Creating a Realistic Simulation of an Ecosystem using 3D Terrain for use in Games

# Introduction

# Background

# Related Work

Simulation of ecosystems are split into two focuses – realism and performance, with realism focusing on the generation of the environment, and performance looking how it can be optimised and used in practical applications. The generative aspect explores the influences of the generation, such as one of the most important decisions, the structure of the vegetation. This can be broken down into the design of the structure, as well as the factors that influence its growth.

The most primitive type of vegetation simulation is the L-System (Lindenmayer, 1968), this model uses recursive rules to create patterns comparable to the behaviour of plant cells and plant development (Aono and Kunii, 1984), albeit lacking the reactionary generation that plants tend to have with their surroundings. A common purpose of L-Systems is to generate fractal patterns due to the design of the model, production rules are used in the model to recursively add to generated structure causing points of generation to create further points, and to allow for the structure to grow further detail. Structures like this can be seen in life as well as in L-System generations such as the Barnsley Fern (citation needed), Fractal Binary Tree, and even animated fractal trees (Oppenheimer, 1986). Such examples refrain from the incorporation of biological processes as compromise was needed to accelerate computation speeds.

Recent examples of vegetation growth simulation adopt biological processes in their generations. This requires a change of model, as L-Systems are not capable of reactionary processes that utilise data from other parts vegetation as well as data from the environment. The forces that act on vegetation are called tropisms; they influence the plant by manipulating the direction of growth for buds, as well as the direction that leaves face. The model favoured in this biological process is a recursive hierarchical model (Longay et al., 2012) that allows for the base of each branch in the vegetation to have knowledge of everything following it as well as their forces. This model breaks the model down into nodes, buds, and leaves, to create procedurally generated plants (Palubicki et al., 2009) in a reactive model (Mech and Prusinkiewicz, 1996).

The generation of each vegetation in the ecosystem can be broken down into several parts, with the first and most common simulated factor being light, this is often expressed though the use of voxel areas, where leaves and other light-obscuring entities cause shadows to be cast, thus filling their respective voxels in a cone-shaped zone. This method is an effective way of portraying light; however, it is predominantly inaccurate to accelerate computational speed (Bornhofen and Lattaud, 2008). To increase the accuracy of the simulation, an alternative method can be used to detect light-interactions; this method utilises the behaviours of light, using ray casting the simulate the travel of a light ray towards the leaf. At the cost of computation time, this method provides the benefit of allowing for the calculation of light penetration, as well as light scattering. This is important as leaves only absorb roughly 30% of light (Stryer, Gumport and Koeppe, 1995). The effects of light and shade towards a plant’s development affect the individual leaf by causing growth of the surface area by 50%, as well as of the surrounding leaves by causing the quantity of them to decrease by 40% (Stuefer and Huber, 1998). As discussed earlier, light acts as a major tropism towards directing and developing plant growth, by causing the plants to develop towards concentrations of light (Beneš, Andrysco and Štava, 2009).

Plants can also display a negative influence from light, this is called aphototropism, and skototropism when present in vines, causing growth away from direct sunlight. This behaviour is seen predominantly in vines such as the monstera gigantea, as a method of finding large structures to use to climb such as mature trees.

The second major tropism is gravitropism, a unique tropism that cause independent influences on different species and plant structures. In biology, the earliest model to describe this tropism is the Chologny-Went model (citation needed), proposing that plant shoots would be positively influenced by sunlight, and roots would positively affected by gravity, causing them to move towards their relative forces. This tropism is caused by the movement of the hormone ‘auxin’ within the plant cells, this dense substance directs is used to decrease the growth of root cells on the side that they have accumulated on, directing the root downwards, and this has the opposite effect on plant shoots, directing them away from the force of gravity.

As a plant matures, gravitropism is still present in influencing its growth alongside phototropism, however, plant organs such as leaves contain no sense of gravity, with their only guiding sense being phototropic responses. As a plant fully matures, the only major tropism affecting them is phototropism, as they eventually lose all gravitropic influence.

Gravitropism is not a uniform influence among species either, with some species of vegetation growing with the influence of gravity, such as salix babylonica or weeping willow, which reach adulthood then show the pendulate growth habit, instead of the expected response. Another example of this is in plants that cannot sustain their structure, and therefore adopt growing along the ground, this is present in species of tomatoes, corn, and rice; this behaviour is known as diagravitropic. This is caused by the substance to detect gravity becoming less dense and therefore less sensitive to gravity, causing the plant to grow atypically to expected gravitropic behaviour also known as agravitropic.

There are also tropisms completely unique to the roots, the use of hydrotropism and chemotropism are the stimulation of growth towards areas of high water and nutrient concentration. This helps act as a determining factor for the procedural growth of roots (Mech and Prusinkiewicz, 1996). Whilst these act on development on a micro scale, a defining factor of ecosystem design is temperature (Makowski et al., 2019) as every plant has an ideal range of temperature and rainfall, with either of these being able to cause the plant to perish (Amissah et al., 2014). Another major factor of the ecosystem is wind, being a unique force forcefully driving the direction of the plant and offsetting the direction of growth. At high wind speeds it can also prohibit growth, causing the stoma in the leaves to be damaged from impacting against each other, making the leaves less efficient at precipitation and thus requiring 88% higher water intake as well as a decreased growth rate of 20% at wind speeds of 3.5ms-1 (Rees and Grace, 1980).

## Acceleration techniques

Acceleration techniques are a common method to decrease computation time, and accelerate a simulation, this allows for tests to be done more frequently, speeding up projects. A common acceleration technique used in simulations is the use of multithreading. Multithreading utilises CPU cores to compute multiple tasks asynchronously, this allows for repetitive tasks that have little to no influence on each other to be computed independently on individual threads.

Multithreading can be implemented in multiple methods; an efficient method of multithreading is through the use of a thread pool. A thread pool is a collection of threads that run constantly, these threads are then assigned jobs, if there are more jobs than threads available then more threads are made. An alternative method is the use of a job queue, this is similar to a thread pool however it uses a fixed number of threads that complete tasks in a queue until all of the tasks are completed, where they wait until more tasks are added.

Another useful acceleration technique is the use of a BVH (bounding volume hierarchy), this is an optimisation focused on decreasing computation time for ray cast intersection detection, this is a popular model used in raytracing due to the high number of rays that need to be tested for intersection. Methods of constructing a BVH involve splitting objects into individual axis-aligned bounding boxes that encapsulate their items, these bounding volumes are then split repeatedly until it is impossible to split the list further. Due to the computation speed of rays and axis-aligned bounded boxes, testing for intersections and compiling a list of potential intersections for more complex intersection calculations.

There are multiple ways to implement a BVH, each with their own benefits of accuracy, computation speed, and model construction. The first model involves splitting the collection repeatedly into multiple collections of objects, of which their total bounding volumes have the least surface area possible, this is done by comparing every possible collection, this is done until every collection consists of a single item. The second model is similar to the first however it pairs collections together in groups that have the least possible surface area, although this makes the creation of the BVH more complex, it has been seen to provide more efficient BVH models (citation needed). The final model is similar to the first model, however it is constructed by inserting objects individually into the BVH, with the entire model to be recalculated periodically to correct inefficient structures, this model has the benefit of being able to be constructed dynamically, with objects introduced to the hierarchy successively.

* Multithreading
  + Job queues
  + Thread pools
  + BVH

# Methodology

## Model

* Generative model
  + Proleptic and Sylleptic
  + Recursive vs L-system

Plant growth will be based on the model introduced by Palubicki et al. (2009) using the proleptic and sylleptic behaviours shown in (figure) displaying the nature of branching nature. Where terminal buds have a chance of becoming proleptic or sylleptic based on parameters dependant on the plant. This can also be expanded on, with lighting promoting different behaviours, such that vegetation in shade prioritise growth upwards to reach the canopy before displaying proleptic behaviours.

## Tropisms

Tropisms drive the uniqueness of the vegetation entity, with this defined by the location of each point in the structure of the vegetation, as well as the location of all other entities and structures. The tropism behaviours that will be implemented will be phototropism, gravitropism, and a spatial tropism, to estimate the influence of indirect light as well as Thigmotropism. Other tropisms would also be possible to add to this model, as the implementation of superimposed directional vectors allows for as many tropisms as wanted.

The implementation of light will use ray casts as a method to recreate light. Although shadow cones and spatial hashing is a common method used by several implementations to calculate shade in individual locations, this implementation will utilise ray casts in a method similar to ray tracers, with several light rays being sent towards a light source to check for intersections with other vegetation, this then allows for the expansion of the shadow cones, in having the ability to calculate light penetration. The disadvantage of using ray casts is the computation time, this can be offset through the use of a BVH (bounding volume hierarchy) that uses the bounds of each structure in the calculation to accelerate the detection of interactable objects. Light penetration will be calculated using a falloff curve, such that each leaf that interacts with the light ray absorbs 30% of light (Stryer, Gumport and Koeppe, 1995), with this reducing the remaining light available.

The gravitropism with be implemented by calculating the force acting on the branch, this will be dependent on the branch mass, increasing the gravitational force acting on the growth. There will also be a constant force defined by the species of the vegetation, this will act as the gravitropic behaviour.

The final tropism implemented is a spatial tropisms, promoting the plant to grow away from others, this is done simulate the influence of indirect light, which would cause phototropism to stimulate growth divergently from other objects, or the opposite in the case of vegetation that uses other objects as structure support such as ivy.

## Structure

Another component the vegetation has is the choice to grow in a sylleptic or proleptic behaviour, with sylleptic growth splitting the direction of the terminal bud, causing an auxiliary bud to grow beside it in the opposite direction. Proleptic growth does the opposite to this, with it maintaining the direction of the terminal bud, however it allows for an auxiliary bud to grow to the side of the terminal branch. This ‘sylleptic chance’ is one of the fundamental defining features of a species, with vegetation such as bushes often preferring to grow in a sylleptic manner, and trees that often prefer proleptic growth to reach as high as possible through the canopy (citation needed).

Sylleptic chance is generally very small, with this occurring in events of abundant resources such as water, nutrients, and light. In fruit trees this behaviour is present, with minimum conditions required to be met for the potential of sylleptic growth (citation needed).

## Acceleration Techniques

To decrease the computation time of calculating the light values and tropisms of the individual vegetation features, this can be multithreaded to optimise and decrease this task. The two multithreading options are the use of a job queue, and the use of a thread pool, a thread pool would be inefficient in this example, as updating the light values would only occur in large batches, this means the threads in the pool would only be used once before waiting to be used again. Therefore a job queue is a more optimal method, as the prespecified quantity of threads taken from the list will be able to take from the job queue until there are no more tasks left, meaning the threads will be able to wait until this happens.

To further accelerate light computation times, a BVH could be implemented, this can further optimise the most complex computational section. However, due to the calculation time of BVHs having O(N2) complexity, this makes it inefficient to recalculate this repeatedly. This inefficiency can be averted based on the structuring of the BVH model, with there being 3 different methods of construction. The first method is a top-down method; however, this requires partitioning the full list of objects. The second method is bottom-up, pairing up objects and groups, although this has the same flaw as the first method. The third method is partitioning by insertion, this method provides the most benefit, as the BVH can be added to over time and does not require the reconstruction of the entire BVH for every new object or vegetation node.

# Method

The method of the implementation for the project was carried out such that there would be a focus on the generative structure with visual aspects as an extension of the primary project – to create a realistic vegetation structure. DirectX was used as a visualisation tool to help with understanding the generated vegetation. This, alongside other fundamental elements of the project, allowed for the foundation of the project.

## Transform

The first foundational aspect of the project was the transform space required to be created in order to create recursive elements to the vegetation structure. The transform space can be split into local/object and global space, to transition between spaces the model matrix would be calculated and multiplied in the following method:

Where is the global space matrix, is the parent transform space, and is the child transform space. The local space matrices are affine transformations, constructed from translation, quaternion rotation, rotation origin, and scale, this is used for vertex position calculations in the model, view, and projection matrix. A similar procedure is taken for rotation, with the inverse possible, so that a global rotation can be converted to a local rotation using the inverse of the parent matrix in the following equation:

This allows for local and global rotations to be set, this is especially useful in the application of tropisms, bud rotations, and geometry creation.

## Vegetation Structure

* Vegetation Structure
  + Defining factors
  + Nodes
    - Branches
    - Buds
    - Leaves
  + Buds
    - Fate
    - Growth Cost
  + Leaves
    - Light absorption
    - Shedding

The vegetation structure consists of the vegetation entity, the structural nodes, and the features. The vegetation entity controls light calculation pass and the update features pass, with this updating each node with the amount of growth directed towards it. The structural nodes use this growth unit to distribute it amongst the next nodes in the hierarchy, with this being calculated in the same way as the vegetation entity does, by weighting them on contribution to the overall growth gained. The final component of the structure are vegetation features, which take the role of the bud and the leaf.

Each bud’s fate is determined each cycle, with the fate of the bud determined by the light it has received in the current cycle. If the light reaches a minimum amount the bud becomes dormant and has the potential of being shed. After updating each connected node, the node checks through its connected and if all of these branches are dormant, they are shed from the main structure. This process is known as Cladoptosis, a self-defense mechanism to remove branches that are diseased or in the shade to prevent a waste of resources.

The buds store growth until their growth cost is met which is based on the volume of what is needed to be created, after this new node is created succeeding the bud, the next node position is extended from the original bud based on the size of the branch it is growing from.

## Tropisms

* Tropisms
  + Light (Phototropism)
  + Gravity (Gravitropism)
  + Spatial tropism

Tropisms have the influence of affecting growth direction, the primary tropism is phototropism, causing the vegetation to grow towards the direction of the source of light, this aids vegetation to grow around obstacles that block direct sunlight and allows them to thrive. The use of raycasts allowed for light to be checked from multiple directions to simulate the path of the sun, with each of these sun rays checked if they intersect with any of the vegetation. The use of raycasts allowed for the sun rays to simulate leaf penetration, as leaves only absorb a portion of the light, as well as a specific range of the wavelength. To accelerate the collision line-sphere intersection detection was used, such that the following equation can be used.

This can detect any intersections where is the centre of the sphere, is the directional vector of the ray, is the radius of the sphere, and is the origin of the ray. This allows the calculation of the amount of light that reaches the leaf, as well as the direction it predominantly comes from. Through using the average of all the light rays that reach the leaf, it creates the phototropic vector.

Gravity is the easiest tropism to estimate, as the influence it has on the individual bud depends on the gravitational moment that is calculated through using the distance the bud has from the centre of gravity, as well as the mass of the branch that causes it to be pulled downwards. The centre of gravity can be calculated using the width of each node, the length, and getting a weighted average of positions, this uses the following equation.

Alongside the following equation to determine the moment.

Where is the moment, is the force perpendicular to the branch, is the distance to the pivot, is gravity, and is the angle the branch is to the ground. This allows for the calculation of the gravitational tropism vector.

The final tropism calculated encapsulates the estimation of the effects of indirect sunlight, as well as Thigmotropism, the reaction to touch. As vegetation reacts to touch differently this can change between different species, with some vegetation growing towards other objects as an act to gain structural integrity, and others moving away to prevent crowding or entanglement.

This is calculated by sampling and comparing the positions of each vegetation feature, this uses an inverse-square falloff to represent the falloff of the indirect light, this uses the following equation to determine the spatial tropism factor:

Where calculates the distance the vegetation feature is from current vegetation feature.

These tropisms all contribute to the final calculated tropism vector by using the sum of all the tropism vectors, this vector can be used to create a look-at rotation matrices of which the growth direction can use to spherical linear interpolate towards using the following equation.

Where is the current growth rotation, is the tropism rotation, and is the interpolation point between the two rotations.

## Species Factors

Throughout the simulation multiple constant factors are used to define how a species of vegetation behaves, these factors define how they respond to tropisms, how frequently they branch, how they distribute growth, and their lifespan.

The response to tropisms is defined by 3 individual factors, as each species can respond to individual tropisms uniquely. Each of these factors acts as a scalar to their relevant tropism, this is applied to the tropism vectors to achieve the overall tropism direction.

The sylleptic chance defines the frequency the vegetation branches into two terminal bud directions instead of one. This scales directly with the abundance of resources allocated to the branch, with branches having a higher chance of growing sylleptically if the branch is more useful. Growth bias is also used to promote growth in terminal branches over branches created proleptically, this allows for growth to be prioritised on the main branch, aiding in growth upwards towards the canopy.

## Interactions

* Interactions
  + Light
    - Raycasts
  + Extension range
  + Shedding

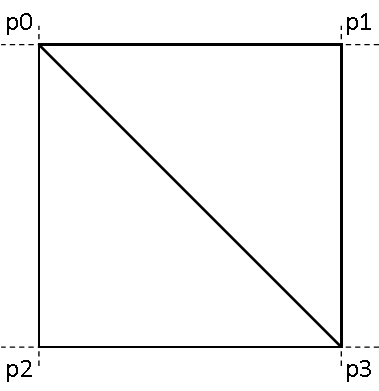
Using these interactions, circumstantial generations are made to simulate realistic growth. To simulate the actions taken by vegetation throughout its life to maintain itself structurally, the plant sheds branches that are deemed unnecessary or disadvantageous to its growth. This situation arises when a branch is in the shade of another or has grown into a direction of little value to the plant. To simulate this a shedding threshold is decided, and if this threshold is met then the branch is removed. This threshold also increases based on the lifespan of the plant, to simulate the gradual overall senescence of the plant as it deteriorates with age. This lifespan-based falloff occurs after the plant has reach its lifespan, after a constant step is used to increase the threshold required to keep the branch until all branches are shed and the entity has fully decayed.

## Visual Generation

* Visual Generation
  + Texturing
  + Cylinder trunks
  + Alpha testing leaves

The visual generation of the vegetation starts with compiling the positions and connections of each of the vegetation nodes, this creates a recursive chain of points, each containing their next node in the chain. For each of these segments a circle of points is created with a radius ‘’ of the node size using the following equation:

is also used to define the detail of the circle. Additionally, a transform the same as the vegetation nodes’ global rotation and translation is applied to these points, by moving the points into global space this allows for points of the previous nodes to be used easily by the next in the chain to construct a cylinder to represent the stalk of the vegetation in the following structure, wrapping around the circle to create a cylinder connecting the current node and the previous node together.



The next stage of generation is the creation of leaves, to optimise the generation time, this is done in clusters per vegetation feature, the falloff of leaves per cluster is linearly proportional to the width of the branch. This is also used in the calculation of light absorption of the vegetation feature. The visual creation of the leaves uses a double-sided quad (4 vertex points in a flat square) by creating two triangles (p0p1p3 and p0p3p2), these are then given a random rotation, as well as an offset translation from the original. This quad is then textured with a scattering of leaves to create the illusion of a volume of leaves.

Texturing Is accomplished by using alpha testing as the leaf textures used mostly contain fully transparent pixels and without any semi-transparent pixels that would require alpha-blending. Alpha testing renders transparent objects from the furthest away to the closest, each time a pixel is drawn to the screen it checks the texture of the object, if the colour is transparent or below a threshold it is discarded, allowing for the pixel below it to be rendered, this is repeated until it has rendered the object closest to the camera.

## Computation Acceleration

* Computation Acceleration
  + Threading
  + Job queues

Computation acceleration is an optimisation to decrease the time for a program to compute a section of code, a common way to do this is through multithreading the code. One method of multithreading is via a job queue, this creates several autonomous threads that run independently and take available jobs off a list until the list is empty. The easiest thing to multithread in vegetation generation is the calculation of tropism factors, as this is updated every cycle and it is carried out by every vegetation feature in the simulation.

The job queue was implemented by running several threads asynchronously in an event loop, during this loop they wait until they are required to complete a job or terminate. The jobs consist of updating a single vegetation feature with its tropisms, during the update of tropisms the threads have access to a list of all vegetation features, this allows for the vegetation features to calculate all the tropisms necessary to them.

Results

Discussion

Conclusion

Project Diary