptions. Exponential power is the best fit for the Riseholme data but for the indoor plants it was the second worse fit. Rayleigh is however still a good fit for the home grown plants but Exponential power is not a good fit for the Riseholme data having the second worse fit.

### 6.2.2 Petiole Length

We use the same set of distributions and procedures to fit the petiole length data from the Riseholme farm and those home grown (see Fig. 44). The fitted distributions (see Table 7) had

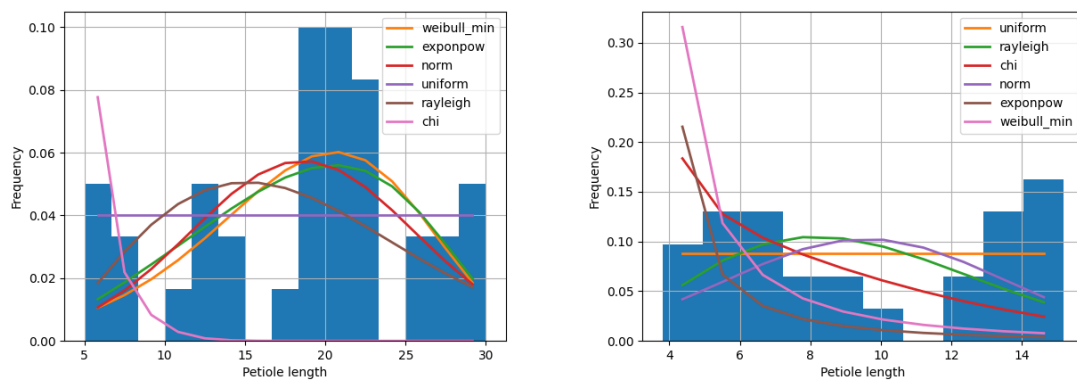
Dist	SSE	AIC	BIC
Expon Pow	0.000065	237.44	-694.60
Rayleigh	0.000066	237.63	-698.29
Chi	0.000066	239.74	-693.97
Weibull Min	0.000068	239.70	-692.76
Uniform	0.000074	231.23	-692.41
Normal	0.000077	238.79	-690.09

**Table 4:** The ranking of the fitted distributions for the Riseholme outdoor plants, from best SSE to worse.

Dist	SSE	AIC	BIC
Rayleigh	0.033633	147.75	-173.98
Chi	0.034276	148.95	-170.17
Normal	0.035067	149.92	-172.85
Uniform	0.038886	138.33	-170.06
Expon Pow	0.054174	229.63	-157.82
Weibull Min	0.059848	201.92	-155.13

**Table 5:** The ranking of the fitted distributions for the home grown plants, from best SSE to worse.

higher errors when compared to the leaf area suggesting that more data may be required to get a more accurate fit.



**Figure 44:** Petiole length distribution models fitted to the data from the Riseholme farm (left) and home indoor plants (right).

The best fit for the Riseholme data was the Weibull distribution and we can see by looking at it that it creates a curve that follows the increase in frequency around the length of 20. Even though Weibull is the best fit its SSE was relatively high when compared to leaf area with a value of 0.028807 which is 3 orders of magnitude higher. The gaps in the petiole length histogram would have increased the error. Having a larger collection of data may have filled in the gaps but also the age and stage of the plant would affect how long its petiole's are. A more shaded plant may have shorter petioles as compared to one getting full sunlight.

The best fit for the home data was Uniform distribution meaning the data is fairly flat with no large spikes of long or short petiole length. From my observation the petiole length did not appear to grow any longer than 15cm unlike with the outdoor Riseholme plants where 30cm was

a consistent peak possibly because of the different varieties and the different growing conditions. The other distributions were not a good fit with Rayleigh the second best fit for the home data but its SSE value was almost double of uniform distribution.

Dist	SSE	AIC	BIC
Weibull Min	0.028807	142.43	-245.95
Expon Pow	0.029303	140.17	-245.33
Uniform	0.030222	132.75	-247.81
Normal	0.030379	140.31	-247.62
Rayleigh	0.033902	139.01	-243.67
Chi	0.060372	569.89	-219.31

**Table 6:** *Riseholme outdoor plants from best to worse fit*

Dist	SSE	AIC	BIC
Uniform	0.023327	52.67	-183.86
Rayleigh	0.040143	56.00	-169.20
Chi	0.041250	60.53	-165.17
Normal	0.045375	56.63	-165.90
Expon Pow	0.076062	87.90	-148.65
Weibull Min	0.095102	75.04	-142.62

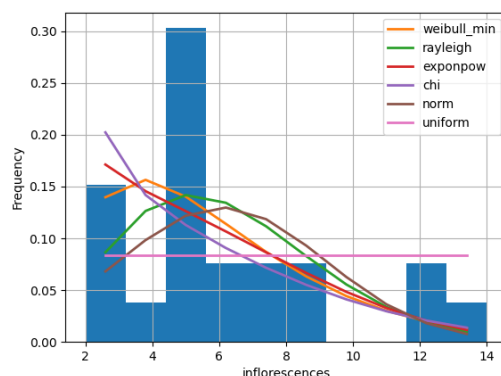
**Table 7:** *Indoor plants plants from best to worse fit*

The second best distribution is Exponential Power which had values close to Weibull, having a higher SSE value but a better AIC value. The AIC value was 140.17 which was just a 2.26 difference. The curve is similar to Weibull but the peak of the curve is lower making the distribution very similar. Its similarity also makes it a viable option to use as a distribution since the difference is so small. Normal distribution had an error of 0.030379 and is similar to Uniform which had a value of 0.030222. The Normal distribution has a curve that is more centered in the data which does not reflect the distribution of the outdoor Riseholme data which leads to it having a weaker fit. Rayleigh distribution creates curves towards the left of the graph unlike Normal or Weibull. This would lead to it having a poorer fit when compared to the other distributions especially with an error of 0.033902 which is a 0.005095 difference when compared to the best fit Weibull. The worse fit distribution was Chi which failed to fit along the data having an error value over double the best fit Weibull at 0.060372. Its AIC was also the worse at 569.89 significantly higher than the other distributions. The gaps and lack of data may have led to the poor distribution similar to the indoor home leaf area plants which also had a couple of distributions also fit incorrectly.

Both Weibull and Exponential power make good fits for the petiole length for Riseholme even with the smaller data set. The gaps and lack of data lead to other distributions having a bad fit. The Uniform distribution had the third best fit despite it being a distribution that produces a flat line. Other environmental variables would affect the petiole length like the amount of sun the plants would receive a day and plants being shaded by each other. For the indoor grown plants the petiole length was more consistent leading to a Uniform distribution with Weibull and Exponential Power having the worse fit. This may be from the different stages of growth that the plants were in and the conditions.

### 6.2.3 Inflorescence

The fruit generation is difficult to implement and evaluate as many factors affect fruit development which are hard to replicate such as pollination, nutrients, and the varying species of plant. The generation used in the proposed simulation takes into account the number of fruit per stem and per plant. To evaluate this functionality, we use the outdoor Riseholme data and note the number of flowers and fruit for each plant. We did not gather any data for the home indoor plants as they only grew a couple of fruit and did not provide enough data. We also used the same fitting procedure with the SSE, AIC and BIC to find the best fit for the inflorescence.



**Figure 45:** *Inflorescence distribution models fitted to the data from the Riseholme farm.*

The best fit for the inflorescence is Weibull which has a curve that similarly matches the distribution of the Riseholme data. There is a high frequency of inflorescence with a few outliers leading to the left-side skewed curve on the graph. Other distributions like Normal have a curve centered between the max and min which does not represent our data well because of the increase in frequency near the lower values.

Rayleigh was also a good distribution having a similar curve to Weibull. It had minor differences in SSE, AIC and BIC values. Rayleigh start of the curve begins lower when compared to Weibull where it starts closer to the frequency of the data at inflorescence values of 2-3. Exponential Power also had a fairly good fit with a slightly worse SSE of 0.051134, only a 0.000411 difference to Rayleigh. Exponential Power creates a decaying line and ignores the peak frequency at inflorescence values of 4-6. Chi creates a curve similar to Exponential Power but starts at higher frequencies. The SSE value is higher than the other top three distributions. Normal distribution also had a poor fit and it can be visually seen with the curve looking misaligned when compared to the histogram and the better fits such as Weibull. Its SSE value was 0.057957 which is similar to Chi which has a value of 0.056393. The Uniform distribution is also not a good fit as it has the worse fit having a high SSE value of 0.071166. This could be caused by the small data sample and also by the spike of frequency between the values of 4-6.

Weibull and Rayleigh make for the best distributions to use with petiole length as they have the lowest SSE values and low AIC and BIC values as well. They also visually match the frequency of the histogram unlike the other distributions. Having more data would help to get a more accurate fit since the SSE was relatively high especially when compared to leaf area of the Riseholme data as the SSE value for expon pow is 0.000065 while Weibull Min for inflorescence was 0.049056 a 99.8% difference. It is worth noting that inflorescence is time dependant and therefore would depend on the harvest season, strawberry species, weather etc.

Dist	SSE	AIC	BIC
Weibull Min	0.049056	62.14	-125.05
Rayleigh	0.051134	60.18	-127.23
Expon Pow	0.051545	61.91	-123.96
Chi	0.056393	63.09	-121.98
Normal	0.057957	61.26	-124.47
Uniform	0.071166	53.69	-119.96

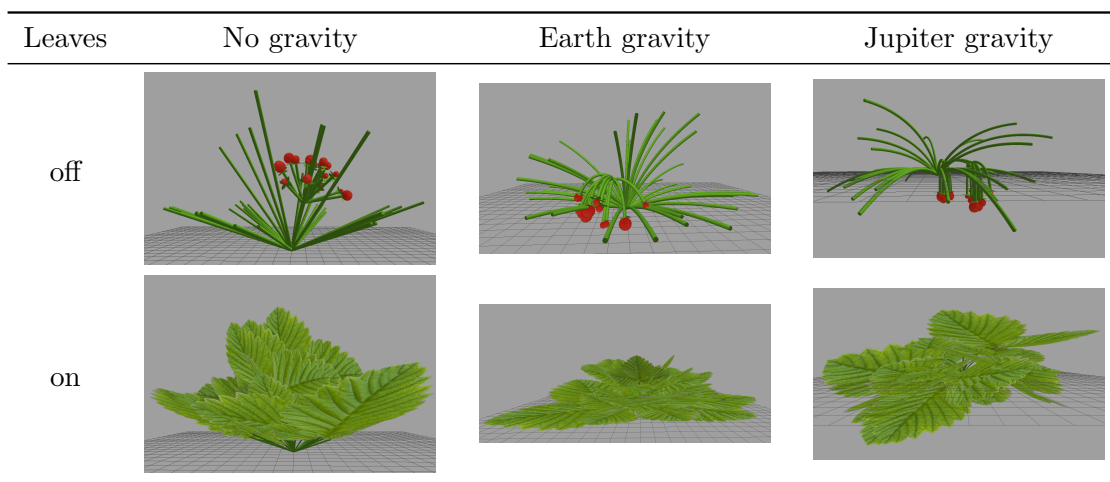
**Table 8:** *Riseholme outdoor plants from best to worse fit in inflorescence*

### 6.3 Evaluation of the Functional Components

Gravity and phototropism are difficult to evaluate quantitatively but we can still assess them visually by using 3D rendering techniques. To showcase the gravity, we compare visually the generated plants, with optimised parameters, with different values of gravity to see how that affects the appearance and structure of the plants. In addition, we also simulated growth on a flat and inclined surface (45-degree angle). Strawberry plant leaves are relatively small in area and therefore their mass is also relatively small with typical average values of 2.7 to 4.2 g/plant, including the roots, leaves, and crown [104].

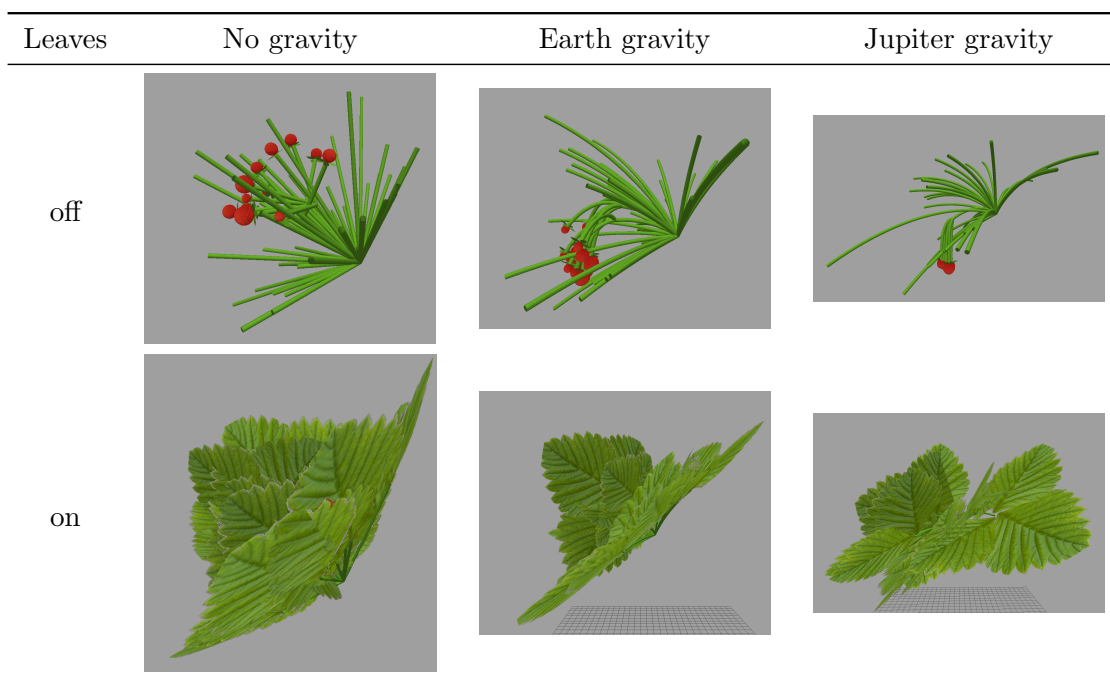
Figure 46 shows the visible effect that gravity has on the growth of the plant. The fruit is the heaviest and therefore pulling down on the stem it grew from. When no gravity is affecting the growth of the plants, then the fruit grows straight in a similar fashion to leaves. Growing the plant with the gravity equivalent of Jupiter (24.79 m/s), we can see the fruit gets pulled straight down while the leaves struggle to grow. While such intense gravity does not exist on Earth, it does showcase how increasing the gravity can affect the growth and possibly theorize how a plant may attempt to grow in such unusual conditions. Plants grown without gravity are not common but such experiments have been done with successful growth of the plants [142].

Disabling gravity causes the leaves and petiole to grow straight in one direction. The fruit also remains in one position showing how much the weight of the fruit influences the way it grows. The lack of gravity shows the larger leaves near the top of the plant rather than being lower to the ground as they usually would from their weight. The general distribution of the leaves is random as before the heavier leaves would tend to rest at the bottom of the plant and have a more significant bend. Attempting to grow strawberries with stronger gravity than the Earth causes major bending of the leaves and fruit. The fruit is dragged straight while most of the leaves struggle to grow upwards.



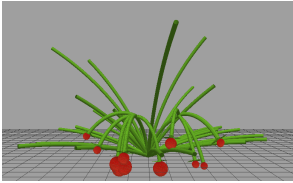
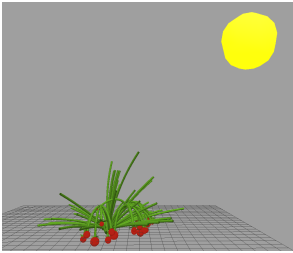
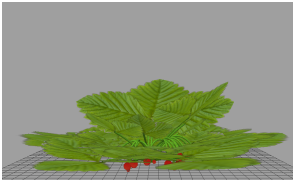
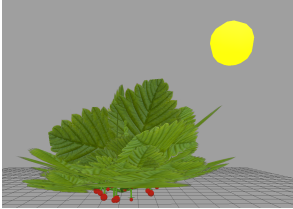
**Figure 46:** Effects of gravity of different strengths on the generated plant models when placed on a flat surface.

Growing the plants at an angle also helps to visualize the effects of gravity in the simulation (see Fig. 47). Without the effects of gravity the plant simply just grows straight, perpendicular to the surface, including the fruit as there is nothing to influence it to bend despite being grown at an angle. Earth gravity shows all the fruit bending to one side where gravity pulls it the most but quite a few of the leaves are still able to grow straight with a slight bend only. Lastly applying a strong gravity causes most of the leaves to be pulled down and especially the fruit which has been all clumped together.



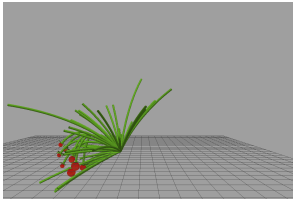
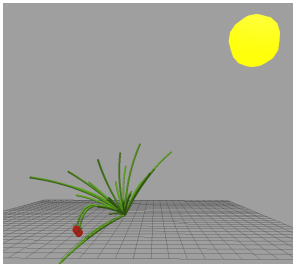
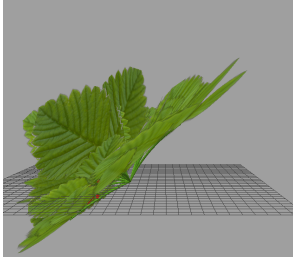
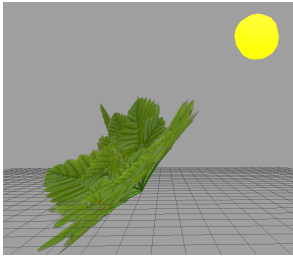
**Figure 47:** Effects of gravity of different strengths on the generated plant models when placed on a 45-degree angle surface.

While gravity remains constant, sunlight changes throughout the year and can be affected by weather. We did not gather data or have the equipment to measure the effects of phototropism at different light levels and light periods. We shall showcase the effects of phototropism in our simulation similar to how we showcased the effects of gravity by enabling and disabling phototropism.

Leaves	No Light	Light
off		
on		

**Figure 48:** *Effects of light on the generated plants when placed on a flat surface.*

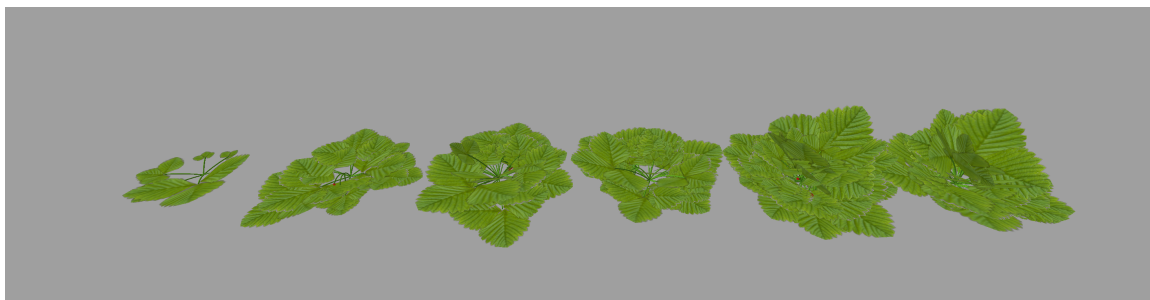
Placing a light when the plants are grown at an angle seen in Fig 49 shows the leaves still bending towards the light despite being at an angle and fighting against gravity. Leaves that have grown on the far side have little influence on the light and therefore follow gravity.

Leaves.	No Light	Light
off		
on		

**Figure 49:** *Effects of light on the generated plants when placed on a 45-degree angle surface.*



### 6.3.1 Plant Growth



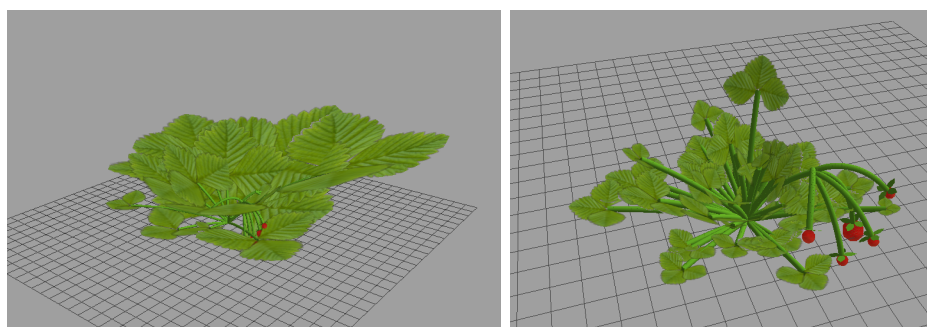
**Figure 50:** *Stages of plant development generated at every month for a period of six months with Earth gravity.*

We have used the optimised distribution parameters for leaf area, petiole length and inflorescence in the final stage of evaluations. The leaf growth stops at 50 leaves which were hardcoded in to prevent the leaves from constantly growing. This was also based on the maximum value from the results found in a study which investigated the leaf growth in different regions [4]. The leaf growth works well and is consistent with real plants.

Fruit generation functionally works with fruit reacting to gravity and being of a certain size. But the main improvement would be to determine fruit based on environmental factors and the plant's current health condition. A smaller plant would grow smaller fruit compared to a big plant other factors that could affect the fruit growth are the number of leaves that is present in the current model. The temperature may also affect the plant and when it will produce fruit.

Gravity works well with the fruit size affecting the bending and the size of the leaf affecting the bend of the petiole. But improvements can be made by accurately knowing the weight of the fruit and understanding the strength of the plant to see how much weight it can take. Phototropism is also a useful addition but it is coupled in code with gravity, therefore it is only possible to switch off both features at the same time. An improvement would be to separate them and have a strength for phototropism to allow it to be influenced alongside gravity.

## 6.4 Evaluation of the Implementation



**Figure 51:** *Generated strawberry plant with the improvements based on best fit for Risholme distribution(left) and Home grown distribution(right)*

The growth model implemented in this simulation is based on temperature, time and gravity and is modifiable. The main growth uses Growth Degree Days to calculate the leaf growth and number of fruit. Gravity influence and phototropism affect the plant by taking in the weight of



the leaves and fruit. The leaf area, temperature and Petiole length are all recorded and can be used to see development over time to predict growth.

One of the editable parameters is being able to create a field of strawberry plants where each are individually generated. The benefit of this is you can predict the growth of multiple plants at once and examine them. Trying to generate a large array of strawberry plants causes framerate drops within the framework. Porting it to a different, more efficient framework may offer better frame rates. Sacrificing some of the details may improve the framerates but this would reduce the accuracy of the plant. The time required to generate all procedural plants does not take long only taking a few seconds even for larger arrays, which is heavily dependant on hardware.

The code written to generate the plants is component-based allowing for sections to be disabled and allow for new components to be added with ease. The primary crown, petiole, leaves and fruit all have their own components which can be modified. It makes it more readable and adaptable for other possible frameworks. The ease of implementing new features like soil nutrition can be done much easier due to its modularity. New components for features like branching crown and runners can be easily created and added into the simulation.

We added some realism to the visuals of the model by incorporating textures for the fruit and leaves of the plant. This makes the plant visually more interesting and easier to compare with real plants. The leaves of a strawberry plant have jagged edges and attempting to build a model of that would be possible but more difficult for the computer to render. Providing a flat image of the leaf makes it easier to render while also providing visual detail. Further improvements can be made to the textures by using more than one leaf texture to add variety. A texture for the stem will also help to provide realism but this can be quite difficult to do for a cylinder like object. Procedurally generating the textures can also be a possible option but the framework lacks support for it but this would provide more variation between each of the models. Blender has procedural textures that can be combined with various noise functions.

## 7 Conclusions and Future Work

In this work considered development and evaluation of a technique capable of randomly generating 3d models of strawberry plants based on certain environment factors which can affect the plant's growth. The model uses days and temperature affect growth of the strawberry plant based on the growth degree days. The leaf area, petiole length and inflorescences were all tuned from data gathered from outdoor and indoor plants using a data fitting python library was used to find the ideal distribution for generating these parameters. Visual appearance of the plant and the way it is grown was observed in the strawberry plants with a timelapse of their growth over time. The proposed system contributes to development of simulations by providing a foundation for procedurally generating strawberry plants with functionality to export the models to be used in different frameworks. The chosen framework L-py is well-documented framework incorporated into many programming languages (e.g. Python, C++) making it easy to understand and implement. The second objective was to look if the generated models could be incorporated with other frameworks (e.g. 3d editors or simulators).

### 7.1 Model Limitations

Whilst the proposed model successfully demonstrates its ability to generate realistic strawberry plants, there are several features that could improve its functionality and quality of the results.

Some important features present in real strawberry plants are not seen in the generated models such as runners/stolon, flowers, and branching crowns. Runners/stolons were investigated and planned to be implemented but due to the limitations of the framework, it became difficult to create daughter plants at the end of stolons as this would require additional information about the precise location of the end points. A chain of plants could be created using runners and it could help predict runner development during the growth of strawberry plants. While strawberry fruit growth is present in the model, it does not have the flowering stage. One obstacle from implementing this feature is the model of the flower that would need to be created which is quite complex. Strawberry plants have a flowering stage and would require further study on when that stage occurs and for how long during growth. Branching crowns, whilst implemented at one stage of model development, were eventually removed to focus on aspects like leaf development and plant growth. It can be difficult to predict crown growth as there would be environmental factors that would affect it. Implementing crown development can help predicting fruit development as well as leaf development more accurately.

Our fruit growth and development does not have pollination since it does not have flowers or a method to determine if the flower has been pollinated. It has an effect on producing fruit and often pollinated by honey bees which has the highest success rate [143] and for allowing crossbreeding between different types of strawberry plants. An implementation of determining if the flower has been pollinated or not can help to improve the simulated yield of the plant and then to go further to introduce the possibility of crossbreeding.

Both gravity and phototropism are hard to evaluate and gather data from. We can visually look at the differences and estimate whether or not we think it is accurate. Acquiring data on average berry weight over a while would be useful to understand how large and heavy a berry can get over some time. More data on the dry weight of leaves alongside the strength of the petiole will also help with understanding how much gravity will influence the plant. The current implementation of phototropism is basic and would need more research and work to make it more accurate. Improvements like energy transfer can be used to determine if the plant is getting sufficient light or too much light. Another would be the photoperiod of the day based on the month and season to examine growth during cold winters and warm summers. A feature to simulate the chaining position of the sun during the day would enable more realistic plant growth rather than just bending towards a static light in the scene.

The model also does not take into account soil nutrition for growth or leaf abscission and decay which may occur from the lack of sunlight when blocked by the other leaves. A more accurate lighting method like ray-casting would help to understand what leaves are receiving sufficient light and which are not. Some of these features like photosynthesis and soil nutrition have been used with L-systems in Job Talle's study on evolution and L-systems [86] which could be utilised in the future strawberry simulation models.

## 7.2 Lpy framework suitability

The Lpy framework is very versatile but still lacking features that would help improve the model's accuracy and its capabilities. There were some features which were inaccessible or difficult to utilise such as precision localisation information for each plant component. This would help with collision detection as the position of each leaf and fruit can be determined. Another feature not used in the proposed model was bounding boxes which would make implementing a collision detection easier because if two boxes collide then the plant can redirect its growth to prevent it from clipping. One last feature that was unavailable was ray-casting for collision detection as this could enhance the phototropism by calculating how much energy transfer has occurred over the period of time rather than currently where the plant just bends towards the light source. The Lpy framework enables exporting the generated model into various file formats such as Polygon File Format (PLY) allowing the model to be used in third-party software such as Blender. However, the PLY format does not support textures or UV maps which results in a grey static model.

## 7.3 Future work

The model provides a foundation for procedural strawberry plants but further enhancements can be made to improve its accuracy. Further improvements to the framework can also help to expand it or a new framework with more features could also be worth investigating. In terms of the model and data used, more real data could be collected to create more accurate synthetic data and more variety of strawberries to accommodate for. L-systems may not be the most flexible solution it is a reliable method and one that can be done within the time frame. While sunlight is an important aspect of plant growth it is difficult to implement due to the complexity of the algorithm and the data needed to understand how the strawberry plants growth is affected. Other aspects like nutrient and soil moisture are also not present for similar reasons. Future implementations are possible as methods and implementations in different frameworks and plants have been shown for soil moisture and lighting [93, 82].

Phototropism is present in the model but it is a basic implementation. A framework that can utilise ray-tracing can be used to calculate light use efficiency for electrical lights [144]. Ray tracing allows for light to be scattered and diffused and so can also be used to calculate the radiation transfer on plant canopies [145]. This is not possible with the current framework, as it utilises ray casting which is not capable of bouncing off surfaces the same way as ray tracing method would. Various game engines now have ray tracing support such as Unreal Engine which uses Nvidia's ray-tracing software. Game engines, however, are not entirely suited for simulations as they are designed to be optimised for visual fidelity and performance rather than accuracy. Various studies have used game engines with L-systems in research but this has mostly been a focus on game content rather than for accurate plant simulations as seen in this thesis where it is used to create landscapes, trees, rocks and water [146]. Another study also uses unreal engine to generate landscapes and environments for video games [147].

3D modeling software like Blender and Houdini offer high quality ray tracing frameworks and physics simulations as well. These focus on visual effects in films with less focus on performance. Blender has been used to visualise and simulate virtual plant canopies [148]. But Blender's most

common use is creating models for other frameworks or methods as seen in this study where Blender is used to create the high quality model before being used in their model [149].

## 8 Appendix

Full L-py code available at <https://github.com/ChrisPHP/L-Strawb-Py>

```

1 import matplotlib.pyplot as plt
2 from openalea.plantgl.all import *
3 import random as rd
4 from numpy import random as rng
5 import decimal
6 import numpy as ny
7 import math
8 import time
9
10 #LeafArea
11 Lf = 0
12
13 #60 days
14
15 Lx = 0
16 Ly = 15
17 Lz = 20
18
19 #Global plant location variable
20 Px = 0
21 Py = 0
22
23 spiral = 0
24
25 #Textures for the plants
26 context().turtle.setMaterial(9,ImageTexture('textures/leaf.png'))
27 context().turtle.setMaterial(8,ImageTexture('textures/stem_drk.png'))
28 context().turtle.setMaterial(10,ImageTexture('textures/strawb.jpeg'))
29
30 def SaveArea(x):
31     global Lf
32     Lf = x
33     #print(Ly)
34
35 #generate a temp for the day
36 def RanTemp(x, max, low):
37     return round(rd.uniform(low[x], max[x]), 1)
38
39 #Number of inflorescences/strawberries on a single stem
40 inflo = [5, 5, 12, 7, 14, 3, 12, 8, 3, 2, 5, 5, 6, 7, 5, 5, 8, 4, 5, 5, 6, 2]
41 Petiole = [28, 29, 23, 21, 5, 22, 5, 22, 30, 20.3, 19, 20, 12, 25, 6.3, 26, 19,
14, 18.3, 19.3, 14.5, 10.1, 13, 19, 22.5, 12, 20, 7, 7, 19, 21, 28, 23, 19,
29, 21]
42
43 #List of mean temperatures from the UK of 2020 and the max and min temps
recorded for that day
44 mean = [5.6, 5.1, 5.6, 9.1, 11.3, 14.0, 14.3, 15.9, 12.9, 9.4, 7.7, 4.3]
45 min = [-7.9, -10.0, -7.6, -6.9, -6.6, -1.9, -0.6, -0.4, -5.0, -3.3, -6.1,
-10.9]
46 max = [15.6, 16.0, 19.4, 26.0, 28.3, 33.4, 37.8, 36.4, 31.3, 19.1, 18.4, 14.9]
47 #List of each month and days (not including leap year)
48 MnthsDays = [[1, 31], [2, 28], [3, 31], [4, 30], [5, 31], [6, 30], [7, 31], [8,
31], [9, 30], [10, 31], [11, 30], [12, 31]]
49 #List to store each temp
50 tempHigh = []
51 tempLow = []
52 #generate sigmoid curve for petiole length
53 def Sigmoid(x, limit, pos):
54     a = []

```

```
55     for item in x:
56         sig = round(5 + (limit/(1+math.exp(pos-item))), 2)
57         a.append(sig)
58     return a
59
60 #Calculate the Growth Degree days
61 def CalGDD(high, low):
62     base = 7
63     if (low < 7):
64         if (high > 30):
65             return 30 + 7 / 2 - base
66         else:
67             return high + 7 / 2 - base
68     elif (low > 30):
69         return 30 + low / 2 - base
70     else:
71         return high + low / 2 - base
72
73
74 #Get Last roll and add new roll in
75 def Spiral():
76     global spiral
77     spiral = spiral + rd.randint(50, 70)
78     return spiral
79
80 #Global variable for plant location print(Lfstore)
81 def PlantLoc(x, y):
82     global Px
83     global Py
84     Px = x
85     Py = y
86
87 def EucDist(x, y, z):
88     A = ny.array((x, y, z))
89     B = ny.array((Lx, Ly, Lz))
90     return ny.linalg.norm(A-B)
91
92 #Compare plant location with the point light location
93 def LightLoc(plant, light):
94     if (Phototropism == True):
95         diff = plant - light
96         if (plant > light):
97             return -1
98         elif (diff <= 5 and diff >= -5):
99             return 0
100        else:
101            return 1
102    else:
103        return 0
104
105 points = ny.arange(0., 8., 0.15)
106 if (SDLD == True):
107     log = Sigmoid(points, 10, 4)
108 else:
109     log = Sigmoid(points, 20, 3)
110
111 for n in range(months):
112     for y in range(MnthsDays[n][1]):
113         #print(days[n][1])
114         tempLow.append(RanTemp(n, mean, min))
115         tempHigh.append(RanTemp(n, max, mean))
116
117 #print(tempLow)
```

```
118
119 GDD = 0
120
121 Days = len(tempHigh)
122 for n in range(len(tempHigh)):
123     GDD = GDD + CalGDD(tempHigh[n], tempLow[n])
124
125 #print(GDD)
126
127 total = GDD
128
129 leafs = round(total / rd.randint(60, 66))
130 if (leafs > 45):
131     leafs = 45
132
133
134 Axiom: Plight() Farm(10, 10, 150/3)
135
136 derivation length: 7
137 production:
138
139 Plight():
140     if (Phototropism == True):
141         nproduce [SetColor(4)@M(Lx, Ly, Lz)@0(3)]
142
143 Farm(rows, columns, dist):
144     st = time.time()
145     x = 0
146     y = 0
147     for a in range(rows):
148
149         #nproduce @M(x, 0, 0)[SetColor(4)PC(leafs)]
150         for b in range(columns):
151             PlantLoc(x, y)
152             #print(EucDist(x, y, 0))
153             #f(10)+(45)
154             nproduce @M(x, y, 0) [SetColor(2)PC(leafs + rd.randint(-5, 5))]
155             y = y + dist
156             y = 0
157             x = x + dist
158         et = time.time()
159         res = et - st
160         print('CPU Execution time:', res, 'seconds')
161
162 #Primary Crown
163 PC(x):
164     #If more than 10 leaves are generated generate fruit
165     if (x >= 10):
166         for a in range(rd.randint(1, 3)):
167             #component to generate the fruit
168             nproduce [Rol()INF(0)]
169     for leaf in range(x):
170         #generate each leaf
171         nproduce [Rol()L(leaf)]
172
173 #Branching Crown
174 BC(x):
175     y = rd.randint(1, 2)
176     if y == 2:
177         for a in range(x):
178             nproduce SetColor(5)+(10)F(0.1)[Rol()L()]
179     else:
180         for a in range(x):
```

```

181     nproduce SetColor(5)-(10)F(0.1)[Ro1()L()
182
183 #Leaf generation
184 LF(x, leaf):
185     #The current leaf
186     T = leafs - leaf
187
188     #Generates leaves with rayleigh distribution based on the mode of the real
189     data.
190     LeafArea = rng.rayleigh(137, 1)
191     #loop to create new values in case the value- generated was above the min or
192     max of the original data
193     while LeafArea < 30.41 or LeafArea > 323.64:
194         LeafArea = rng.rayleigh(137, 1)
195
196     LeafArea = ((LeafArea[0] / 3) / 3) / 3
197     #LeafArea = round(T / 5) + 1
198
199     #Get weight based on the leaf area
200     DryWeight = round(2000 / 1 + float(decimal.Decimal(LeafArea)/5))
201
202     #the older the leaf the bigger the angle
203     angle = T * rd.randint(1, 10)
204     #Cannot exceed 70 degrees
205     if (angle > 70):
206         nproduce +(70)
207     else:
208         nproduce +(angle)
209
210     for a in range(x):
211         #If hair is enabled generate
212         if (Hair == True):
213             nproduce Hr(100)
214             nproduce @Tp(LightLoc(Px, Lx), LightLoc(Py, Ly),-1) Elasticity(abs(tropism)
215             /DryWeight) nF(1, 0.2)
216             nproduce /(90)P(LeafArea)
217
218 #Calculate leaf weight (Unused)
219 LWeight(x):
220     DryWeight = round(100 / x)
221     nproduce @Tp(0,0,-1) Elasticity(abs(tropism)/DryWeight)
222
223 #Generate stem hairs
224 Hr(x):
225     for a in range(x):
226         len = float(decimal.Decimal(rd.randrange(0, 10))/10)
227         nproduce: [SetColor(7)/(rd.randint(0, 360))nf(len, len)+(rd.randint(45,
228         120))_(0.001)@B(0.25)]
229
230 #Fruit
231 INF(x):
232     #Create number of inflorescence based on the mode of Riseholme data using
233     Rayleigh
234     s = rng.rayleigh(5, 1)
235
236     nproduce SetColor(2)/(5)+(15)@Tp(0,0,-1) Elasticity(abs(tropism)/1000.)nF
237     (1.5, 0.1)[nF(2, 0.1)Berries(int(s))]
238
239 #Berry generation
240 Berries(x):
241     for a in range(x):
242         size = float(decimal.Decimal(rd.randrange(2, 5))/10)

```



```

238     nproduce /(rd.randint(-360, 360))[BWeight(size)+(rd.randint(0, 75))nF(3,
239     0.1) BerryLeaf() f(size) SetColor(10)TextureScale(1)@0(size)]
240 BerryLeaf():
241     nproduce [(+90)/(90)~1(0.5)] [-(90)/(90)~1(0.5)] [(90)+(90) /(90)~1(0.5)]
242     [(90)-(90) /(90)~1(0.5)]
243 BWeight(x):
244     Size = round(50 / x)
245     nproduce @Tp(0,0,-1) Elasticity(abs(tropism)/Size)
246
247 #Stolon/Runner generation
248 ST(x):
249     for a in range(x):
250         nproduce +(5)@Tp(0,0,-1) Elasticity(abs(tropism)/1000.)nF(1, 0.1)
251
252 #Leaf polygon generation
253 L(x):
254     #generate petiole length
255     petiole = rng.uniform(ny.min(Petiole), ny.max(Petiole), 1)
256
257     #set the colour and scale the texture
258     nproduce SetColor(2) TextureScale(1)LF(round(petiole[0] / 3), x)
259
260 og(x):
261     nproduce ~1(x/2+0.5)+(90)~1(x/2+0.5)-(180)~1(x/2+0.5)
262
263 #Component to generate the trifoliate
264 P(x):
265     nproduce SetColor(9) TextureRotation(-90) [(90)+(5)f(x/2)/(-90)@o(x/2)]
266     TextureRotation(-90) +(90) [(90)+(15)f(x/2)/(-90)@o(x/2)] -(180)/(90)+(15)f
267     (x/2)/(-90)@o(x/2)
268
269 Spr():
270     nproduce /(rd.randint(-360, 360))
271
272 #Spiral generation
273 Rol():
274     nproduce F(0.05)/(Spiral())
275
276 endlsystem

```

## References

- [1] James F Hancock. *Strawberries*, volume 34. CABI, 2020.
- [2] CG Guttridge. Observations on the shoot growth of the cultivated strawberry plant. *Journal of Horticultural Science*, 30(1):1–11, 1955.
- [3] Kikas Ave, Arus Liina, Kaldmäe Hedi, and Libek Asta-Virve. Newly introduced strawberry genotypes for nordic baltic conditions. *Horticultural Science*, 44(3):141–147, 2017.
- [4] Christopher M Menzel. No evidence of excessive leaf production by strawberries grown in the subtropics. *Agriculture*, 9(9):197, 2019.
- [5] SE Arney. Studies in growth and development in the genus fragaria i. factors affecting the rate of leaf production in royal sovereign strawberry. *Journal of Horticultural Science*, 28(2):73–84, 1953.
- [6] Kenneth J Boote, James W Jones, and Nigel B Pickering. Potential uses and limitations of crop models. *Agronomy journal*, 88(5):704–716, 1996.
- [7] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1):1–22, 2013.
- [8] Adrian Salazar Gomez, Erchan Aptoula, Simon Parsons, and Petra Bosilj. Deep regression versus detection for counting in robotic phenotyping. *IEEE Robotics and Automation Letters*, 6(2):2902–2907, 2021.
- [9] Sariah Mghames, Marc Hanheide, and Amir Ghalamzan. Interactive movement primitives: Planning to push occluding pieces for fruit picking. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2616–2623. IEEE, 2020.
- [10] Tsvetan Zhivkov, Adrian Gomez, Junfeng Gao, Elizabeth Sklar, Simon Parsons, et al. The need for speed: How 5g communication can support ai in the field. 2021.
- [11] Adam Binch, Gautham P Das, Jaime Pulido Fentanes, and Marc Hanheide. Context dependant iterative parameter optimisation for robust robot navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3937–3943. IEEE, 2020.
- [12] David Simpson. The economic importance of strawberry crops. In *The genomes of rosaceous berries and their wild relatives*, pages 1–7. Springer, 2018.
- [13] P. WERNECKE, J. MÜLLER, T. Dornbusch, A. WERNECKE, and Wulf Diepenbrock. The virtual crop-modelling system ‘vica’ specified for barley. 01 2007.
- [14] Eike Luedeling, Philip J Smethurst, Frédéric Baudron, Jules Bayala, Neil I Huth, Meine Van Noordwijk, Chin K Ong, Rachmat Mulia, Betha Lusiana, Catherine Muthuri, et al. Field-scale modeling of tree–crop interactions: Challenges and development needs. *Agricultural Systems*, 142:51–69, 2016.
- [15] Kun Zhou, Peng Du, Lifeng Wang, Yasuyuki Matsushita, Jiaoying Shi, Baining Guo, and Heung-Yeung Shum. Decorating surfaces with bidirectional texture functions. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):519–528, 2005.

- [16] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.
- [17] Ruoxi Sun, Jinyuan Jia, Hongyu Li, and Marc Jaeger. Image-based lightweight tree modeling. In *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry*, pages 17–22, 2009.
- [18] Jean-Eudes Marvie, Julien Perret, and Kadi Bouatouch. The fl-system: a functional l-system for procedural geometric modeling. *The Visual Computer*, 21(5):329–339, 2005.
- [19] Juliana Freitas Santos Gomes and Fabiana Rodrigues Leta. Applications of computer vision techniques in the agriculture and food industry: a review. *European Food Research and Technology*, 235(6):989–1000, 2012.
- [20] Baohua Zhang, Wenqian Huang, Jiangbo Li, Chunjiang Zhao, Shuxiang Fan, Jitao Wu, and Chengliang Liu. Principles, developments and applications of computer vision for external quality inspection of fruits and vegetables: A review. *Food Research International*, 62:326–343, 2014.
- [21] Xu Liming and Zhao Yanchao. Automated strawberry grading system based on image processing. *Computers and Electronics in Agriculture*, 71:S32–S39, 2010.
- [22] Yousef Al Ohali. Computer vision based date fruit grading system: Design and implementation. *Journal of King Saud University-Computer and Information Sciences*, 23(1):29–36, 2011.
- [23] Sergio Cubero, Nuria Aleixos, Francisco Albert, Antonio Torregrosa, Coral Ortiz, O García-Navarrete, and José Blasco. Optimised computer vision system for automatic pre-grading of citrus fruit in the field using a mobile platform. *Precision Agriculture*, 15(1):80–94, 2014.
- [24] Tadhg Brosnan and Da-Wen Sun. Improving quality inspection of food products by computer vision—a review. *Journal of food engineering*, 61(1):3–16, 2004.
- [25] Longsheng Fu, Fangfang Gao, Jingzhu Wu, Rui Li, Manoj Karkee, and Qin Zhang. Application of consumer rgb-d cameras for fruit detection and localization in field: A critical review. *Computers and Electronics in Agriculture*, 177:105687, 2020.
- [26] Sajad Sabzi, Yousef Abbaspour-Gilandeh, and Juan Ignacio Arribas. An automatic visible-range video weed detection, segmentation and classification prototype in potato field. *Heliyon*, 6(5):e03685, 2020.
- [27] Xavier P Burgos-Artizzu, Angela Ribeiro, Alberto Tellaeche, Gonzalo Pajares, and Cesar Fernández-Quintanilla. Analysis of natural images processing for the extraction of agricultural elements. *Image and Vision Computing*, 28(1):138–149, 2010.
- [28] Ch Gée, Jérémie Bossu, Gawain Jones, and Frederic Truchetet. Crop/weed discrimination in perspective agronomic images. *Computers and Electronics in Agriculture*, 60(1):49–59, 2008.
- [29] Zhenbo Li, Ruohao Guo, Meng Li, Yaru Chen, and Guangyao Li. A review of computer vision technologies for plant phenotyping. *Computers and Electronics in Agriculture*, 176:105672, 2020.
- [30] Akshay L Chandra, Sai Vikas Desai, Wei Guo, and Vineeth N Balasubramanian. Computer vision with deep learning for plant phenotyping in agriculture: A survey. *arXiv preprint arXiv:2006.11391*, 2020.

- [31] Paul Albert, Mohamed Saadeldin, Badri Narayanan, Brian Mac Namee, Deirdre Hennessy, Aisling O'Connor, Noel O'Connor, and Kevin McGuinness. Semi-supervised dry herbage mass estimation using automatic data and synthetic images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1284–1293, 2021.
- [32] Sylvain Jay, Gilles Rabatel, Xavier Hadoux, Daniel Moura, and Nathalie Gorretta. In-field crop row phenotyping from 3d modeling performed using structure from motion. *Computers and Electronics in Agriculture*, 110:70–77, 2015.
- [33] Thiago T Santos and Alberto A De Oliveira. Image-based 3d digitizing for plant architecture analysis and phenotyping. In *Embrapa Informática Agropecuária-Artigo em anais de congresso (ALICE)*. In: CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES, 25., 2012, Ouro Preto . . . , 2012.
- [34] Siddharth Srivastava, Swati Bhugra, Brejesh Lall, and Santanu Chaudhury. Drought stress classification using 3d plant models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.
- [35] Steve Borkman, Adam Crespi, Saurav Dhakad, Sujoy Ganguly, Jonathan Hogins, You-Cyuan Jhang, Mohsen Kamalzadeh, Bowen Li, Steven Leal, Pete Parisi, et al. Unity perception: Generate synthetic data for computer vision. *arXiv preprint arXiv:2107.04259*, 2021.
- [36] Weichao Qiu and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine. In *European Conference on Computer Vision*, pages 909–916. Springer, 2016.
- [37] Daniel Ward, Peyman Moghadam, and Nicolas Hudson. Deep leaf segmentation using synthetic data. *arXiv preprint arXiv:1807.10931*, 2018.
- [38] Hoang-An Le, Thomas Mensink, Partha Das, Sezer Karaoglu, and Theo Gevers. Eden: Multimodal synthetic dataset of enclosed garden scenes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1579–1589, 2021.
- [39] Matthias Müller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. Sim4cv: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision*, 126(9):902–919, 2018.
- [40] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.
- [41] Adrian Boeing and Thomas Bräunl. Evaluation of real-time physics simulation systems. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 281–288, 2007.
- [42] Joshua P Vandenbrink and John Z Kiss. Plant responses to gravity. In *Seminars in cell & developmental biology*, volume 92, pages 122–125. Elsevier, 2019.
- [43] E.S. Deutsch, E.W. Bork, and W.D. Willms. Soil moisture and plant growth responses to litter and defoliation impacts in parkland grasslands. *Agriculture, Ecosystems & Environment*, 135(1/2):1 – 9, 2010.
- [44] Anne-Sophie Fiorucci and Christian Fankhauser. Plant strategies for enhancing access to sunlight. *Current Biology*, 27(17):R931–R940, 2017.

- [45] Sang Gyu Lee, Sung Kyeom Kim, Hee Ju Lee, Hee Su Lee, and Jin Hyoun Lee. Impact of moderate and extreme climate change scenarios on growth, morphological features, photosynthesis, and fruit production of hot pepper. *Ecology and evolution*, 8(1):197–206, 2018.
- [46] JB Evers, J Vos, X Yin, P Romero, PEL Van Der Putten, and PC Struik. Simulation of wheat growth and development based on organ-level photosynthesis and assimilate allocation. *Journal of Experimental Botany*, 61(8):2203–2216, 2010.
- [47] GJ O’Leary and DJ Connor. A simulation study of wheat crop response to water supply, nitrogen nutrition, stubble retention, and tillage. *Australian Journal of Agricultural Research*, 49(1):11–20, 1998.
- [48] Amy Marshall-Colon, Stephen P Long, Douglas K Allen, Gabrielle Allen, Daniel A Beard, Bedrich Benes, Susanne Von Caemmerer, AJ Christensen, Donna J Cox, John C Hart, et al. Crops in silico: generating virtual crops using an integrative and multi-scale modeling platform. *Frontiers in plant science*, 8:786, 2017.
- [49] Apostolia Tsirikoglou, Joel Kronander, Magnus Wrenninge, and Jonas Unger. Procedural modeling and physically based rendering for synthetic data generation in automotive applications. *arXiv preprint arXiv:1710.06270*, 2017.
- [50] Cesar Roberto de Souza, Adrien Gaidon, Yohann Cabon, and Antonio Manuel Lopez. Procedural generation of videos to train deep action recognition networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4757–4767, 2017.
- [51] Sebastian Risi and Julian Togelius. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, 2(8):428–436, 2020.
- [52] Alessio Gambi, Marc Mueller, and Gordon Fraser. Automatically testing self-driving cars with search-based procedural content generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 318–328, 2019.
- [53] James Arnold and Rob Alexander. Testing autonomous robot control software using procedural content generation. In *International Conference on Computer Safety, Reliability, and Security*, pages 33–44. Springer, 2013.
- [54] VSR Veeravasaru, Constantin Rothkopf, and Ramesh Visvanathan. Model-driven simulations for computer vision. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1063–1071. IEEE, 2017.
- [55] Benny Onrust, Rafael Bidarra, Robert Rooseboom, and Johan van de Koppel. Procedural generation and interactive web visualization of natural environments. In *Proceedings of the 20th International Conference on 3D Web Technology*, pages 133–141, 2015.
- [56] Bethesda Softworks. The elder scrolls ii: Daggerfall. [CD-ROM], 1996.
- [57] Stefan Greuter, Jeremy Parker, Nigel Stewart, and Geoff Leach. Real-time procedural generation of pseudo infinite cities. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 87–ff, 2003.
- [58] Julian Kenwood, James Gain, and Patrick Marais. Efficient procedural generation of forests. 2014.

- [59] Gillian Smith. The future of procedural content generation in games. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2014.
- [60] Rafael Pereira de Araujo and Virginia Tiradentes Souto. Game worlds and creativity: The challenges of procedural content generation. In *International Conference of Design, User Experience, and Usability*, pages 443–455. Springer, 2017.
- [61] R. van der Linden, R. Lopes, and R. Bidarra. Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(1):78–89, 2014.
- [62] Christoph Salge, Michael Cerny Green, Rodgrigo Canaan, and Julian Togelius. Generative design in minecraft (gdmc) settlement generation competition. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, pages 1–10, 2018.
- [63] Jonathon Doran and Ian Parberry. Controlled procedural terrain generation using software agents. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):111–119, 2010.
- [64] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.
- [65] Simon Green. Implementing improved perlin noise. *GPU Gems*, 2:409–416, 2005.
- [66] Adam Runions, Martin Fuhrer, Brendan Lane, Pavol Federl, Anne-Gaëlle Rolland-Lagan, and Przemyslaw Prusinkiewicz. Modeling and visualization of leaf venation patterns. In *ACM SIGGRAPH 2005 Papers*, pages 702–711. 2005.
- [67] Adam Runions, Brendan Lane, and Przemyslaw Prusinkiewicz. Modeling trees with a space colonization algorithm. *NPH*, 7:63–70, 2007.
- [68] Yanchao Liu, Jianwei Guo, Bedrich Benes, Oliver Deussen, Xiaopeng Zhang, and Hui Huang. Treepartnet: neural decomposition of point clouds for 3d tree reconstruction. *ACM Transactions on Graphics*, 40(6), 2021.
- [69] Derek Bradley, Derek Nowrouzezahrai, and Paul Beardsley. Image-based reconstruction and synthesis of dense foliage. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013.
- [70] RB Harvey. Growth of plants in artificial light. *Botanical Gazette*, 74(4):447–451, 1922.
- [71] Jonas Freiknecht and Wolfgang Effelsberg. A survey on the procedural generation of virtual worlds. *Multimodal Technologies and Interaction*, 1(4):27, 2017.
- [72] Xuhao Eric Chen and Brian J Ross. Deep neural network guided evolution of l-system trees. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 2507–2514. IEEE, 2021.
- [73] Michael Hansmeyer. L-systems in architecture. *Distinguishing Digital Architecture in 6th Far Eastern International Digital Architecture Design Award, Taiwan*, 2003.
- [74] Yotam Livny, Feilong Yan, Matt Olson, Baoquan Chen, Hao Zhang, and Jihad El-Sana. Automatic reconstruction of tree skeletal structures from point clouds. In *ACM SIGGRAPH Asia 2010 papers*, pages 1–8. 2010.

- [75] Francisco Yandun, Abhisesh Silwal, and George Kantor. Visual 3d reconstruction and dynamic simulation of fruit trees for robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 54–55, 2020.
- [76] Alan Gara, Matthias A Blumrich, Dong Chen, GL-T Chiu, Paul Coteus, Mark E Giampapa, Ruud A Haring, Philip Heidelberger, Dirk Hoenicke, Gerard V Kopcsay, et al. Overview of the blue gene/l system architecture. *IBM Journal of research and development*, 49(2.3):195–212, 2005.
- [77] Christian Fournier and Bruno Andrieu. Adel-maize: an l-system based model for the integration of growth processes from the organ to the canopy. application to regulation of morphogenesis by light availability. *Agronomie*, 19(3-4):313–327, 1999.
- [78] Jim Hanan. Virtual plants—integrating architectural and physiological models. *Environmental Modelling & Software*, 12(1):35–42, 1997.
- [79] W. Ding, H. Jin, Z. Cheng, and Q. Chen. A visualization system for tomato plant modeling. In *2011 Eighth International Conference Computer Graphics, Imaging and Visualization*, pages 160–165, 2011.
- [80] Arne Pommerening, Guillermo Trincado, Christian Salas-Eljatib, and Harold Burkhart. Understanding and modelling the dynamics of data point clouds of relative growth rate and plant size. *Forest Ecology and Management*, 529:120652, 2023.
- [81] Frans P Boogaard, Eldert J Van Henten, and Gert Kootstra. Improved point-cloud segmentation for plant phenotyping through class-dependent sampling of training data to battle class imbalance. *Frontiers in plant science*, 13:838190–838190, 2022.
- [82] Cyril Soler, François X Sillion, Frédéric Blaise, and Philippe Dereffye. An efficient instantiation algorithm for simulating radiant energy transfer in plant models. *ACM Transactions on Graphics (TOG)*, 22(2):204–233, 2003.
- [83] Dae Ho Jung, Joon Woo Lee, Woo Hyun Kang, In Ha Hwang, and Jung Eek Son. Estimation of whole plant photosynthetic rate of irwin mango under artificial and natural lights using a three-dimensional plant model and ray-tracing. *International journal of molecular sciences*, 19(1):152, 2018.
- [84] Jiyong Shin, Inha Hwang, Dongpil Kim, Taewon Moon, Jaewoo Kim, Woo Hyun Kang, and Jung Eek Son. Evaluation of the light profile and carbon assimilation of tomato plants in greenhouses with respect to film diffuseness and regional solar radiation using ray-tracing simulation. *Agricultural and Forest Meteorology*, 296:108219, 2021.
- [85] Fabio Fiorani and Ulrich Schurr. Future scenarios for plant phenotyping. *Annual review of plant biology*, 64:267–291, 2013.
- [86] Job Talle and Jiří Kosinka. Evolving l-systems in a competitive environment. In *Computer Graphics International Conference*, pages 326–350. Springer, 2020.
- [87] Jon McCormack et al. Interactive evolution of l-system grammars for computer graphics modelling. *Complex Systems: from biology to computation*, pages 118–130, 1993.
- [88] Christian Jacob. Genetic l-system programming. In *International Conference on Parallel Problem Solving from Nature*, pages 333–343. Springer, 1994.

- [89] Sebastian Risi, Kasper Stoy, Andres Faína, and Frank Veenstra. Generating artificial plant morphologies for function and aesthetics through evolving l-systems. In *ALIFE 2016, the Fifteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 692–699. MIT Press, 2016.
- [90] Stefan Paulus, Jan Dupuis, Anne-Katrin Mahlein, and Heiner Kuhlmann. Surface feature based classification of plant organs from 3d laserscanned point clouds for plant phenotyping. *BMC bioinformatics*, 14(1):1–12, 2013.
- [91] Stefan Paulus. Measuring crops in 3d: using geometry for plant phenotyping. *Plant Methods*, 15(1):1–13, 2019.
- [92] Christine Holdredge, Mark D Bertness, Eric Von Wettberg, and Brian R Silliman. Nutrient enrichment enhances hidden differences in phenotype to drive a cryptic plant invasion. *Oikos*, 119(11):1776–1784, 2010.
- [93] Dallan R Prince, Marianne E Fletcher, Chen Shen, and Thomas H Fletcher. Application of l-systems to geometrical construction of chamise and juniper shrubs. *Ecological modelling*, 273:86–95, 2014.
- [94] ES Deutsch, EW Bork, and WD Willms. Soil moisture and plant growth responses to litter and defoliation impacts in parkland grasslands. *Agriculture, ecosystems & environment*, 135(1-2):1–9, 2010.
- [95] Shenglian Lu, Xinyu Guo, Weiliang Wen, Boxiang Xiao, Chuanyu Wang, Jianjun Du, and Chunjiang Zhao. Plantcad: an integrated graphic toolkit for modelling and analyzing plant structure. In *2014 IEEE International Conference on Progress in Informatics and Computing*, pages 378–384. IEEE, 2014.
- [96] Christian Fournier and Bruno Andrieu. A 3d architectural and process-based model of maize development. *Annals of botany*, 81(2):233–250, 1998.
- [97] Hongjun Li, Xiaopeng Zhang, Weiliang Meng, and Lin Ge. Visualization of tomato growth based on dry matter flow. *International Journal of Computer Games Technology*, 2017, 2017.
- [98] Carine Cocco, Jerônimo Luiz Andriolo, Francieli Lima Cardoso, Ligia Erpen, and Odair José Schmitt. Crown size and transplant type on the strawberry yield. *Scientia Agricola*, 68(4):489–493, 2011.
- [99] Otto L Jahn and Malcolm N Dana. Crown and inflorescence development in the strawberry, *fragaria ananassa*. *American Journal of Botany*, 57(6Part1):605–612, 1970.
- [100] G Savini, D Neri, F Zucconi, and N Sugiyama. Strawberry growth and flowering: an architectural model. *International Journal of Fruit Science*, 5(1):29–50, 2005.
- [101] George M Darrow. Inflorescence types of strawberry varieties. *American Journal of Botany*, pages 571–585, 1929.
- [102] Hiroyuki Miura, Mio Yoshida, and Atushi Yamasaki. Effect of temperature on the size of strawberry fruit. *Journal of the Japanese Society for Horticultural Science*, 62(4):769–774, 1994.
- [103] Gianluca Savini, Veronica Giorgi, Emanuela Scarano, and Davide Neri. Strawberry plant relationship through the stolon. *Physiologia Plantarum*, 134(3):421–429, 2008.



- [104] Christopher M Menzel and Lindsay Smith. The growth and productivity of ‘festival’strawberry plants growing in a subtropical environment. *New Zealand journal of crop and horticultural science*, 42(1):60–75, 2014.
- [105] Jeffrey P Hill and Elizabeth M Lord. A method for determining plastochron indices during heteroblastic shoot growth. *American journal of botany*, 77(11):1491–1497, 1990.
- [106] M Spiegelman and Patricia L Dudley. Morphological stages of regeneration in the planarian *dugesia tigrina*: A light and electron microscopic study. *Journal of morphology*, 139(2):155–183, 1973.
- [107] Anita Sønsteby and Ola M Heide. Dormancy relations and flowering of the strawberry cultivars korona and elsanta as influenced by photoperiod and temperature. *Scientia Horticulturae*, 110(1):57–67, 2006.
- [108] Daphne Vince-Prue, CG Guttridge, and MW Buck. Photocontrol of petiole elongation in light-grown strawberry plants. *Planta*, 131(2):109–114, 1976.
- [109] Charles Darwin. *The power of movement in plants*. Appleton, 1897.
- [110] Rujin Chen, Elizabeth Rosen, and Patrick H Masson. Gravitropism in higher plants. *Plant physiology*, 120(2):343–350, 1999.
- [111] Karl J Niklas. The influence of gravity and wind on land plant evolution. *Review of Palaeobotany and Palynology*, 102(1-2):1–14, 1998.
- [112] Mário Sérgio Galhiane, Sandra Regina Rissato, Lucídio de Sousa Santos, Gilberto Orivaldo Chierice, Marcos Vinícius de Almeida, Terezinha Fumis, Inês Chechim, and Aloísio Costa Sampaio. Evaluation of the performance of a castor-oil based formulation in limiting pesticide residues in strawberry crops. *Química Nova*, 35:341–347, 2012.
- [113] Thomas A McMahon. The mechanical design of trees. *Scientific American*, 233(1):92–103, 1975.
- [114] Tim Hohm, Tobias Preuten, and Christian Fankhauser. Phototropism: translating light into directional growth. *American journal of botany*, 100(1):47–59, 2013.
- [115] María Victoria Díaz-Galián, Magdalena Torres, Jose David Sanchez-Pagán, Pedro J Navarro, Julia Weiss, and Marcos Egea-Cortines. Enhancement of strawberry production and fruit quality by blue and red led lights in research and commercial greenhouses. *South African Journal of Botany*, 140:269–275, 2021.
- [116] Tegan Armarego-Marriott, Omar Sandoval-Ibañez, and Lucja Kowalewska. Beyond the darkness: recent lessons from etiolation and de-etiolation studies. *Journal of experimental botany*, 71(4):1215–1225, 2020.
- [117] Juliane M Henschel, Juliano TV Resende, Patrícia C Giloni-Lima, André R Zeist, Renato B Lima, and Matheus H Santos. Production and quality of strawberry cultivated under different colors of low tunnel cover. *Horticultura Brasileira*, 35:364–370, 2017.
- [118] Jason Weber and Joseph Penn. Creation and rendering of realistic trees. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 119–128, 1995.
- [119] Tomas Plachetka. Pov ray: persistence of vision parallel raytracer. In *Proc. of Spring Conf. on Computer Graphics, Budmerice, Slovakia*, volume 123, page 129, 1998.

- [120] Armando de la Re, Francisco Abad, Emilio Camahort, and M Carmen Juan. Tools for procedural generation of plants in virtual scenes. In *International Conference on Computational Science*, pages 801–810. Springer, 2009.
- [121] Disney animation studios: Zootopia, Aug 2016.
- [122] Jonah Hall. Pushing the limits of l-systems for time-lapse vine growth in” the time machine”. In *ACM SIGGRAPH 2002 conference abstracts and applications*, pages 234–234, 2002.
- [123] Eka Wahyu Hidayat, Ida Ayu Dwi Giriantari, Made Sudarma, and I Made Oka Widyantara. Visualization of a three-dimensional tree modeling using fractal based on l-system. In *2020 IEEE International Conference on Sustainable Engineering and Creative Computing (ICSECC)*, pages 418–422. IEEE, 2020.
- [124] Alexander Schwank, Callum James James, and Tony Micilotta. The trees of the jungle book. In *ACM SIGGRAPH 2016 Talks*, pages 1–2. 2016.
- [125] Zhi Yong Bai and Xin Yuan Huang. Plum tree visualization based on speedtree. In *Key Engineering Materials*, volume 474, pages 511–516. Trans Tech Publ, 2011.
- [126] Qing Xiong and Xin-yuan Huang. Speed tree-based forest simulation system. In *2010 International Conference on Electrical and Control Engineering*, pages 3033–3036. IEEE, 2010.
- [127] Frédéric Boudon, Christophe Pradal, Thomas Cokelaer, Przemyslaw Prusinkiewicz, and Christophe Godin. L-py: an l-system simulation framework for modeling plant architecture development based on a dynamic language. *Frontiers in plant science*, 3:76, 2012.
- [128] Christophe Pradal, Samuel Dufour-Kowalski, Frédéric Boudon, Christian Fournier, and Christophe Godin. Openalea: a visual programming and component-based software platform for plant modelling. *Functional plant biology*, 35(10):751–760, 2008.
- [129] Przemyslaw Prusinkiewicz, Radoslaw Karwowski, Radomír Měch, and Jim Hanan. L-studio/cpfg: a software system for modeling plants. In *International Workshop on Applications of Graph Transformations with Industrial Relevance*, pages 457–464. Springer, 1999.
- [130] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 313–324, 1996.
- [131] Niko Leopold. Algorithmische botanik durch lindenmayer systeme in blender.
- [132] Przemyslaw Prusinkiewicz, Radoslaw Karwowski, Brendan Lane, and J Vos. The l+ c plant-modelling language. *Functional-structural plant modelling in crop production*, 22:27–42, 2007.
- [133] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.
- [134] Weather and climate change.
- [135] Hüsni Demirsoy, Leyla Demirsoy, and Ahmet Öztürk. Improved model for the non-destructive estimation of strawberry leaf area. *Fruits*, 60(1):69–73, 2005.

- [136] C Francone, V Pagani, M Foi, G Cappelli, and R Confalonieri. Comparison of leaf area index estimates by ceptometer and pocketlai smart app in canopies with different structures. *Field Crops Research*, 155:38–41, 2014.
- [137] Eli Bressert. Scipy and numpy: an overview for developers. 2012.
- [138] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.
- [139] Kadierdan Kaheman, Steven L Brunton, and J Nathan Kutz. Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data. *Machine Learning: Science and Technology*, 3(1):015031, 2022.
- [140] Yosiyuki Sakamoto, Makio Ishiguro, and Genshiro Kitagawa. Akaike information criterion statistics. *Dordrecht, The Netherlands: D. Reidel*, 81(10.5555):26853, 1986.
- [141] Andrew A Neath and Joseph E Cavanaugh. The bayesian information criterion: background, derivation, and applications. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(2):199–203, 2012.
- [142] Anna-Lisa Paul, Claire E Amalfitano, and Robert J Ferl. Plant growth strategies are remodeled by spaceflight. *BMC plant biology*, 12(1):1–15, 2012.
- [143] Shahera Talat Zaitoun, AA Al-Ghzawi, Hail Kamel Shannag, and Abdel Rahman M Al-Tawaha. Comparative study on the pollination of strawberry by bumble bees and honeybees under plastic house conditions in jordan valley. *Journal of Food Agriculture and Environment*, 4(2):237, 2006.
- [144] Jaewoo Kim, Woo Hyun Kang, and Jung Eek Son. Interpretation and evaluation of electrical lighting in plant factories with ray-tracing simulation and 3d plant modeling. *Agronomy*, 10(10):1545, 2020.
- [145] BN Bailey, Matthew Overby, Pete Willemsen, ER Pardyjak, WF Mahaffee, and Rob Stoll. A scalable plant-resolving radiative transfer model based on optimized gpu ray tracing. *Agricultural and forest meteorology*, 198:192–208, 2014.
- [146] Sebastian Ekman, Anders Hansson, Thomas Högberg, Anna Nylander, Alma Otteadag, and Joakim Thorén. Seamscape a procedural generation system for accelerated creation of 3d landscapes. B.S. thesis, 2016.
- [147] Oscar Blomqvist, Pierre Kraft, Hampus Lidin, Rimmer Motzheim, Adam Tonderski, and Gabriel Wagner. Generating compelling procedural 3d environments and landscapes. B.S. thesis, 2016.
- [148] Lukas Roth, Helge Aasen, Achim Walter, and Frank Liebisch. Extracting leaf area index using viewing geometry effects—a new perspective on high-resolution unmanned aerial system photography. *ISPRS Journal of Photogrammetry and Remote Sensing*, 141:161–175, 2018.
- [149] Weiwei Liu, Jonathan Mark Atherton, Matti Mottus, Alasdair MacArthur, Hakala Teemu, Kadmiel Maseyk, Iain Robinson, Eija Honkavaara, Juan Alberto Porcar Castell, et al. Upscaling of solar induced chlorophyll fluorescence from leaf to canopy using the dart model and a realistic 3d forest scene. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2017.