Bali 2005, *Vue 5 Infinite* demo. image

# *Generating Realistic Foliage*

**FIT3094** AI, A-Life and Virtual Environments

Alan Dorin

MONASH University
Information Technology

## Learning Objectives

To recognise the potential applications of foliage in virtual enviornments.

To understand the benefits and drawbacks to different virtual plant generating techniques.

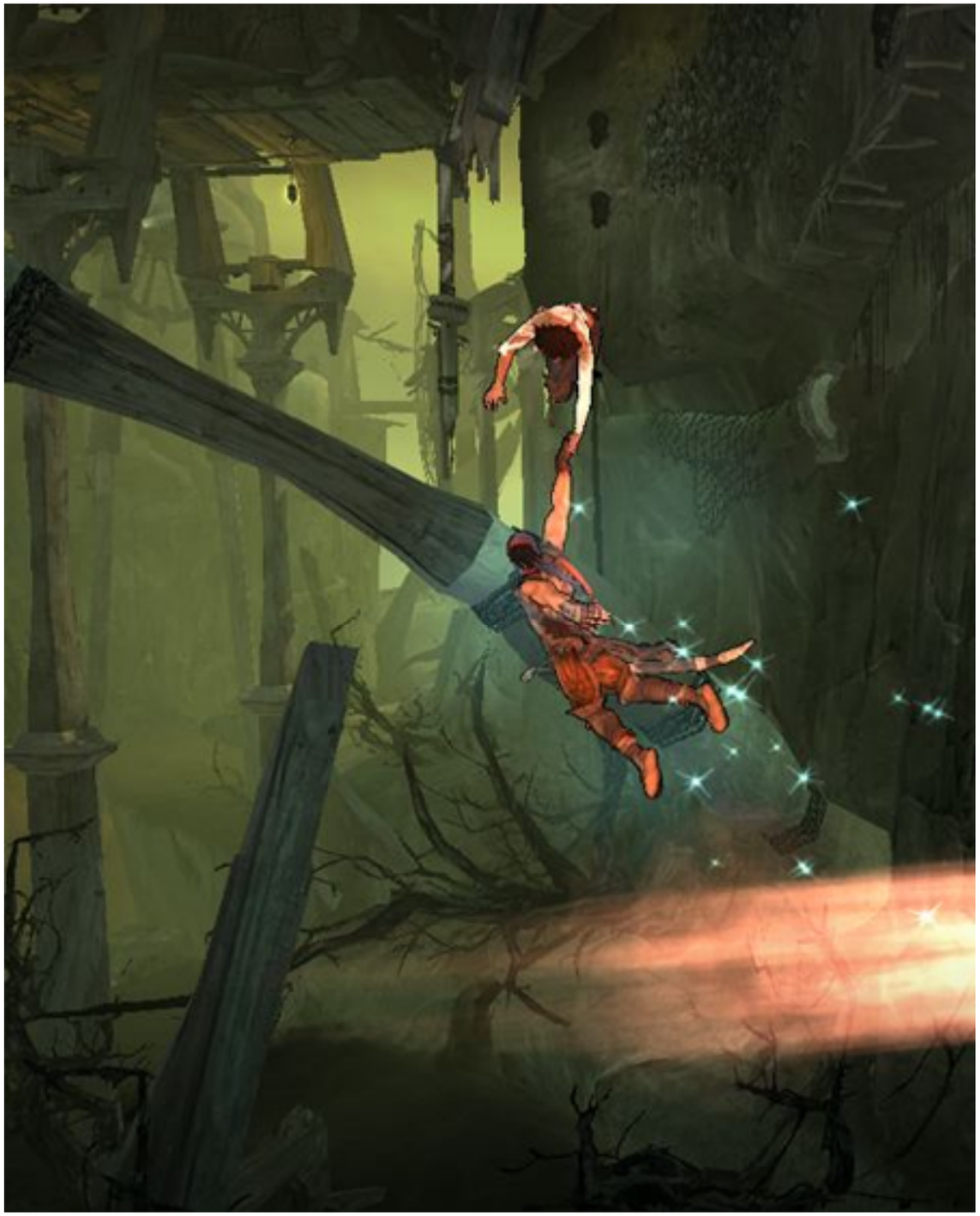
To understand *Lindenmayer Systems* for plant model generation.

To understand how billboarding allows for the realtime rendering of complex forest geometry.

To understand how to generate virtual hills and mountains and populate them with virtual plants.


*The Sims 3*, Electronic Arts Inc. 2009

# Foliage, Game and Virtual Environment Design



*Prince of Persia*, Ubisoft 2008

Plants provide mood.


*Halo Wars*, Ensemble Studios 2009

Architectural model, Kordela Studio
www.kordelastudio.com

Plants also act as visual queues about the climate.

Openc2e Creatures engine
openc2e.org/screenshots

Architectural model, Kordela Studio
www.kordelastudio.com

What is the climate here?

Plants provide cover from bullets (and other projectiles) in shooter games.

*Jungle Boogie*, Ando, 2007

Plants provide decoration and obstacles for racing and sports games.

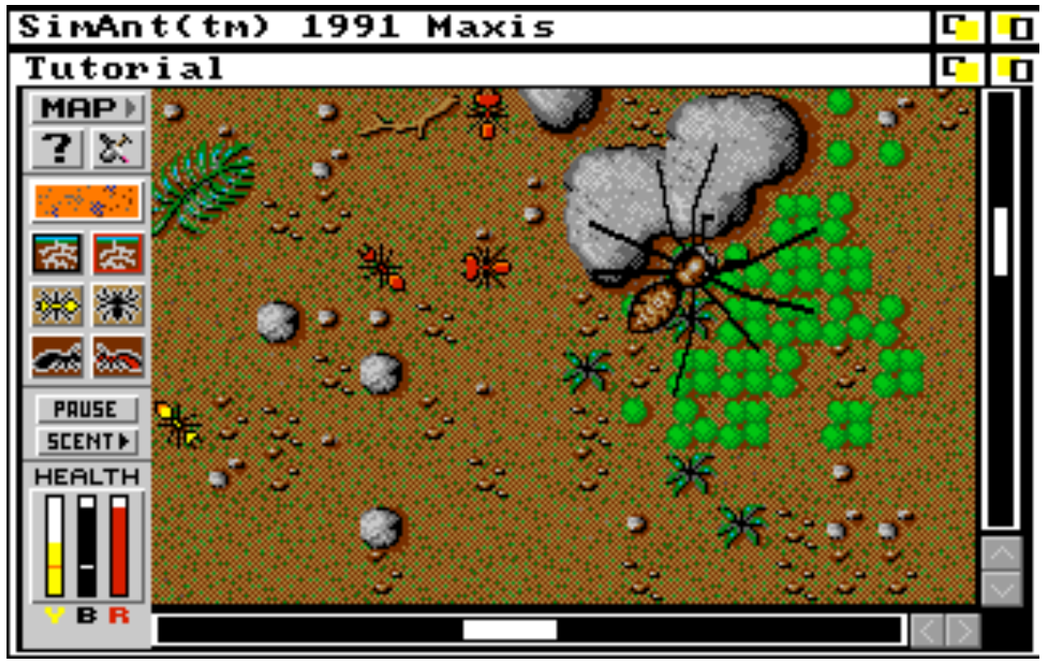
*Shaun White Snowboarding*, UbiSoft 2008


*World Rally Champion 4*, Evolution Studios 2005

Growing plants may be the subject of a game.




Plants provide complex environmental patterns for (often simple) agents to interact with.


## Difficulties with Virtual Plants in Games

**The main difficulty:** complexity of plant model generation is not conducive to realtime rendering.

**Complex geometry**: forests may have thousands of trees, each with thousands of branches, each with thousands of leaves.

**Complex placement and form:** plants of different types grow in different locations.

**Complex and variable form:** plants change form dramatically during their lifetimes, across the seasons and depending on conditions.

*Mt. St. Helens* 2005. Oshyan 3.5 billion polygons!

These might be the first computer generated images of a tree.

They are a branching structures! ...and are therefore similar in form to the trees we specify as data structures in software.

IBM 360, CalComp 565

Programmed in FORTRAN
Produced at McGill University, Montreal
Created 1969 in Montréal, Canada.

Material: computer-generated drawing, ink on paper.
MSU, Museum of Contemporary Art, Zagreb.

from the *Flora* series Petar Milojevic 1968

**Overcoming Difficulties with Virtual Gardening for Games**

## Generating trees

Photograph real trees to use as texture maps.

Model a few different trees by hand.

Model trees *procedurally* - i.e. use software to generate the branching structure.

## Generating the trees

Photographing real trees to use as texture maps is difficult to do well and is very time-consuming.

Modelling a few different trees by hand is possible, but slow, using interactive 3D modelling software.

*Stream scene* 1998, Bernd Lintermann

To automatically generate many unique trees, *procedural* modelling is the way to go!

There are several tree-generating algorithms. We shall look at one, *L-Systems*.

## Lindenmayer Grammars (L-Systems) for Plant Modelling

A plant is viewed as a branching structure whose components are *modules*.

An L-System is a *developmental* model of plant growth: a grammar is used to specify the changing shape of a growing plant.

### What is a grammar?

A set of rules governing what strings are valid or allowable in a language or text.


Prusinkiewicz, P, Lindenmayer, A, Hanan, J.
"Developmental Models of Herbaceous Plants for Computer Imagery Purposes"
Computer Graphics, Vol 22 No. 4, August 1988 (SIGGRAPH 88), ACM Press, pp141-150

Prusinkiewicz, P, Lindenmayer, A.
"The Algorithmic Beauty Of Plants",
Springer Verlag, 1990 [Available in PDF format: `http://algorithmicbotany.org/papers/`]

## How is an L-System specified?

**Alphabet**

A set of symbols called the alphabet of the grammar representing the *modules* of the plant.

e.g. A =  B =  C =  D = 

**Axiom**

An axiom or initial string consisting of symbols from the alphabet representing the first module of the plant.

e.g. A

**Production Rules**

A set of production rules that replace a symbol (called the *predecessor* module) with zero, one or several other modules (called *successor* modules).

e.g.

| *predecessor* modules | | *successor modules* |
|---|---|---|
| p1: A | → | AB |
| p2: B | → | CD |
| p3: C | → | C |
| p4: D | → | D |

## A Simple Example of L-System development

alphabet A =  B = 

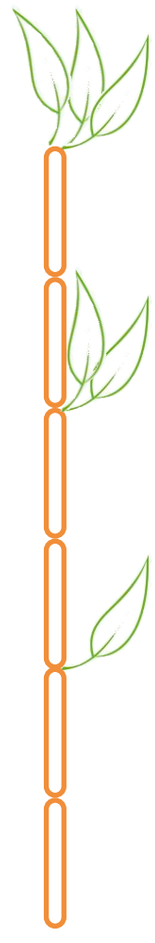
axiom A

production rules p1: A → AAB

p2: B → B

Beginning from the axiom, the production rules are applied to every character in the string in discrete time steps.

At any stage of the process, the string can be converted into an image by applying graphical rules that specify how to draw the individual plant modules.





A
(axiom)

t=0

AAB

t=1

AABAABB

t=2



AABAABBAABAABBB

t=3

etc...

## L-System Branches

A branching structure may be achieved using L-systems by incorporating brackets [ and ] into the alphabet.

alphabet

A = 

B = 

[ = *branch*. Push current position and orientation onto a stack and rotate orientation 45 degrees anticlockwise.

] = *un-branch*. Pop current position and rotation off stack.

axiom A

production rules

p1: A → BB[A]A

p2: B → B

p3: ] → ]

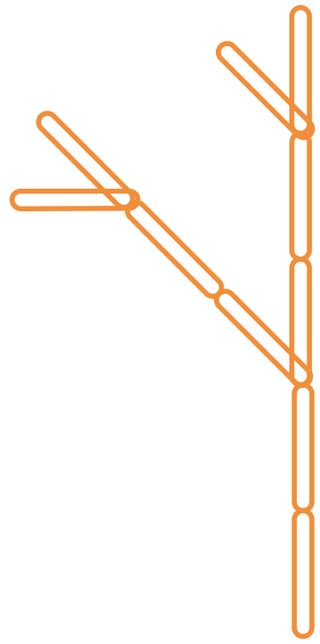
p4: [ → [

A
(axiom)

t=0

BB[A]A

t=1



BB[BB[A]A]BB[A]A

t=2

You can draw this one in your own time.

As you can see, L-Systems generate a *lot* of data automatically.

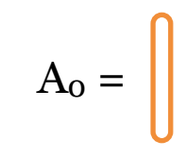
BB[BB[BB[A]A]BB[A]A]BB[BB[A]A]BB[A]A

t=3 etc...

## Parametric L-Systems

We can add a numerical *parameter* to a plant module
to allow for growth of a node over time.

alphabet $A_0$ =



production rules

p1: $X_i \rightarrow X_{i+1}$ where $X \in \{ A, B \}$ and $i>=0$

p2: $A \rightarrow AA$

axiom $A_0$

$A_0$
(axiom)
t=0

$A_0\ A_1$
t=1

$A_0\ A_1\ A_2$
t=2

You can draw this
one in your own time!

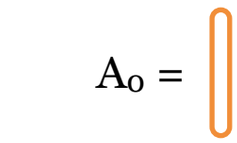
$A_0\ A_1\ A_2\ A_3$
t=3

etc...

## Parametric L-Systems

We can add a numerical *parameter* to a plant module to allow for growth of a node over time.

alphabet $A_0$ =



production rules

p1: $X_i \rightarrow X_{i+1}$ where $X \in \{ A, B \}$ and $i>=0$

p2: $A \rightarrow A$

axiom $A_0$

$A_0$

(axiom)

t=0

$A_1$

t=1

$A_2$

t=2

You can draw this one in your own time!

$A_0 A_1 A_2 A_3$

t=3

etc...

## Non-deterministic L-Systems

In all of the examples above, exactly one production matches each instance of a symbol. This is a *deterministic* L-System.

It is also possible to have multiple rules that *might* apply to a single symbol and to determine which rule to apply in any particular instance using a stochastic mechanism.

This is a non-deterministic L-System.

*Stochastic*, randomly determined; having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.

## Non-deterministic L-Systems

alphabet A B F I L [ ]

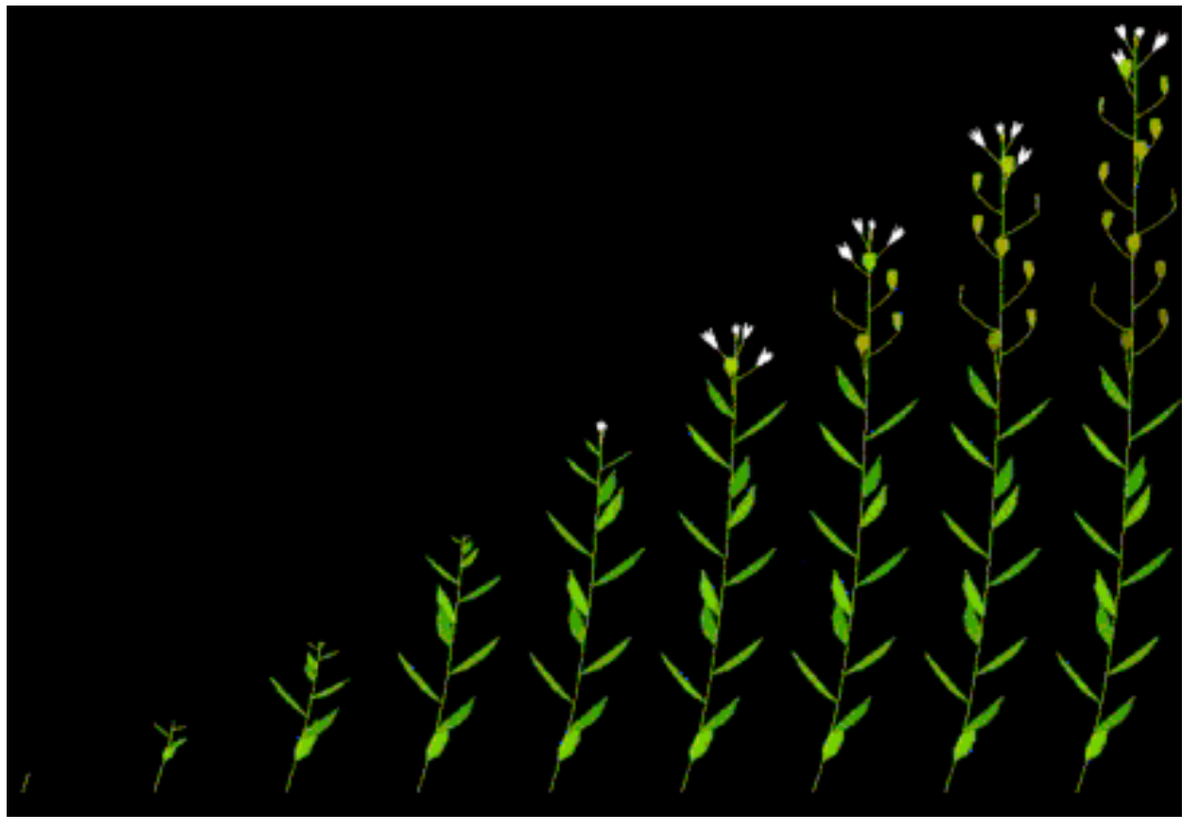
axiom A

production rules

p1: $A \rightarrow I_0 [ L_0 ] A$
p2: $A \rightarrow I_0 [ L_0 ] B$
p3: $B \rightarrow I_0 [ L_0 F_0 ] B$
p4: $X_i \rightarrow X_{i+1}$ where $i \geq 0$ and $X \in \{I, L, F\}$

p1 describes the initial vegetative growth of **L**eaves and **I**nternodes from apex **A**.

At some point, p1 changes to p2 and **A** is changed to its flowering state **F**.



Development of Shepherd's Purse, Prusinkiewicz et. al 1988

How would you implement an L-System parser?

## Controlling Non-deterministic L-Systems

**A delay mechanism** : ambiguity is avoided in the selection of production rules by adding a mechanism to the grammar which, after some number of iterations of a rule (or set of rules), replaces one production with another.

**A stochastic mechanism** : ambiguity is avoided by specifying a probability with which each ambiguous rule in a particular instance may operate in precedence over all others. This allows variation in plant models that use the same rule-set.

**Environmental change** : the entire set of production rules may be altered to another set after some external factor triggers the change.

*The Sims 3*, Electronic Arts Inc. 2009

Image: JL. Power, AJB. Brush, DH. Salesin, P. Prusinkiewicz, *Interactive Arrangement of Botanical L-System Models*, 1999 ACM Symposium on Interactive 3D Graphics.

## Rendering Forests for Interactive Frame Rates

Plant geometry, including that generated by L-Systems, is complex.

Rendering multiple trees at interactive frame rates can stretch graphics hardware.

Tricks:

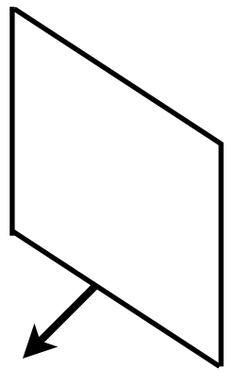
Generate a few images of trees and map them onto *billboard* polygons.

Pre-render some detailed tree and forest models and use them as a background.

## Billboarding

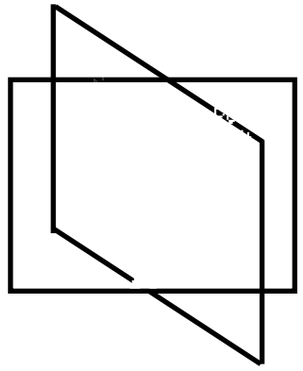
A billboard is a polygon, onto which a texture is mapped.

The billboard can be rotated in space to face the camera view point so that no “edge on” view of the polygon is seen by the player.



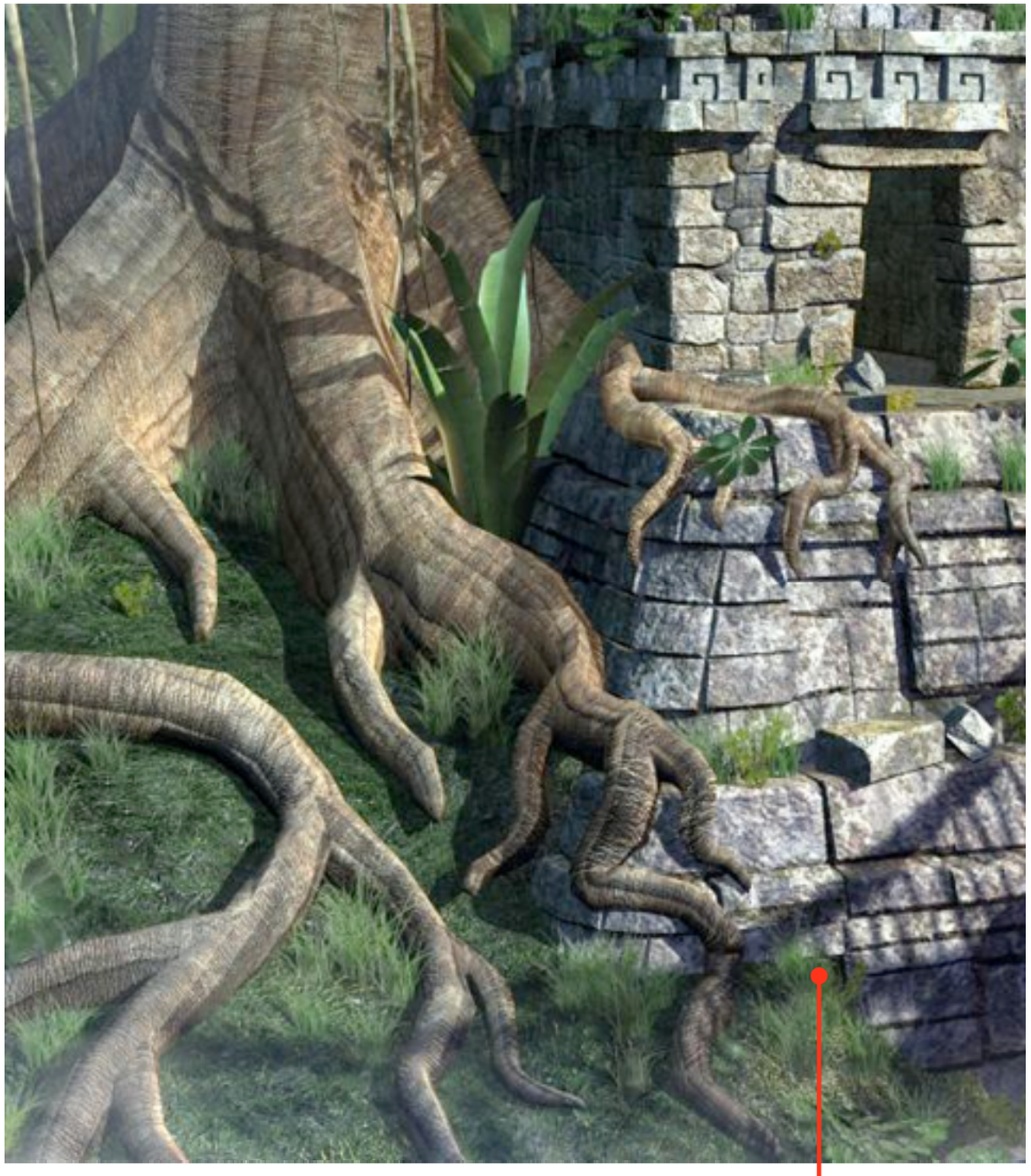
Orient polygon normal towards the camera.

An X-billboard can be rendered to present the view from different sides so that no “edge on” view of the image is seen by the player *and* there is no need to rotate the billboard when the camera position moves.



Multiple billboards can give a good impression of a forest, whilst avoiding the need to generate and render complex geometry. Instead, only a few polygons need to be texture mapped.

Billboards are also easy to test against for collisions!



Architectural model, Kordela Studio
www.kordelastudio.com

Billboarding used on
long grass and weeds.

*Jungle Boogie*, Ando, 2007

Billboarding used on all
trees and grass.

Complex plant geometry can be pre-rendered and mapped to a background cube, sphere, cylinder or polygon as a texture.

This can be combined with textured billboards or simple geometric models in the foreground.

*Soul Caliber*, Namco

## Forest layout and terra-forming: for class discussion

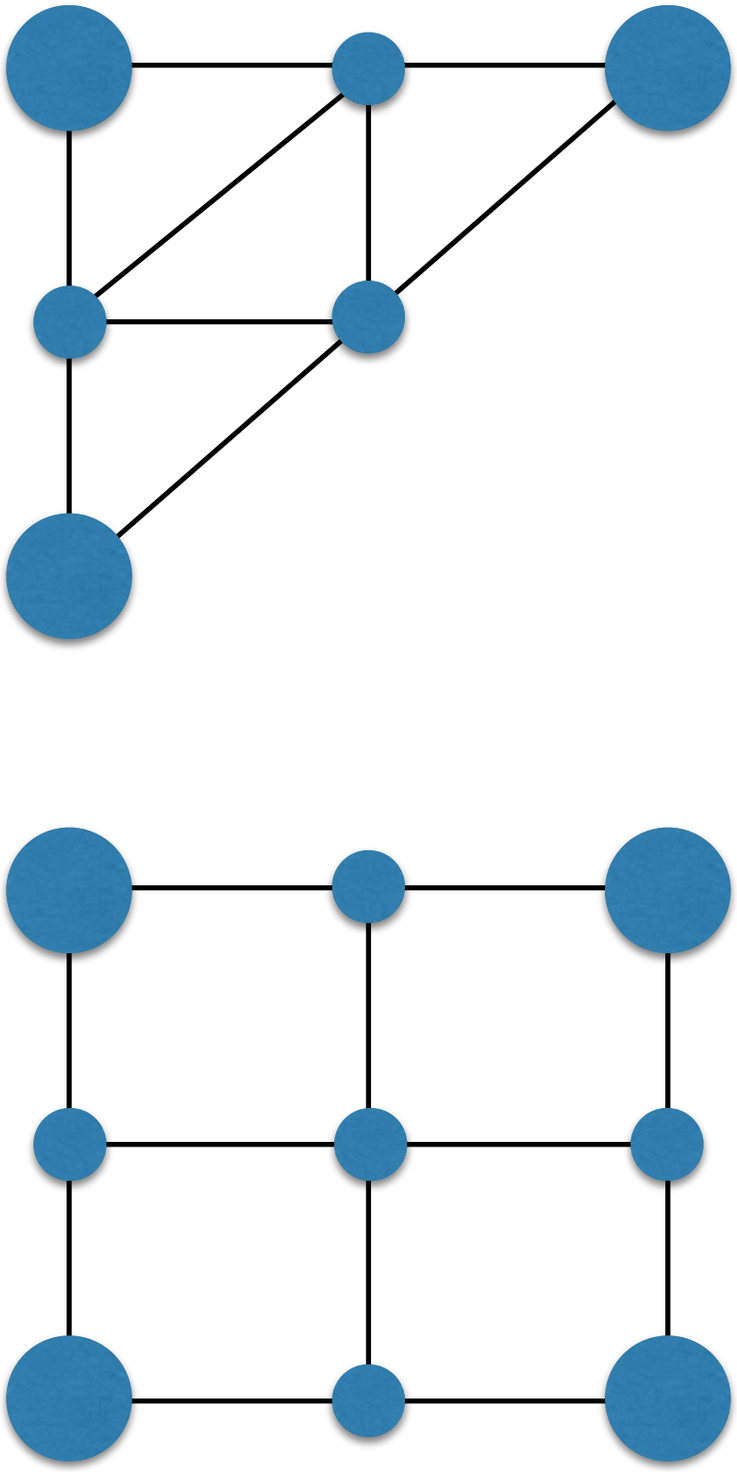
A *height map* represents the height ***y*** of a point in the ***x / z*** plane as the brightness of each pixel.

This information is converted to a series of triangle vertices to create a 3D landscape.

Given a landscape model represented as a height map, how could we write software to intelligently place foliage including grass, shrubs and trees of different types?

Images by *Mike*.
blog.nostatic.org/2007/11/3d-landscape-rendering-with-texture.html

# Terra-forming using mid-point displacement : for class discussion



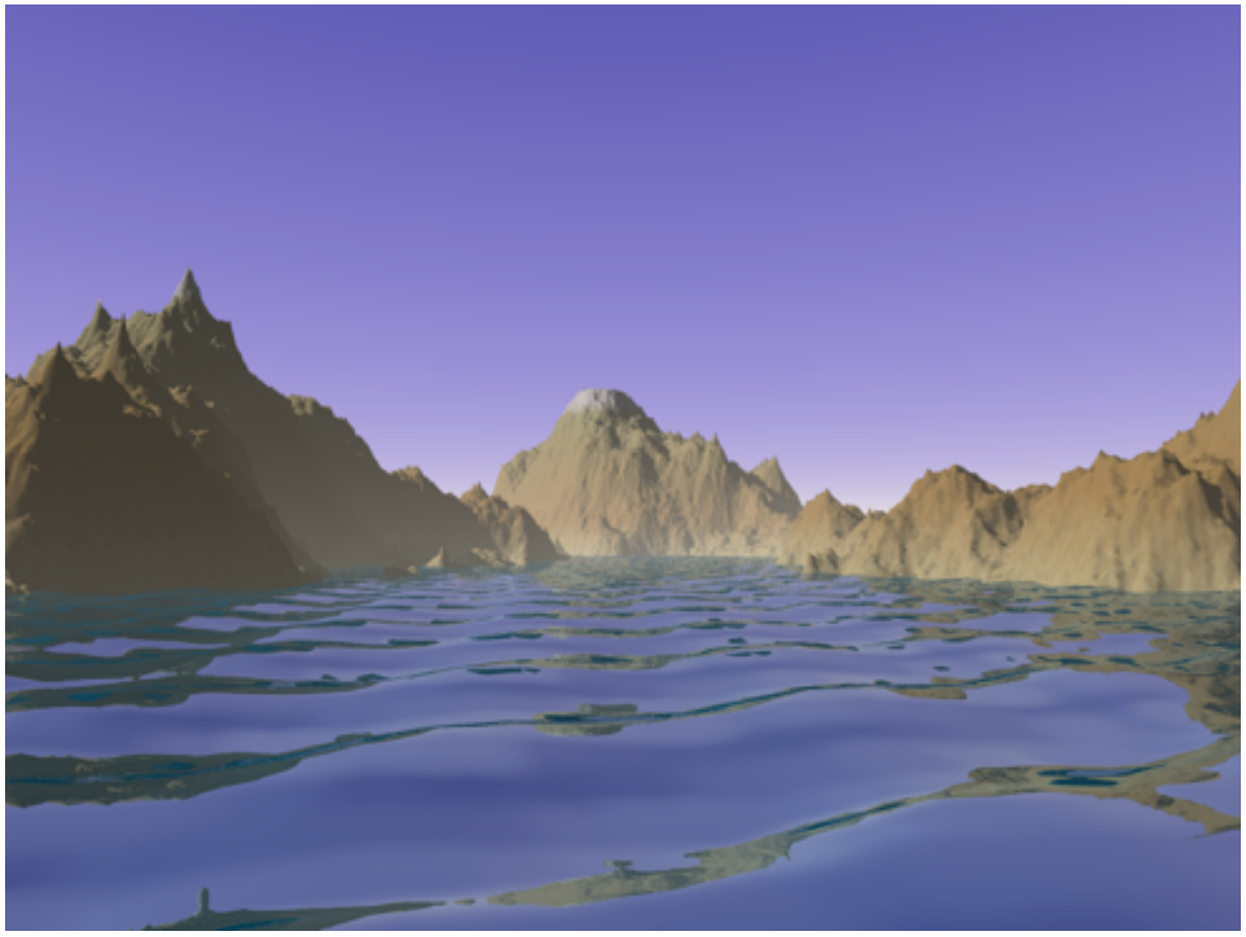


Image © Adam Koh, 2002

**Have you met the learning objectives?**

Can you list a few different applications of plant models to computer games?

What is an L-System? How does it work?

Could you write some simple L-System grammars and evaluate them by hand?

What strategies could you use to generate and interactively render a forest scene?