Influence of the underwater environment in the procedural generation of marine alga *Asparagopsis Armata*

Nelson Rodrigues
FEUP
Porto, Portugal
up200705576@edu.fe.up.pt
https://orcid.org/0000-0002-0519-7151

António Augusto Sousa
FEUP / INESC TEC
Porto, Portugal
aas@fe.up.pt
https://orcid.org/0000-0002-9883-2686

Rui Rodrigues
FEUP / INESC TEC
Porto, Portugal
rui.rodrigues@fe.up.pt
https://orcid.org/0000-0003-4883-1375

António Coelho
FEUP / INESC TEC
Porto, Portugal
acoelho@fe.up.pt
https://orcid.org/0000-0001-7949-2877

ABSTRACT

Content generation is a heavy task in virtual worlds design. Procedural content generation techniques aim to agile this process by automating the 3D modelling with some degree of parametrisation. The novelty of this work is the procedural generation of the marine alga (*Asparagopsis armata*), taking into consideration the underwater environmental factors. The depth and the occlusion were the two parameters in this study to simulate how the alga growth is influenced by the environment where the alga grows. Starting by building a prototype to explore different L-systems categories to model the alga, the stochastic L-systems with parametric features were selected to generate different alga plasticities. Qualitative methods were used to evaluate the designed grammar and alga's animation results by comparing videos and images of the *Asparagopsis armata* with the computer-generated versions.

Keywords

Procedural Generation, Parametric L-systems, Underwater Environment, Asparagopsis Armata

1 INTRODUCTION

The recreation of virtual worlds is a process that involves multi-disciplinary skills from mathematics to design, and art. 3D asset generation is a heavy processing task, and it takes specialised knowledge from the designer. To agile this gap, procedural content generation techniques were developed.

The novelty of this work is the application of generative grammars to procedural generate a virtual marine alga (Asparagopsis armata) using stochastic and parametric L-systems, influenced by depth and occlusion as environment impact growth factors.

L-systems are a generative grammar used for procedural techniques to model different species of plants. The designed L-system grammar can generate parametric non-deterministic algae that look like *Asparagopsis armata* influenced by external factors of the underwater environment. Depth and occlusion are the two parameters associated with simulating the underwater environment's impact factors on the plant growth.

The grammar was tested by building a prototype based on WebGL to apply the proposed L-system grammar. The application permits to control the axiom and production rules of the grammar and to parametrize the underwater environment. For example, the user can define the number of algae to be spawn, the space between them and the spots where the algae are sown.

A "wave "animation based on shaders was added to the algae, as well as a terrain with rocks and an animated water surface were used to enhance the underwater environment sensation.

The rest of the paper is structured as follows. Section 2 presents the literature review related to procedural content generation, L-systems and its taxonomy. The review process explores how L-systems can generate plants and vegetation content look-alike and finishes by presenting the alga's growth dependency on the context of seeded and particularities of *Asparagopsis armata*. Section 3 describes how the experimental design was set up, how different types of L-system were applied. Section 4 details how depth and occlusion penalty factors were applied to the plant growth. Section 5 presents the results, the visual comparison between the computer-generated and real-world algae, and section 6 discusses the results. The paper ends by presenting the conclusions and future work.

2 LITERATURE REVIEW

Content creation is one of the main costs of developing a video game, it is estimated to be around 30%-40% of the US 20$M-$150M average budget for AAA games [Bar19]. To agile the content creation process, procedural content generation is an alternative to manual design of the game assets, providing techniques to automate the content generation [Hen13]. Procedural Content Generation (PCG) is the algorithm creation of the game content with limited or indirect user input [Tog11a].

The term content is related to the assets in a game: levels, maps, game rules, textures, stories, buildings, music, etc [Tog11b]. The procedural content can be characterized by grouping in what kind of content is generated, how the content is represented, and how the quality/fitness of the content is evaluated [Noo16]:

- Online-Offline Generation: the content is generated in real-time or before the game start.
- Necessary-Optional Content: if it is necessary or optional for the particular game.
- Control Degrees: type of generation algorithm that is used and how it can be parameterised.
- Deterministic or Stochastic Generation: the degree of randomness in the build process.
- Constructive or Generative-with-test: output of the algorithm. Constructive algorithms generate the content and end execution, producing the output result.

The methods and techniques used to generate the content can categorise the procedural content generation (PCG) in the following groups [Hen13]:

- Pseudo-Random Generation (PRNG): e.g., algorithms that produce random numbers based on mathematical formulas can be used to generate textures.
- Generative grammars (GG), e.g., Lindermayer-system, split grammars, wall grammars, shape grammars.
- Image Filtering (IF), e.g., Binary Morphology, Convolution Filters.
- Spatial Algorithms (SA), e.g., Tiling and layering, Grid subdivision, Fractals, Voronoi Diagrams
- Modelling and Simulation of Complex Systems (CS), e.g., Cellular Automata, Tensor fields, Agent-based simulation
- Artificial Intelligence (AI), e.g., Genetic Algorithms, Artificial Neural Networks, Constraint Satisfaction and planning

Networks, Constraint Satisfaction and planning.

Aristid Lindenmayer [Lin68] created Lindenmayer systems (L-systems) to model multi-cellular organisms, but their versatility proved suitable for modelling plants [Fit18] with three-like structures [Cio09]. L-systems are a formal grammar that produces strings that get rewritten over time, in parallel. The plant structure is decomposed into modules or components corresponding to the plant's physical units, e.g. a leaf. Each component is represented by a character. The axiom, composed of a specific string of components, defines the initial state of the plant. The production rules are composed of strings with components and information about the rotations to be applied to x, y and z axis [Bou12]. The plant life cycle can be reproduced by applying the production rules to the initial axiom through successive accumulative repetitions. As an example, the following three components can formally define an L-system {G, R, a}, where: a) G : set of finite symbols that represent the plant components; b) R : production rules, specifying the transitions; c) a : axiom representing the initial state of the plant; Then apply an L-system grammar defined by:

a:	a_r
R1:	$a_r \rightarrow a_l b_r$
R2:	$a_l \rightarrow b_l a_r$
R3:	$b_r \rightarrow a_r$
R4:	$b_l \rightarrow a_l$
G:	a_r, a_l, b_r,b_l

Will produce the following sequence:

a_r
$a_l b_r$
$b_l a_r a_r$
$a_l a_l b_r a_l b_r$
$b_l a_r b_l a_r a_r b_l a_r a_r$
(...)

The L-system taxonomy can be grouped into the following types [Pru96]:

- DOL-systems (Deterministic and context-free): These are the simplest L-system. They are deterministic: the rules applied to the axiom will always generate the same grammar. As a consequence, all produced content is the same.
- Bracketed L-systems (Improve branching by saving state): Group a set of components inside a closed pair of the brackets. Restore the initial state after applying the rules inside the brackets.
- Stochastic L-systems (Define rules to a symbol based on a probability): The non-deterministic

technique, named stochastic L-system, was used to generate different plants from the same axiom and rules. Each plant component has more than one rule associated. Then, for each iteration is randomly chose a different rule associated with a component.

- Context-sensitive L-systems (Depends on neighbour's symbols context): Captures the interaction between adjacent elements of the developing structure. The activation of the production rules is based not only on substituted symbols but also on their neighbours.
- Differential L-systems: Enables interactions between plant components during development. Combination of discrete-continuous models of development, modules are created discretely, obeying the production rules but developed continuously described by differential equations.
- Parametric L-systems (Add numerical parameters, e.g., line length): The previous techniques produce plants with the same angles and components length. The visual result of the plant is complete symmetrically. Parametric L-system overcomes this limitation by configuring the length of the different components and the values of the rotation angles.

Turtle geometry [Gol04] can be used to draw the L-systems. The plant components are drawn one by one, moving forward in a defined heading. As a metaphor, each component is drawn on paper without lifting the pen. At each iteration step, the turtle saves a state composed of a position and heading. For L-systems that support branching when the production rules parser find a '[', it will push the turtle state to a stack, and when it finds ']', the turtle state will be popped, and the turtle states switch to the state before the '['.

The virtual generation of virtual crops has into consideration the environment's influence on the growth and the plasticity of the plant [Mar17] essential to generate reliable simulations to study the evolution of the crops. Soil fertility and space distribution are examples of influencing factors [Tal20] that can be used to define a fit function for modelling plant growth. Small mutations were added to the L-system grammar generation process to augment the plant diversity and validate the more suitable plants to survive in the simulated environment [Bor09]. This work will focus on the procedural generation of underwater environments. Diverse impact factors can influence the sea-life distribution, such as space competition, wave energy and sunlight attenuation [Li14].

The algae intended to be procedurally generated in this work lives in a marine ecosystem. These ecosystems are composed of multiple actors that can be procedural generated: terrain, biofouling, vegetation, water (caustics), and sea life. The red alga *Asparagopsis armata* was chosen to perform the procedural generation of marine algae. The *Asparagopsis armata* was first described in 1885 by Harvey and is an invasive alga from Australia [And04] present in the North Atlantic ocean. The alga commonly grows at depths from zero to ten meters and has high potential growth in environments with more light intensity [Mon05].

3 METHODS AND MATERIALS

Identifying the main alga components and the influence growth factors were the initial steps to set up the experiment. Then, an L-system grammar was designed to model the alga and impact factors on alga growth were defined. The experimental setup is based on a web application where a user can write axioms and production rules to generate a 3D visualisation of the alga.

A WebGL based library was used to build the prototype. In addition, different categories of L-system grammars are supported by the application, and marine environment parameterisations are available to the users through a minimal user interface.

3.1 Algae Components

To correctly model the alga, an initial study about alga *Asparagopsis armata* morphology was performed. Three main components were identified to model the alga plasticity: branches, stolons, and filaments. The filaments are a thin branch that looks like cotton, and stolons look like a hook/harpoon, and the branches constitute alga structure support (Figure 1 [1]).

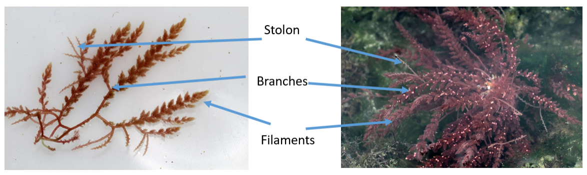

Figure 1: Algae plasticity.

Each component is computer-generated using basic geometric solids. A cylinder with a sphere on top is a branch. Filaments are represented by a cone. The stolon is composed of a cylinder as a base, a sphere and a cone (Figure 2).

3.2 Algae Grammar

A generative grammar was defined to model the alga, (Table 1). The branch is represented by the character 'B', the stolon by the character 'S', and filaments by the character 'F'. The '+' and '-' were used to rotate on the z-axis, the '\' and '/' were used to rotate on the x-axis, and the characters '<' and '>' to rotate on the y-axis. The characters '[' and ']' were used to create a new state and return to the previous form.

[1] http://www.omare.pt/pt/galeria-especies/687

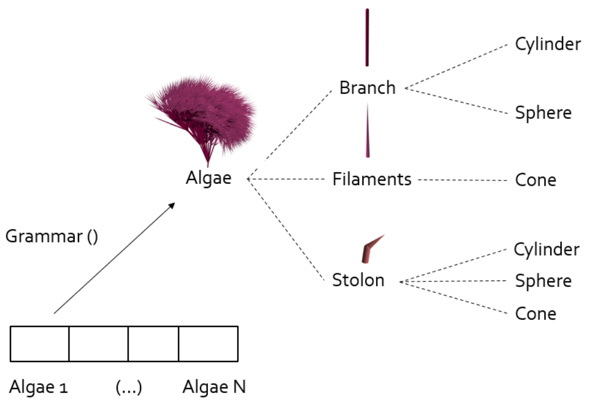

Figure 2: Alga's components.

Character	Action
Component	
B	Create a Branch component
S	Create a Stolon component
F	Create a Filament component
Axis rotation	
+	Rotate positive in the z-axis
-	Rotate negative in the z-axis
\	Rotate positive in the x-axis
/	Rotate negative in the x-axis
<	Rotate positive in the y-axis
>	Rotate negative in the y-axis
State	
[	Push (create a new state)
]	Pop (return to the previous state)

Table 1: L-system Grammar

The advance on the drawing process of the alga is performed by using the turtle graphics technique.

Different categories of L-systems were explored, starting by Deterministic Context-Free (DOL) with bracket support to branch preservation state, non-deterministic stochastic and parametric L-systems.

3.2.1 *Deterministic Context-Free (DOL) Systems*

With this category of L-system the rules applied to the axiom will always generate the same grammar. As a consequence, all algae look precisely the same. Starting by a simple axiom, a 'Branch 'will create a 'Stolon 'with a negative rotation of 30Â° degrees on the z-axis and a filament 'Stolon 'with a negative rotation of 30Â° degrees on the z-axis (Table 2).

Axiom	B
Rule for Branch	B-[S]-[F]
Rule for Stolons	S
Rule for Filaments	BF

Table 2: DOL-system grammar

The following character sequence is generated by iterating three times over the defined grammar:

Iteration 1 → B-[S]-[F]
Iteration 2 → B-[S]-[F]-[S]-[BF]
Iteration 3 → B-[S]-[F]-[S]-[BF]-[S]-[B-[S]-[F]BF]

The alga evolution dictated by the generated grammar is illustrated in Figure 3. Each iteration is added a new Branch, Stolon, and Filaments with a negative rotation in the z-axis.

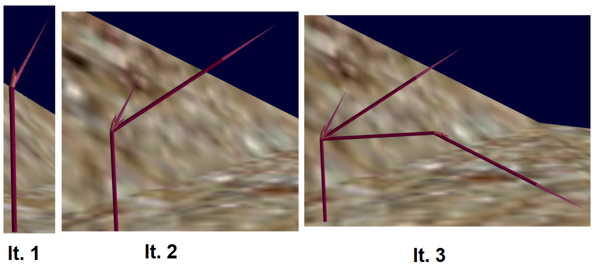

Figure 3: Iterating over the DOL-system grammar.

3.2.2 *Stochastic L-systems*

In nature, each alga is unique. To obtain similar results, non-deterministic grammar with stochastic behaviour was added to the prototype. Each component has more than one production rule, and each iteration is chosen randomly (Table 3). As a result, the same initial axiom and production rules will produce different characters chains and consequently different algae.

A weight factor can be attributed to each rule. This parametrization enables the creation of models of a stochastic alga in a specific direction or form. Figure 4, illustrates that, by randomly assigning different rules to the Filaments component made it possible to obtain three different algae at the end (Figure 5). The process is repeated for each component in iteration.

3.2.3 *Parametric L-System*

The previous techniques produce algae with the same angles and same components length. The final visual result of the algae is completely symmetrical. To configure the length of the different components and the values of the rotation angles two new parameters were added to the grammar, (Table 4).

A numeric value is assigned to the rotation angle before a character that corresponds to an axis rotation. If the character corresponds to a component, the numeric value will be assigned to the component's scale (Figure 6). If no value is present before the character, a default value is assigned.

Thus, the value of the advance on the turtle geometry is the same as the component scale.

4 ALGAE GROWTH PENALTIES

According to [Mon05] the light intensity influences the alga (*Asparagopsis armata*) growth. In the underwater environment, the light tends to decrease with the increase of depth. Also, the occlusion by neighbour algae

Axiom	B
Rule for Branch 1	B[-S]<[[F]/[F]]+
Rule for Branch 2	B[B]>[B]
Rule for Stolons 1	[S]<[S]>[S]
Rule for Stolons 2	[-S][-S]>[-S]
Rule for Filaments 1	[F][B]
Rule for Filaments 2	[–B[F]-[F]-»[F]-\[F]-/ / [F]-»[F]]
Rule for Filaments 3	[B[F]-[F]-»[F]++\[F]–[F]+-//[F]–[F]]

Table 3: L-Stochastic grammar

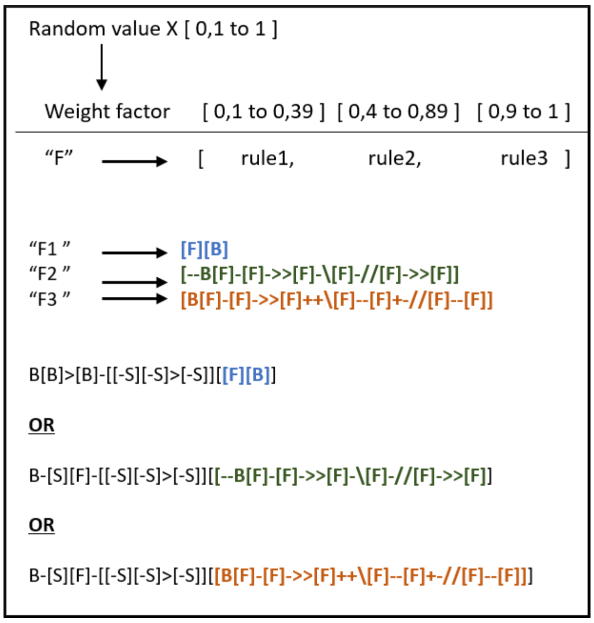
Random value X [0,1 to 1]
Weight factor
[0,1 to 0,39]
[0,4 to 0,89]
[0,9 to 1]
"F"
[rule1, rule2, rule3]
"F1 "
"F2 "
"F3 "
[F][B]
[--B[F]-[F]->>[F]-\[F]-//[F]->>[F]]
[B[F]-[F]->>[F]++\[F]--[F]+-//[F]--[F]]
B[B]>[B]-[[-S][-S]>[-S]][[F][B]]
OR
B-[S][F]-[[-S][-S]>[-S]][[--B[F]-[F]->>[F]-\[F]-//[F]->>[F]]
OR
B-[S][F]-[[-S][-S]>[-S]][[B[F]-[F]->>[F]++\[F]--[F]+-//[F]--[F]]]

Figure 4: Applying L-Stochastic production rules for the Filaments.

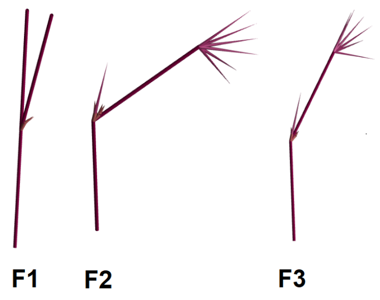

Figure 5: Possible outputs of applying L-stochastic Filaments production rules.

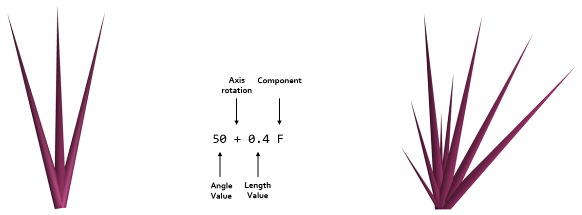

Figure 6: DOL-systems on the left with fixed angles and Filaments length, L-Parametric on the right enables to configure angles and components length.

Value	Action
0.1 ... 0.9	Value before a component character, corresponding to a scale value, e.g [0.9B] will scale the length of the branch to 90% of its unitary value.
1 ... 90	Value before a rotation component, corresponding to the value of the rotation angle, e.g [45+F] will rotate a Filaments component in positive on z-axis 45 degrees.

Table 4: Parametric L-system grammar

impacts the amount of light absorbed by s single alga. These environmental impact factors were chosen to influence the algae growth supported by the developed prototype, as illustrated on the fit function "(1)".

$$seed = \begin{cases} position = x, y, z \\ scale = 1 - (depth\, penalty + occlusion\, penalty) \end{cases} \tag{1}$$

4.1 Depth

In underwater environments, with the increase of depth, the light intensity decrease. The *Asparagopsis armata* produces algae with longer length on spots with higher light intensity. When the alga is seeded, a penalty factor is set to simulate the relation between depth and growth behaviour. With the decrease of y-value, a high level of penalty factor is set. This factor will correspond to a decrease in the scale of algae (Figure 7).

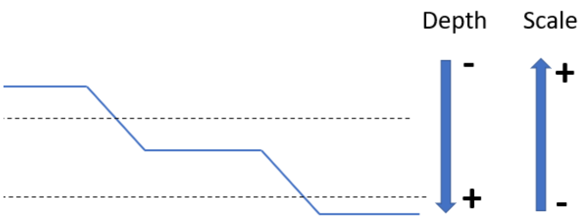

Figure 7: Seabed cross-cut relating depth with alga's scale.

4.2 Occlusion

The first approach to simulate the algae occlusion penalty was based on the z-value of the seed position related to the light spot. The algae that were further away from the light origin is assigned a higher penalty

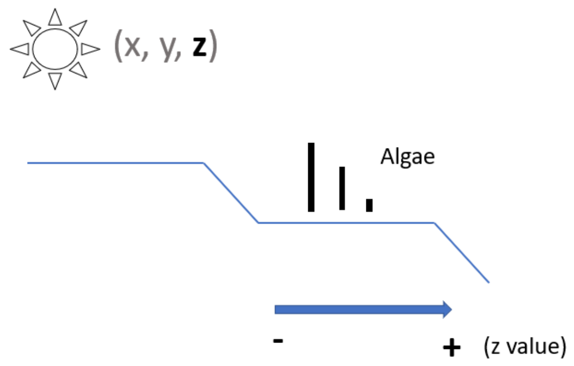

Figure 8: Seabed cross-cut representing algae further away from the fixed spotlight have a smaller scale.

factor. High values of this factor will correspond to a decrease in the alga's scale (Figure 8).

The second approach was based on the number of components produced from alga's grammar. Alga with more components will need more space to evolve and will occlude algae with fewer components. Algae with fewer components will have a high penalty factor that will decrease the scale of the alga (Figure 9).

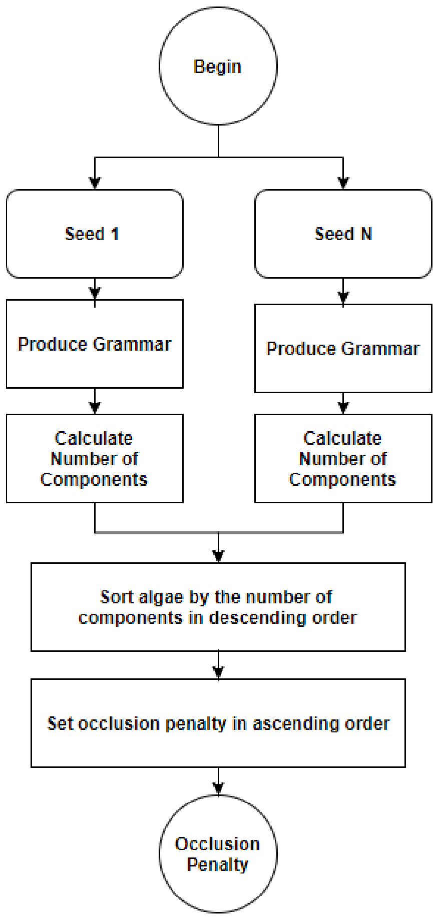
Begin
Seed 1
Seed N
Produce Grammar
Produce Grammar
Calculate Number of Components
Calculate Number of Components
Sort algae by the number of components in descending order
Set occlusion penalty in ascending order
Occlusion Penalty

Figure 9: Definition of the occlusion penalty based on the number of components.

5 ALGA ANIMATION

To animate the alga two different techniques were used. The first approach uses the CPU to animate the alga based on translations and rotations by changing only the x and y-coordinate values in the world coordinate system. When the alga is instantiated, a slightly random value is assigned to the z-axis (upwards direction) to give the sensation of individual wave movement to the alga within the group. The loop of the movement was based on a sinusoidal function (Figure 10).

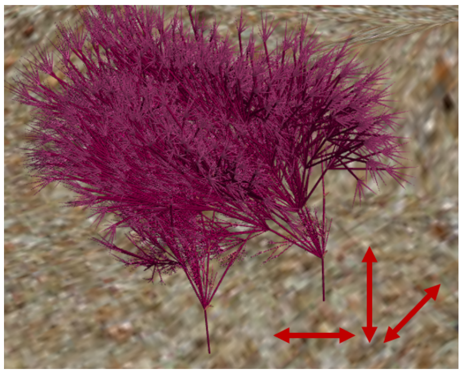

Figure 10: Alga animation based on translations.

The second approach of the alga animation was performed using shaders (GPU) that permit to obtain a dedicated animation for each alga. The vertice transformations are performed on the vertex shader. The x-coordinate is displaced, having as reference the y-coordinate value in the object coordinate system. The z-coordinate value is calculated based on a sinusoidal function to simulate the wave loop movement (Figure 11). The current 3D representation of the alga does not have a component hierarchy relation. The vertices with high values will decrease the offset value to avoid the algae components getting apart.

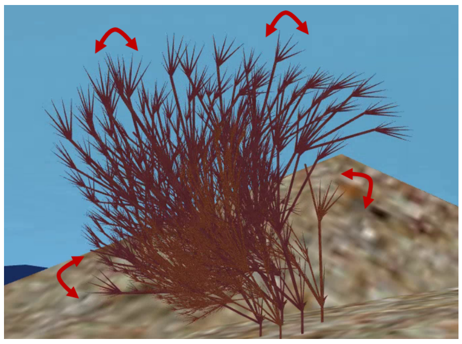

Figure 11: Individual alga animation using shaders.

6 RESULTS

6.1 L-system grammars

It was possible to explore different types of L-systems, starting from the most basic deterministic DOL-System to parametric L-systems, adding a non-deterministic behaviour (Figure 12).

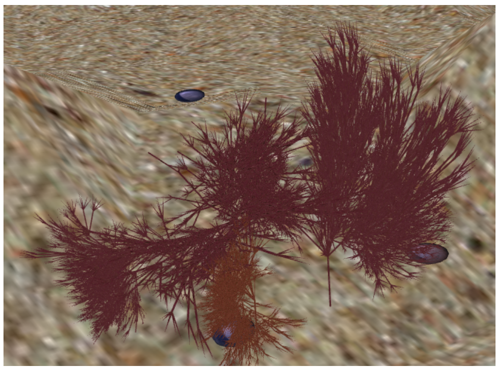

Figure 12: Three algae using parametric L-system with stochastic behaviour.

6.2 Depth penalty factor

The depth factor penalty produces the effect illustrated in Figure 13: algae instantiated in positions with lower y-values, are smaller than their neighbours with higher values on y-coordinates.

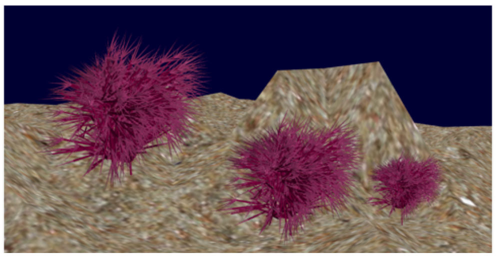

Figure 13: Visual representation of depth penalty.

6.3 Occlusion penalty factor

The first approach, using the z-coordinate as a reference relative to the light source, draws smaller algae comparing with algae spawned with high z-coordinate values (Figure 14).

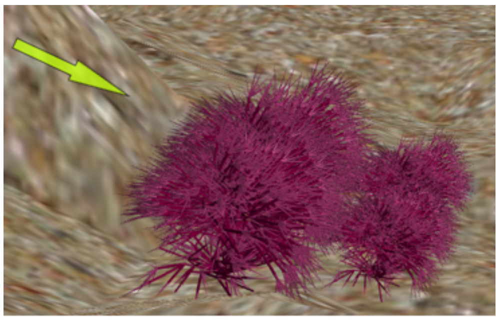

Figure 14: Occlusion using the alga distance to the spotlight. The green arrow illustrated the light direction from the light source

The second approach used to apply the occlusion penalty was based on the number of components generated by the initial grammar when sown. Algae with more number of components will have a minor penalty in their scale and, as a consequence, will produce larger algae (Figure 15).

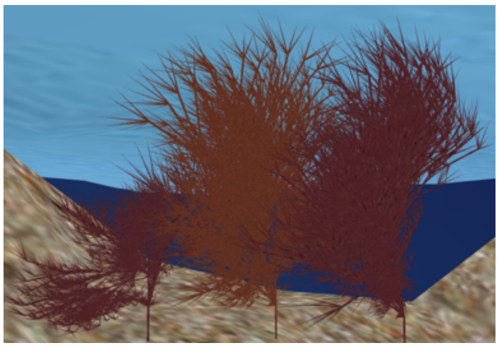

Figure 15: Occlusion is based on the number of components generated by the grammar.

6.4 Grammar render time

The grammar's performance was measured by logging the time it takes to display the L-system grammar during 1000 render loops. Table 5 presents the average time in milliseconds for the render loop session. The first trials were defined a DOL grammar and iterated 3 and 5 times through it. The same procedure was followed for the L-system parametric grammar.

Scene	Time (ms)
1 alga 3 iterations (DOL)	2.3
1 alga 3 iterations (Parametric)	100.7
1 alga 5 iterations (DOL)	39.8
1 alga 5 iterations (Parametric)	269.2
3 algae 3 iterations (DOL)	39.8
3 algae 3 iterations (Parametric)	179.5
3 algae 5 iterations (DOL)	54.1
3 algae 5 iterations (Parametric)	801.4

Table 5: Average time to render the L-system grammar.

The tests were performed in a machine with the following specifications: Processor: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz; RAM: 16 GB; Graphics card: NVIDIA GeForce RTX 2070 Max-Q; .OS: Windows 11.

7 DISCUSSION

The procedural content generation of the alga is validated using the qualitative methods by comparing the “look-and-feel ”of the alga generated by the grammar and penalty growth factors with images and videos of real *Asparagoris Armata*.

The procedural generation algorithm to generate the alga has evolved through the use of different categories of L-systems. The DOL-system grammar does not correctly model the alga. The Branches have all the same angle and length, which is not the default behaviour present in nature. The Stolons have the wrong proportions. With the introduction of parametric features to the L-system grammar, it was possible to model the components with different lengths and set different angles to the rotations to create a more natural look. The

Figure 16, illustrates the evolution of alga procedural generation starting on the left by presenting the output of the DOL-systems and on the right the output of parametric L-systems.

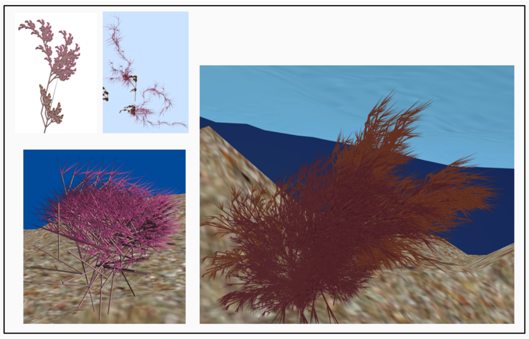
Figure 16: Alga procedural generation time-lapse.

In Figure 17, the alga generated with DOL-systems has the same angle for each rotation and length for all components of the same type. Consequently, this L-system generates algae that look symmetric and "quadratic". On the other side, using the parametric L-system with stochastic behaviour, it is possible to model algae with different rotation angles and component lengths. The final visual result is algae result with plasticities close to what is possible to encounter in nature.

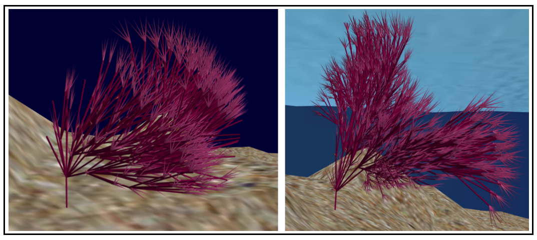
Figure 17: Comparison between the computer-generated DOL-system (left) and L-Parametric (right).

The occlusion penalty based only on the depth value was discarded (Figure 14) because the light is scattered in the underwater environment, and it is not directional from a single source. Having as a reference the distance to the spotlight will not get close to actual behaviour. The best occlusion technique is based on the number of alga components and the space it needs to evolve. The alga with more components will occlude the light to their neighbours that will grow less (Figure 15).

Concerning the alga animation, the algorithm that runs on the CPU is not visually correct because the algae move exclusively in a group and not individually, so the global algae visualisation looks too uniform. However, with the implementation of the animation on the shader (GPU), each alga has its own movement pattern and does not suffer from the uniform translation behaviour.

Using parametric L-systems with stochastic behaviour, plus the alga animation and the recreation of the underwater environment, it is possible to generate non-deterministic content similar to the alga *Asparagopsis armata*. Also, combining generative grammars based on the parametric L-systems, animation, and growth penalty factors makes it possible to model algae similarly to *Asparagopsis armata*. (Figure 18 [2]).

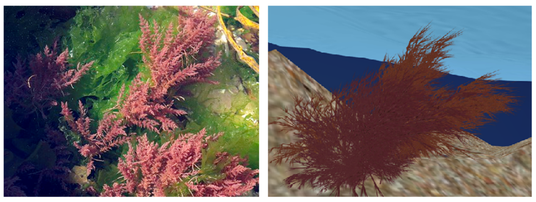
Figure 18: Comparison between alga in the underwater environment (left) and the computer-generated (right).

A comparison video between the algae in the underwater environment and the computer-generated version is available on supplemental material.

The DOL-Systems are faster regarding the grammar's render time to generate the alga(e). The parametric L-system uses a scale factor to resize individual components of the alga, and for each iteration, it is necessary to save the component's state, leading to high values related to the render time to display the alga(e).

8 CONCLUSIONS AND FUTURE WORK

With this work was possible to generate 3D models of the marina algae *Asparagopsis armata* look-alike. The developed prototype made it possible to explore different types of L-systems to model the marine alga *Asparagopsis armata*. Furthermore, the procedural generation process included external environmental factors (depth and occlusion) that influence algae growth. The generative grammar with the best results was the combination of parametric L-system with stochastic behaviour, enabling the generation of unique look for each alga and, at the same time, algae with similar morphologies to those found in nature. The procedural generation of the occlusion impact in the alga's growth, considering the sparsity of the light of the underwater environment, the number of the components of the alga's neighbours was the technique chosen.

As future work, it will be created a relationship or hierarchy between the different alga components. This hierarchy will enable the exploration of animation mechanisms, e,g. Kinematics, and add rules to the grammar to procedural generate them. Also the generation of multiple alga's component for a high number of iterations will be improved. When the complexity of the grammar grows, it is resource-heavy to draw all the primitives. Writing complex L-systems grammars is

[2] https://flic.kr/p/Mc8uLp

prone to errors, and it is challenging to design hierarchical relations between the plant and the components only by typing text into the system. In future iterations, incorporate strategies of inverse procedural modelling [Sta10], and control mechanism based on data-driven procedural techniques to infer the grammar from images or sketch-based inputs to provide intuitive interactions to the designers [Len21].

9 REFERENCES

[And04] Andreakis, N., Procaccini, G. & Wiebe HCF Kooistra Asparagopsis taxiformis and Asparagopsis armata (Bonnemaisoniales, Rhodophyta): genetic and morphological identification of Mediterranean populations. *European Journal Of Phycology*. 39, 273-283 (2004).

[Bar19] Barriga, N. A Short Introduction to Procedural Content Generation Algorithms for Videogames. *International Journal On Artificial Intelligence Tools*. 28, 1930001 (2019).

[Bor09] Bornhofen, S. & Lattaud, C. Competition and evolution in virtual plant communities: a new modeling approach. *Natural Computing*. 8, 349-385 (2009), https://doi.org/10.1007/s11047-008-9089-5

[Bou12] Boudon, F., Pradal, C., Cokelaer, T., Prusinkiewicz, P. & Godin, C. L-Py: An L-System Simulation Framework for Modeling Plant Architecture Development Based on a Dynamic Language. *Frontiers In Plant Science*. 3 pp. 76 (2012).

[Cio09] Ciosek., K. & Kotowski., P. GENERATING 3D PLANTS USING LINDENMAYER SYSTEM. *Proceedings Of The Fourth International Conference On Computer Graphics Theory And Applications - GRAPP, (VISIGRAPP 2009)*. pp. 76-81 (2009).

[Fit18] Fitch, B., Parslow, P. & Lundqvist, K. Evolving Complete L-Systems: Using Genetic Algorithms for the Generation of Realistic Plants. *Artificial Life And Intelligent Agents*. pp. 16-23 (2018).

[Gol04] Goldman, R., Schaefer, S. & Ju, T. Turtle geometry in computer graphics and computer-aided design. *Computer-Aided Design*. 36, 1471-1482 (2004).

[Hen13] Hendrikx, M., Meijer, S., Van Der Velden, J. & Iosup, A. Procedural content generation for games: A survey. *ACM Transactions On Multimedia Computing, Communications And Applications*. 9, 1-22 (2013).

[Len21] Lena Gieseke, Paul Asente, Radomír Měch, Bedrich Benes, and Martin Fuchs. A survey of control mechanisms for creative pattern generation. *Computer Graphics Forum*, 40(2), pp.585-609, 2021

[Li14] Li, R., Ding, X., Yu, J., Gao, T., Zheng, W., Wang, R. & Bao, H. Procedural generation and real-time rendering of a marine ecosystem. *Journal Of Zhejiang University SCIENCE C*. 15, 514-524 (2014).

[Lin68] Lindenmayer, A. Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. *Journal Of Theoretical Biology*. 18, 300-315 (1968).

[Mar17] Marshall-Colon, A., Long, S. P., Allen, D. K., Allen, G., Beard, D. A., Benes, B., von Caemmerer, S., Christensen, A. J., Cox, D. J., Hart, J. C., Hirst, P. M., Kannan, K., Katz, D. S., Lynch, J. P., Millar, A. J., Panneerselvam, B., Price, N. D., Prusinkiewicz, P., Raila, D., Shekar, R. G., Shrivastava, S., Shukla, D., Srinivasan, V., Stitt, M., Turk, M. J., Voit, E. O., Wang, Y., Yin, X., and Zhu, X.-G. Crops In Silico: Generating Virtual Crops Using an Integrative and Multi-scale Modeling Platform *Frontiers in Plant Science*, vol.8, (2017).

[Mon05] Monro, K. & Poore, A. Light quantity and quality induce shade-avoiding plasticity in a marine macroalga. *Journal Of Evolutionary Biology*. 18, 426-435 (2005).

[Noo16] Noor Shaker, Julian Togelius & Mark J. Nelson Procedural Content Generation in Games. (Springer, Cham,2016).

[Pru96] Prusinkiewicz, P. & Lindenmayer, A. The Algorithmic Beauty of Plants. (Springer-Verlag,1996).

[Sta10] Št'ava, Ondrej and Beneš, Bedrich and Měch, Radomir and Aliaga, Daniel G and Krištof, Peter. Inverse procedural modeling by automatic generation of L-systems. *Computer Graphics Forum*, vol. 29, pp.665-674, (2010)

[Tal20] Talle, J. & Kosinka, J. Evolving L-Systems in a Competitive Environment. *Advances In Computer Graphics*. pp. 326-350 (2020).

[Tog11a] Togelius, J., Kastbjerg, E., Schedl, D. & Yannakakis, G. What is Procedural Content Generation? Mario on the Borderline. *Proceedings Of The 2nd International Workshop On Procedural Content Generation In Games*. (2011).

[Tog11b] Togelius, J., Yannakakis, G., Stanley, K. & Browne, C. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions On Computational Intelligence And AI In Games*. 3, 172-186 (2011).