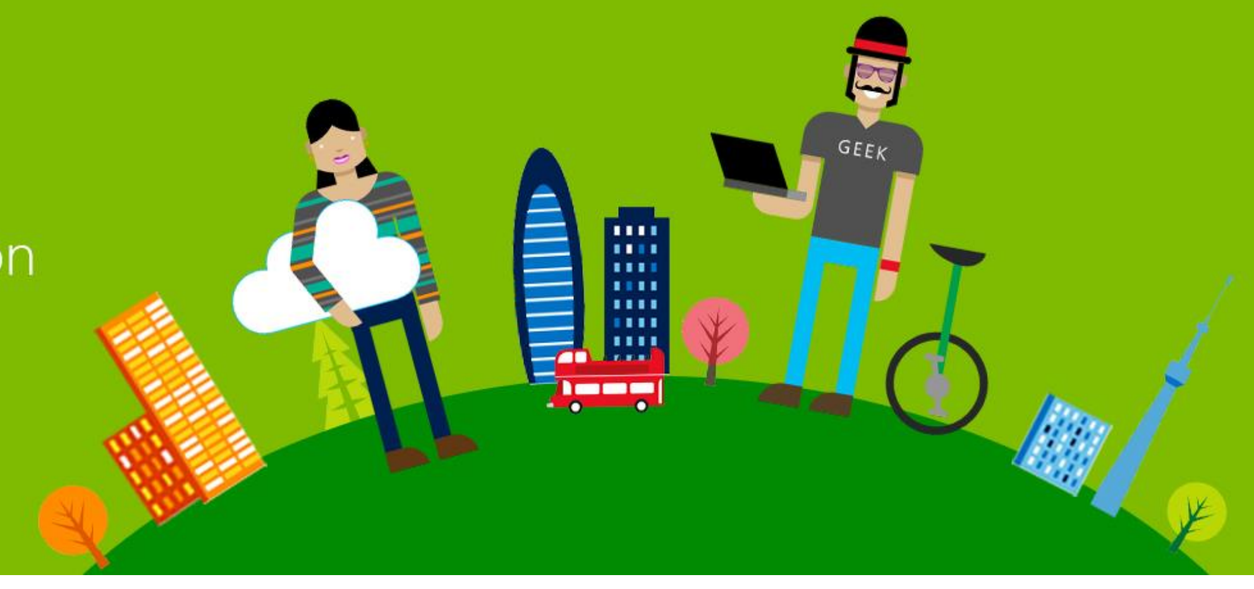
Microsoft Research Asia

Student TechFest

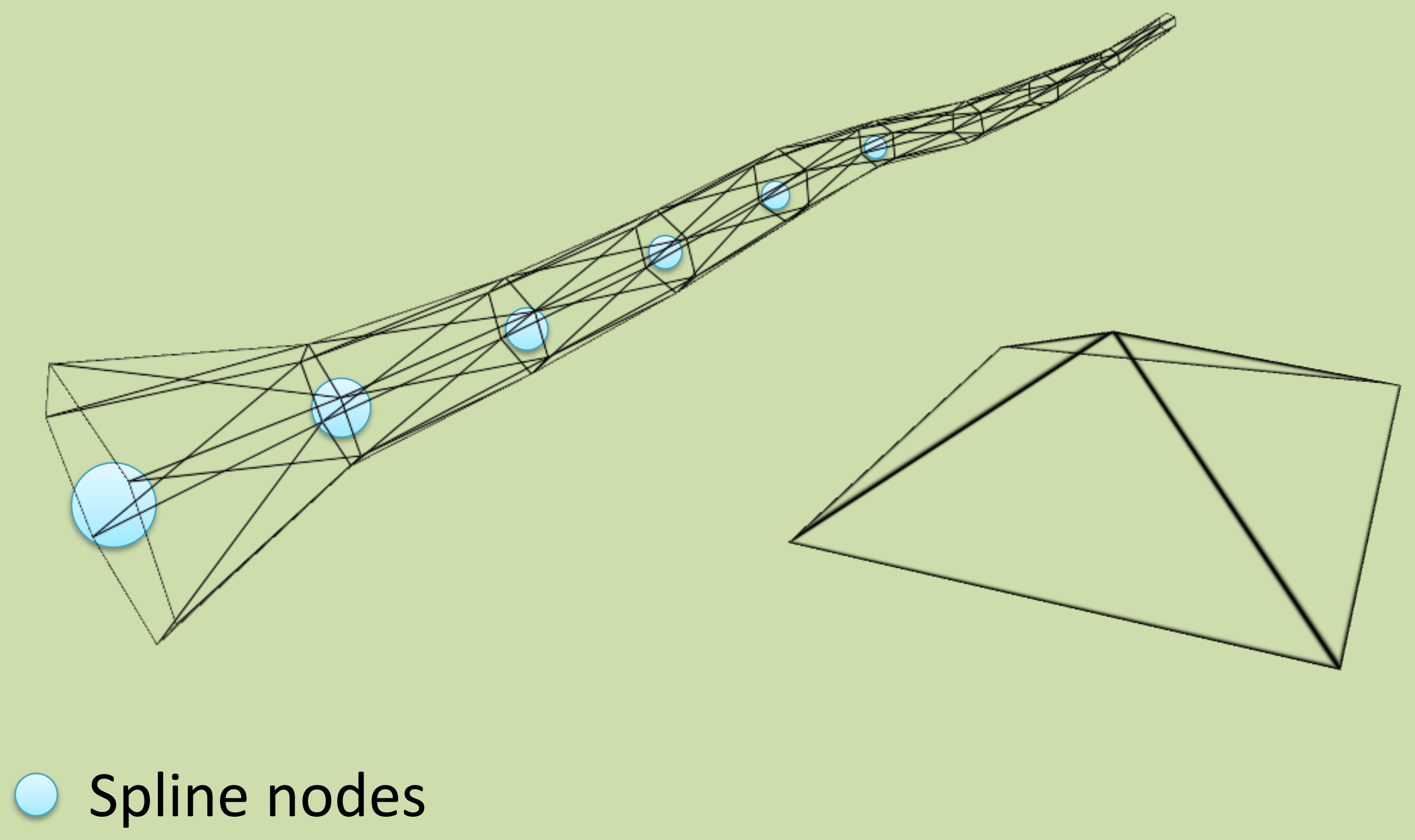
2016

Empower every person and every organization
on the planet to achieve more.



Real-time Realistic Procedural Fractal-based Tree Generating, growing and rendering

Geometry : Branches and leaves



Branches was controlled by a "spline". Each spline was controlled by 2~10 nodes. The i th node controls the $6i \sim 6i+5$ th vertices of the branch.

Nodes has normalized direction and radius. The radius of i th node R_i equals :

$$R_i = \left((R_{end} - R_{start}) * \frac{i}{N_{node}} + R_{start} \right) * x, where\ x \sim U(0.9, 1.1)$$

R_{start}, R_{end} are const. Unfortunately, nodes has a same length.

I use a simple pyramid geometry for leaves.

Fractal Iteration & Growing

Child branches grows by "steps". There're 1 leaves layer, 3 branch layer and 1 main branch layer.

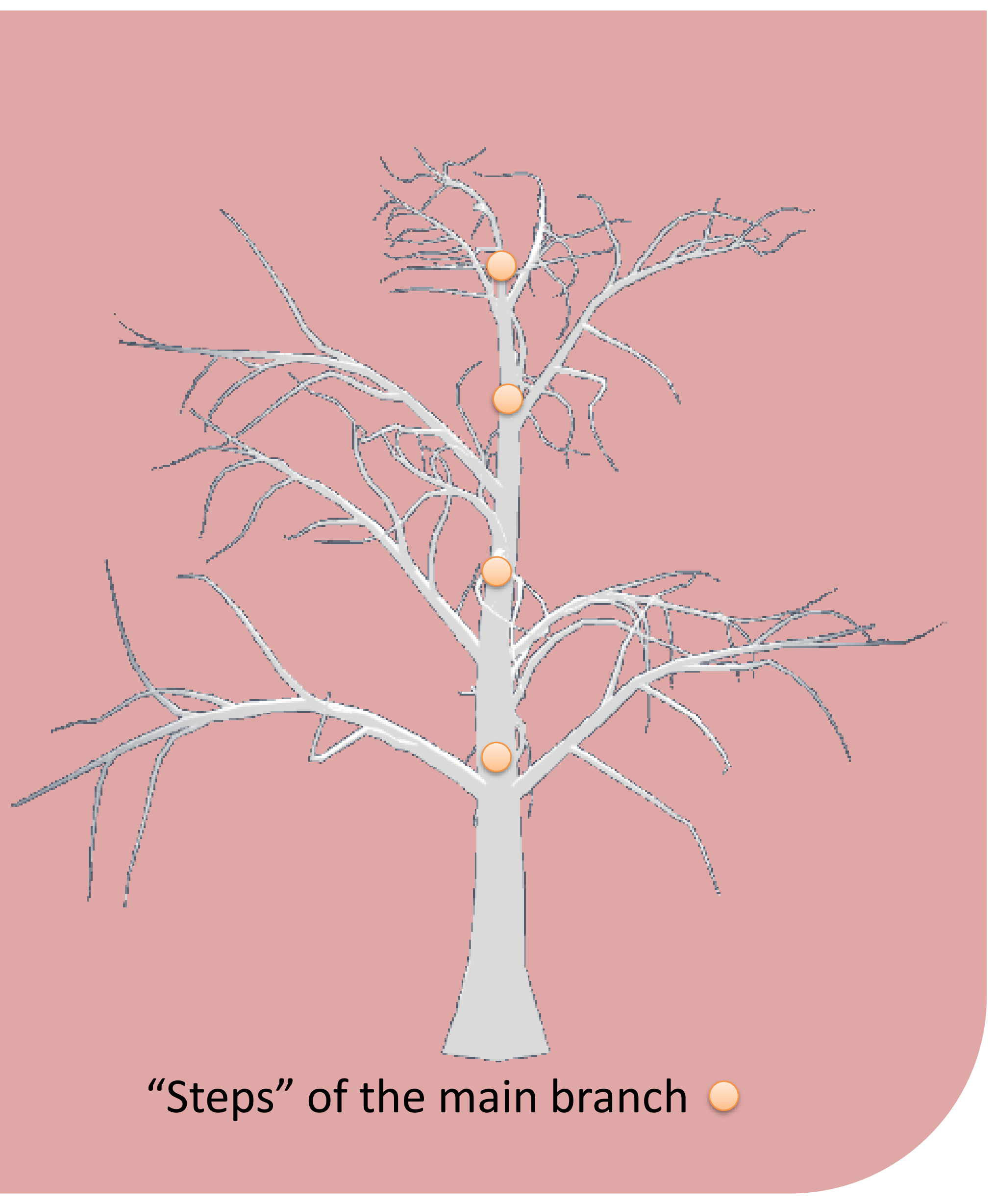
Branches are affected by gravity and gnarl force, and a set of perlin noises.

Branches has grow rate G_b. If $G_b > G_{grow}$, then generate children for this branch ($G_{grow} = 0.4$ in the demo) .

Parameters of child branch (direction, G_i etc.) was calculated from its parent branch by the child's relative position and random numbers.

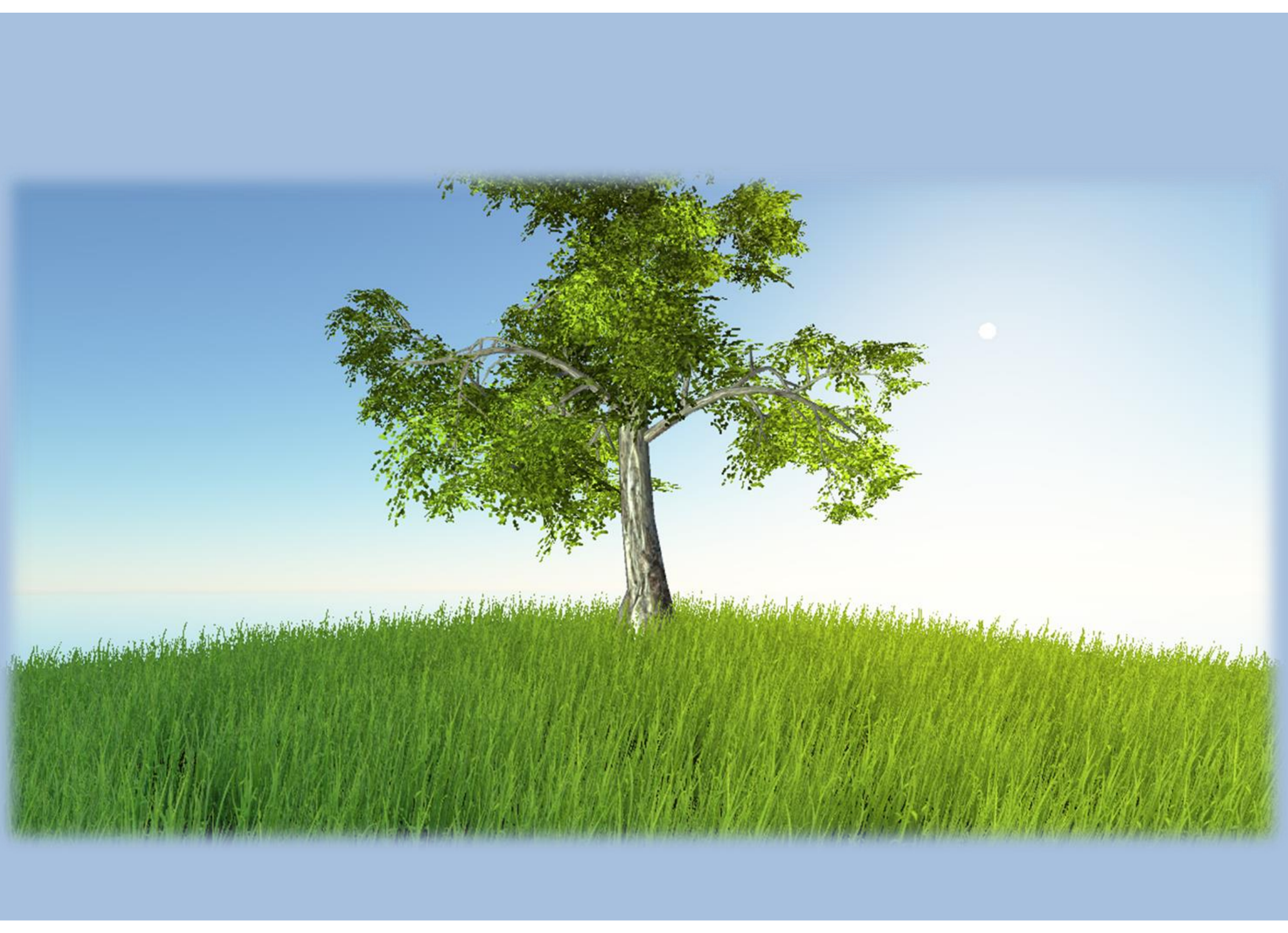
Layer	Steps	Child/Step
Main	4	3
Branch 1	5	1
Branch 2	5	1
Branch 3	1	1 (leaves)

In the demo, the maximum G_b value of the main branch is 18. Adjust this will grow the tree.

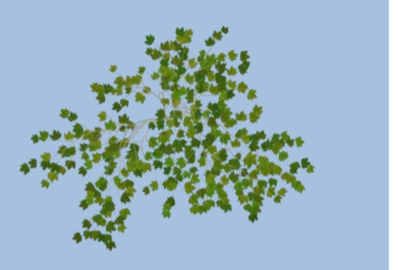


"Steps" of the main branch ●

Shading: Branch, Leaves, Sky and misc.

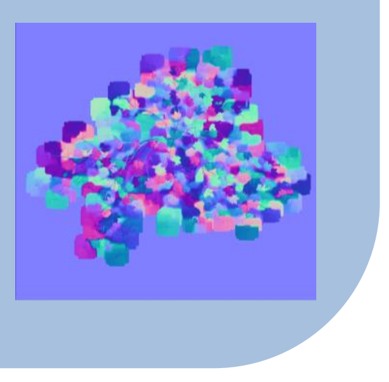


A 512x512x24bit, Depth only Shadow map was used as the tree's self-shadow.
Normal mapping & basic phong shading on leaves.



A trick changing color of leaves/grass under direct light to camera:

$Color_{new} = (Red * 0.9, Green, Green * 0.2)$ (From GPU gems 3 Ch4)



Sky shading algorithm from:

Precomputed Atmospheric Scattering by E.Bruneton & F.Neyret 2008 [1]

…

Performance (@i7-4700MQ / 16GB RAM / 2x NVIDIA GTX 780m(SLI))

Idle: Stable 60fps with Vsync
Growing (while changing G_{start}) : BAD, about 15~35 fps

Work	Cost when $G_s = 1$	Cost when $G_s = 18$
Fractal iteration	30.01ms (74.8%)	54.03ms (83.2%)
Garbage collection	5.31ms (13.2%)	5.84ms (8.9%)
Render cubemap	0.84ms (2.7%)	0.99ms (1.5%)
Main&shadow cam	0.51ms (1.2%)	0.57ms (0.7%)

Controls:

W,A,S,D	moving cam
Q,E	Zoom in/out
R	reset random seed
+ / -	Adjust G_{start}

*A simple water plane uses 256x cubemap. C# in Unity introduces GC. Main cam: 763x535x32

2.1M Triangles, 44.7k Vertices.
44.7MB memory allocated for total 559x Vertex buffer objects,
10.1MB memory allocated for total 19x texture maps.

Future works

- Ambient Obscurance
- Compute fractal on GPU (?)
- Move it to low-level APIs for memory management (d3d12/vulkan)
- Tree animations during wind
- More realistic shading
- REDUCE MAGIC NUMBERS
- Etc..

Ref: [1] Eric Bruneton, Fabrice Neyret. Precomputed Atmospheric Scattering. Computer Graphics Forum, Wiley, 2008, Special Issue: Proceedings of the 19th Eurographics Symposium on Rendering 2008, 27 (4), pp.1079-1086. .

Yuhui Zhang, ARD Incubation

Microsoft Research