Representation of natural scenes

Teacher: A. Prof Chengying Gao(高成英)

E-mail: mcsgcy@mail.sysu.edu.cn

School of Data and Computer Science

Outline

- Representation of natural scenes
  - Fractal(分形)
  - L-System (L系统*)
  - Particle system (粒子系统*)
  - Cloth simulation

Representation of natural scenes

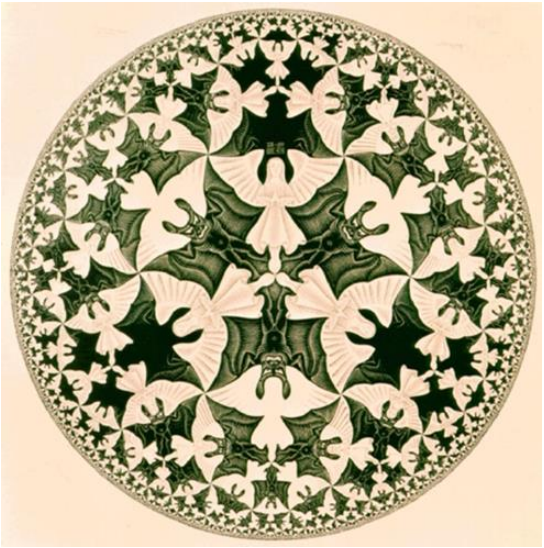
- Modeling natural scenes is a challenge problem
  - Mountains, trees, flowers and grass,flame,cloud, smoke, fluid，cloth.....
- Three kinds of approaches
  - Fractal (分形)
  - L-System based on grammar rules (语法规则)
  - Particle system (粒子系统)

Fractal (分形)

- They’re famously found in nature. Fractals purely a wonder--- too irregular for Eulidean geometry; iterative and recursive and seemingly infinite.
- They turn up in food and germs, plants and animals, mountains and water and sky.



Fractal (分形)



Fractal (分形)

- Main features of fractal
  - Self similarity (自相似性)：local regions of a fractal are similar to the fractal itself
  - Infinitesimal detail (无限小细节)：A fractal always exhibits details no matter how small the region is
  - Fraction dimension (分数维)

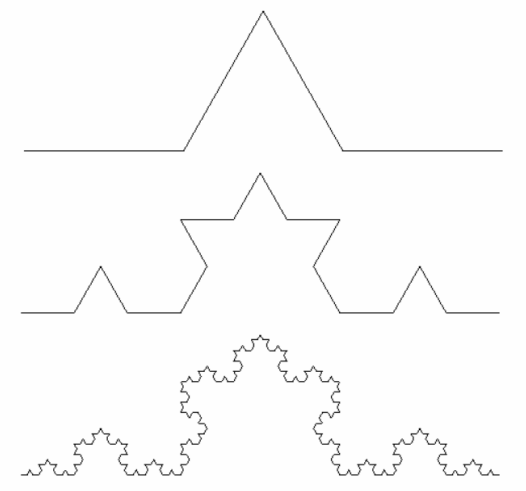
- In computer graphics, we use fractal functions to create complex objects.

Dimension of fractal

- s—scale of the primitive(图形缩放尺度)
- n—the number of primitives used to replace the primitive (替换图元的个数)
- D—the dimension of the fractal(维数定义)

$$ns^D = 1 \Longrightarrow D = -\frac{\ln n}{\ln s}$$

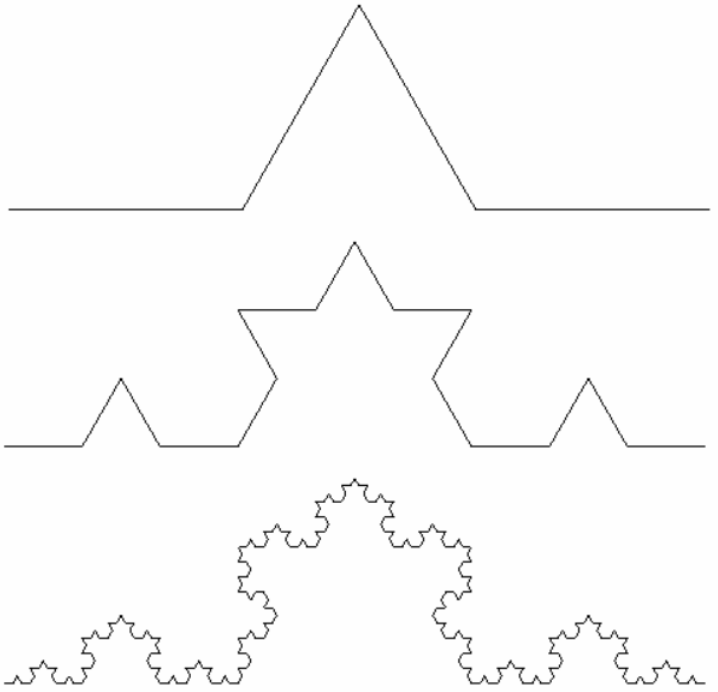
n越大, s越接近1➔D越大



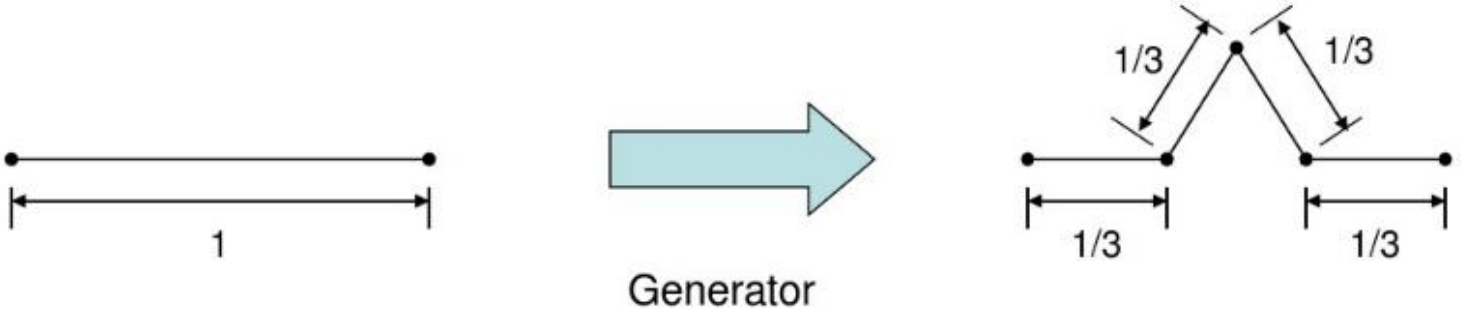
Example： Koch snow flakes

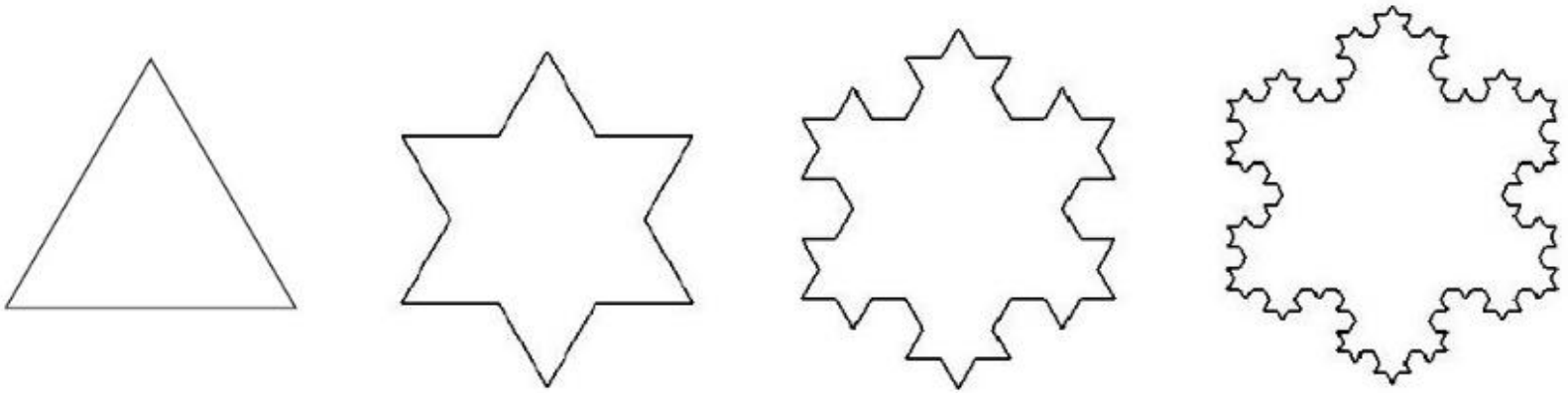
- Koch snow flake curve (怪兽曲线)
  - Initial phrase： four segement
  - 迭代：一段折线由初始折线缩小1/3原
  - Repeat the process

$$D = -\frac{\ln 4}{\ln 1/3} = \frac{\ln 4}{\ln 3}$$



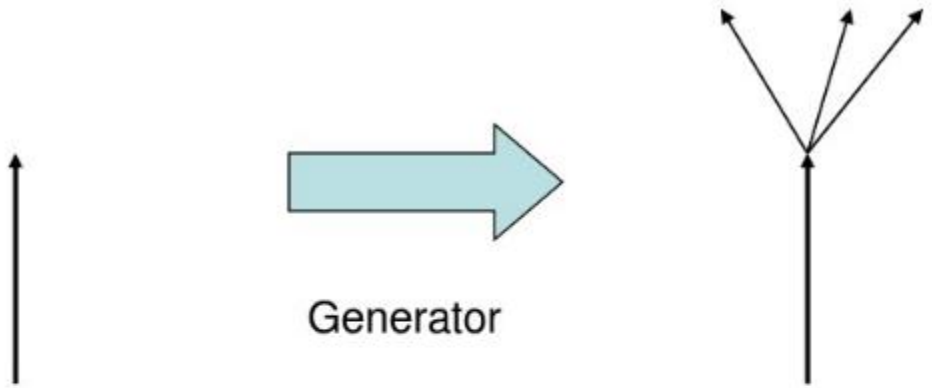
Example： Koch snow flakes

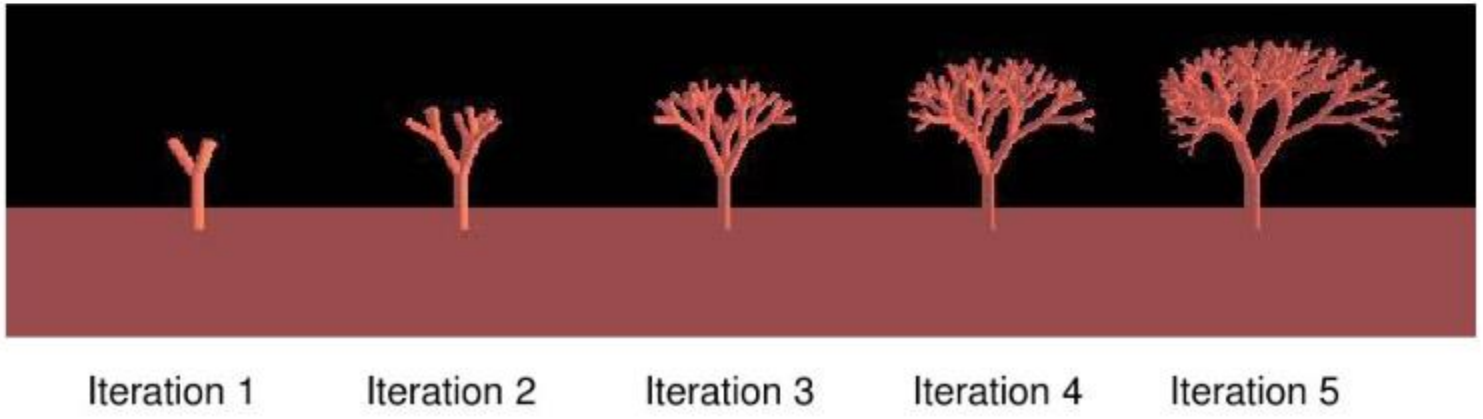




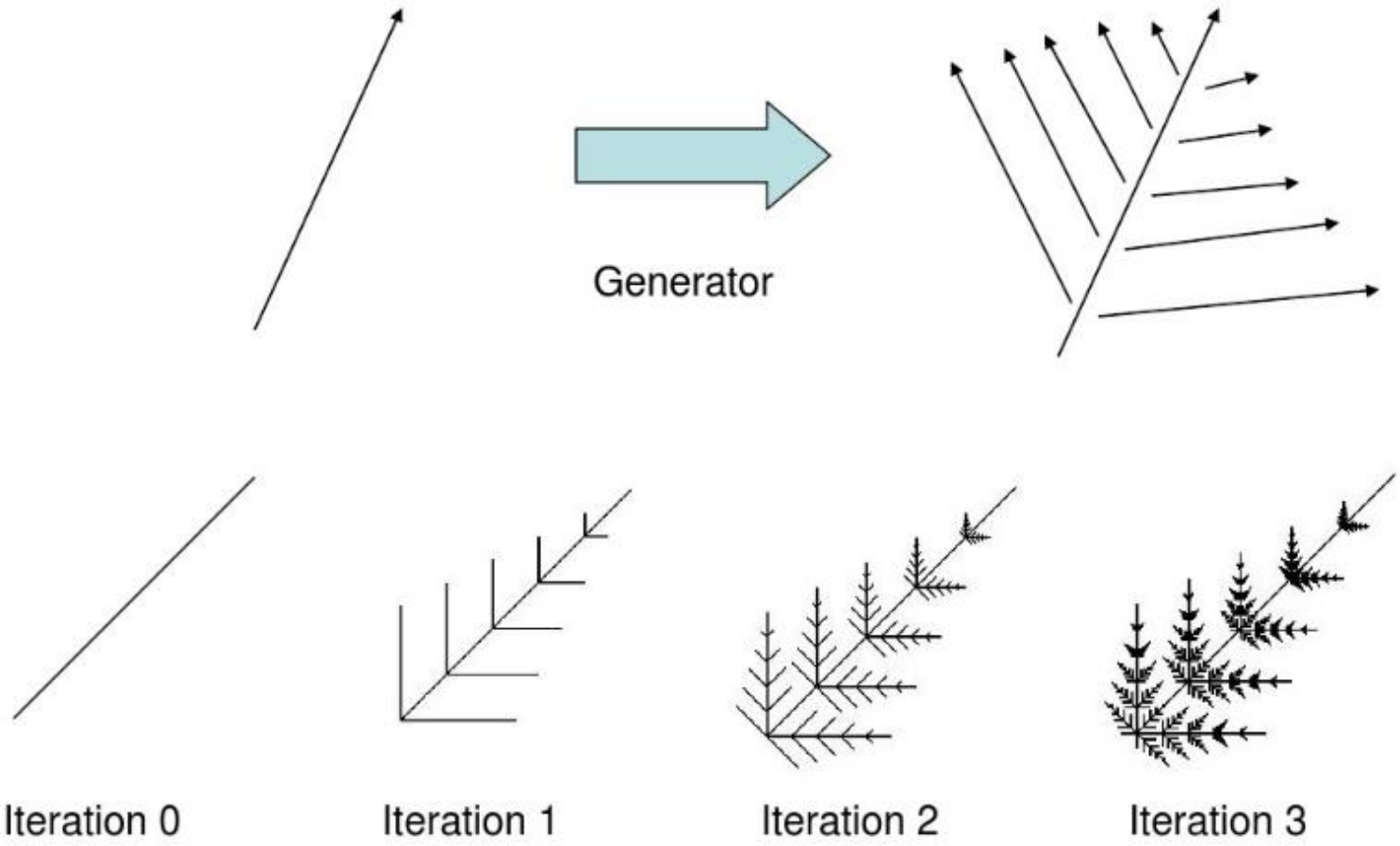
Iteration 0 Iteration 1 Iteration 2 Iteration 3

Example： Fractal tree

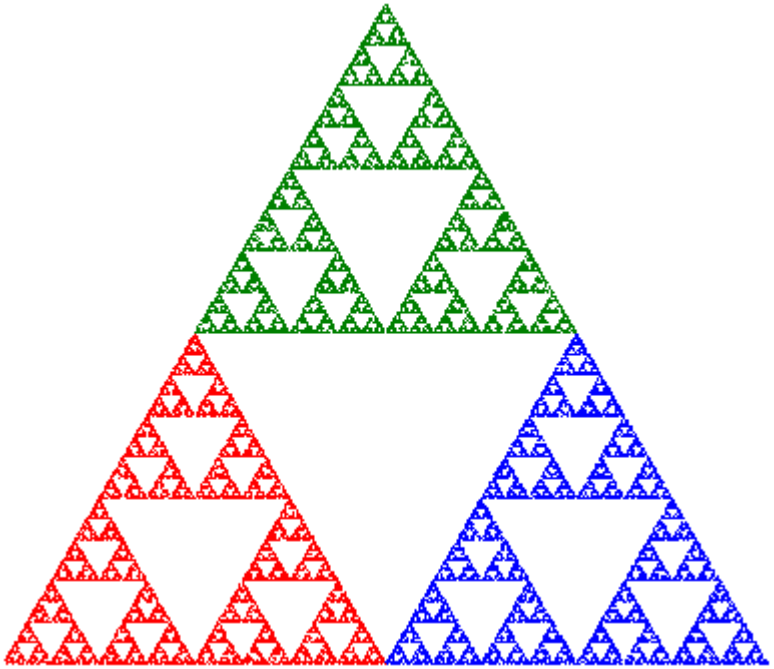


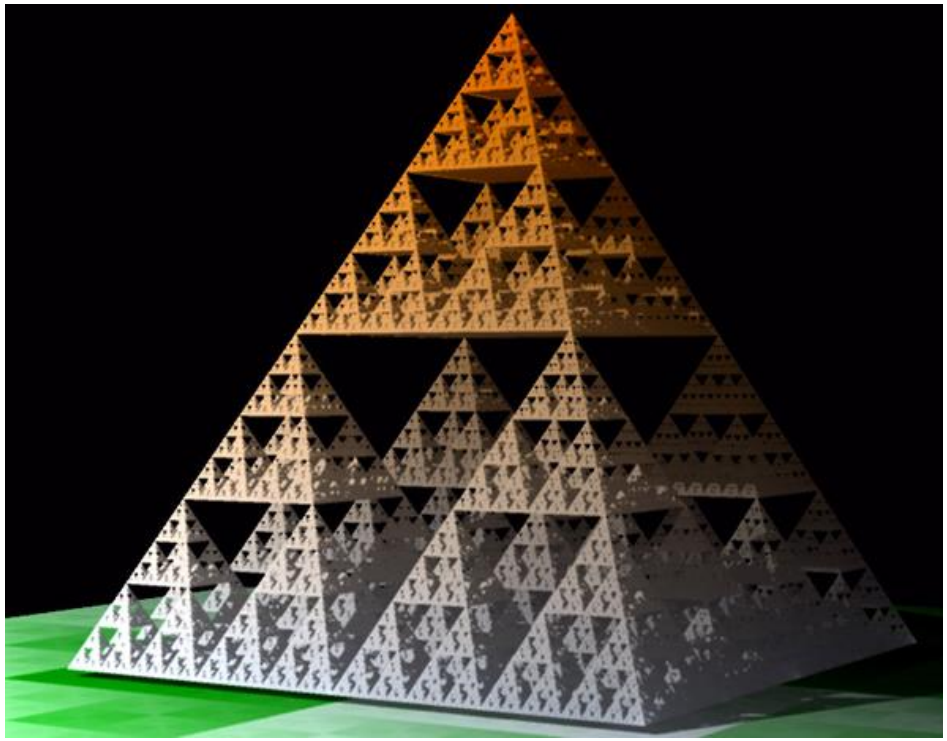


Example： Fern



Sierpinski Triangle/Pyramid(谢尔宾斯基三角形)





www.wikimedia.com

Add Some Randomness

- The fractals we've produced so far seem to be very regular and "artificial".
- To create some realism and variability, simply change the angles slightly sometimes based on a random number generator.
- For example, you can curve some of the ferns to one side.
- For example, you can also vary the lengths of the branches and branching factor.

Mountain generation based fractal

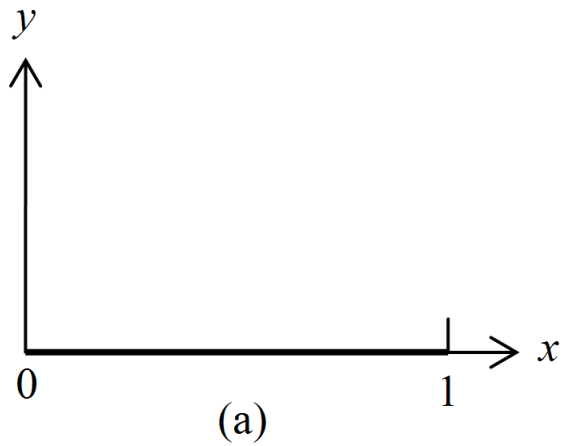
- 一维分形山
  - Let (x_i, y_i)、(x_{i+1}, y_{i+1}) be two endpoints, the newly inserted point (x_{new}, y_{new}) is calculated as：

$$x_{new} = \frac{1}{2}(x_i + x_{i+1})$$

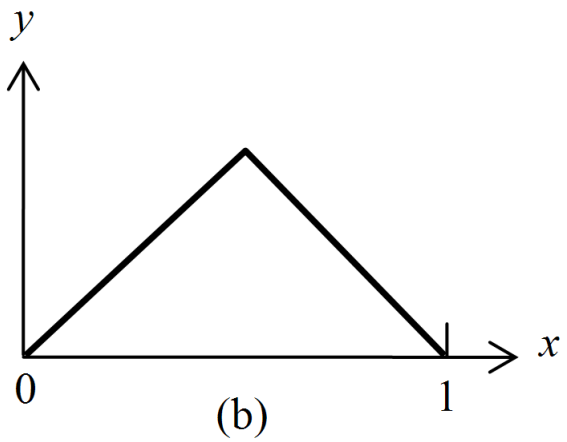
$$y_{new} = \frac{1}{2}(y_i + y_{i+1}) + \mathrm{P}(x_{i+1} - x_i)\mathrm{Random}(x_{new})$$

  - Random(•) generates a random number in [0, 1], P(•) is used to control the scale. For example, in the s-th round of iteration, we set $\mathrm{P}(s)=2^{-s}$

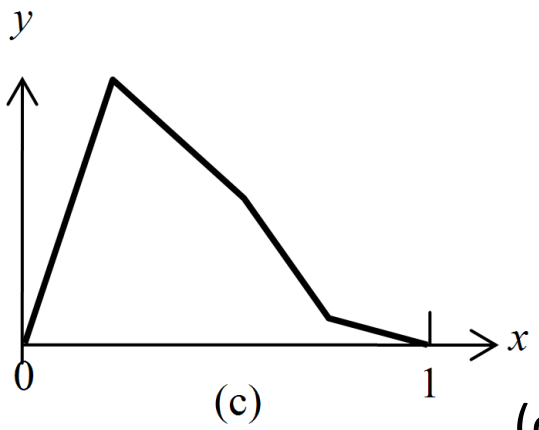
1D Mountain based on fractal: example



(a) Given line segment



(b) Move the midpoint along y-axis by a random distance



(c) Repeat the process again

Terrain(Random Mid-point Displacement)

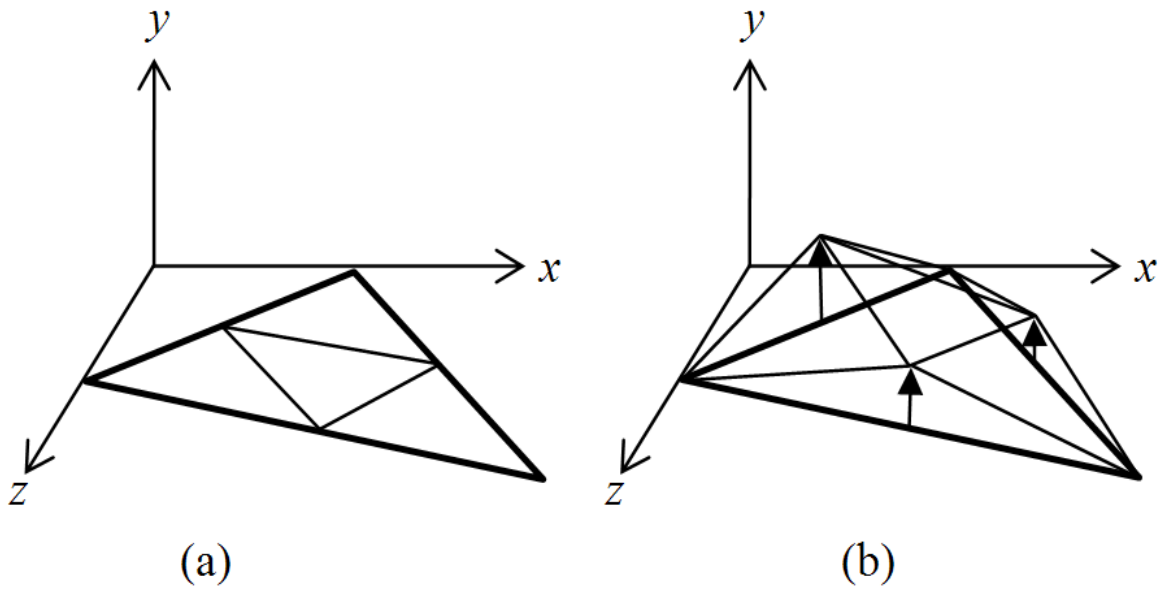
- Given the heights of two end-points, generate a height at the mid-point.
- Suppose that the two end-points, are a and b. Suppose the height at a is y(a), and the height at b is y(b).
- Then, the height at the mid-point will be:

$y_{mid} = (y(a)+y(b))/2 + r$, where
  - r is the random offset

- This is how to generate the random offset r:

$r = sr_g|b-a|$, where
  - s is a user-selected "roughness" factor, and
  - r_g is a Gaussian random variable with mean 0 and variance 1

2D mountain: example



(a) A triangle is split into four. (b) The midpoint of each edge is disturbed along y-direction

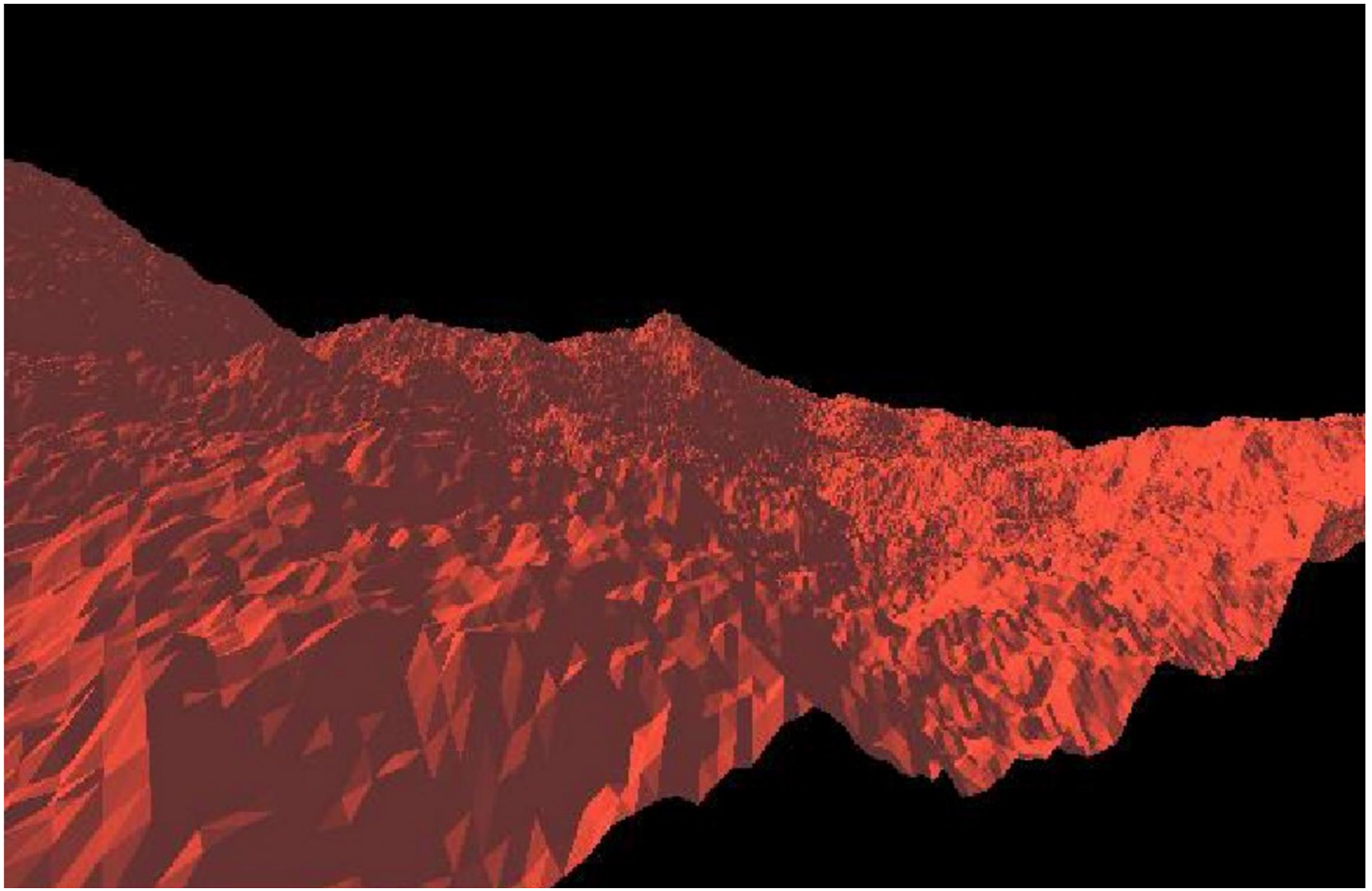
Remarks: You can

1. use a more complex initial mesh
2. repeat the refinement using subdivision topologic rules

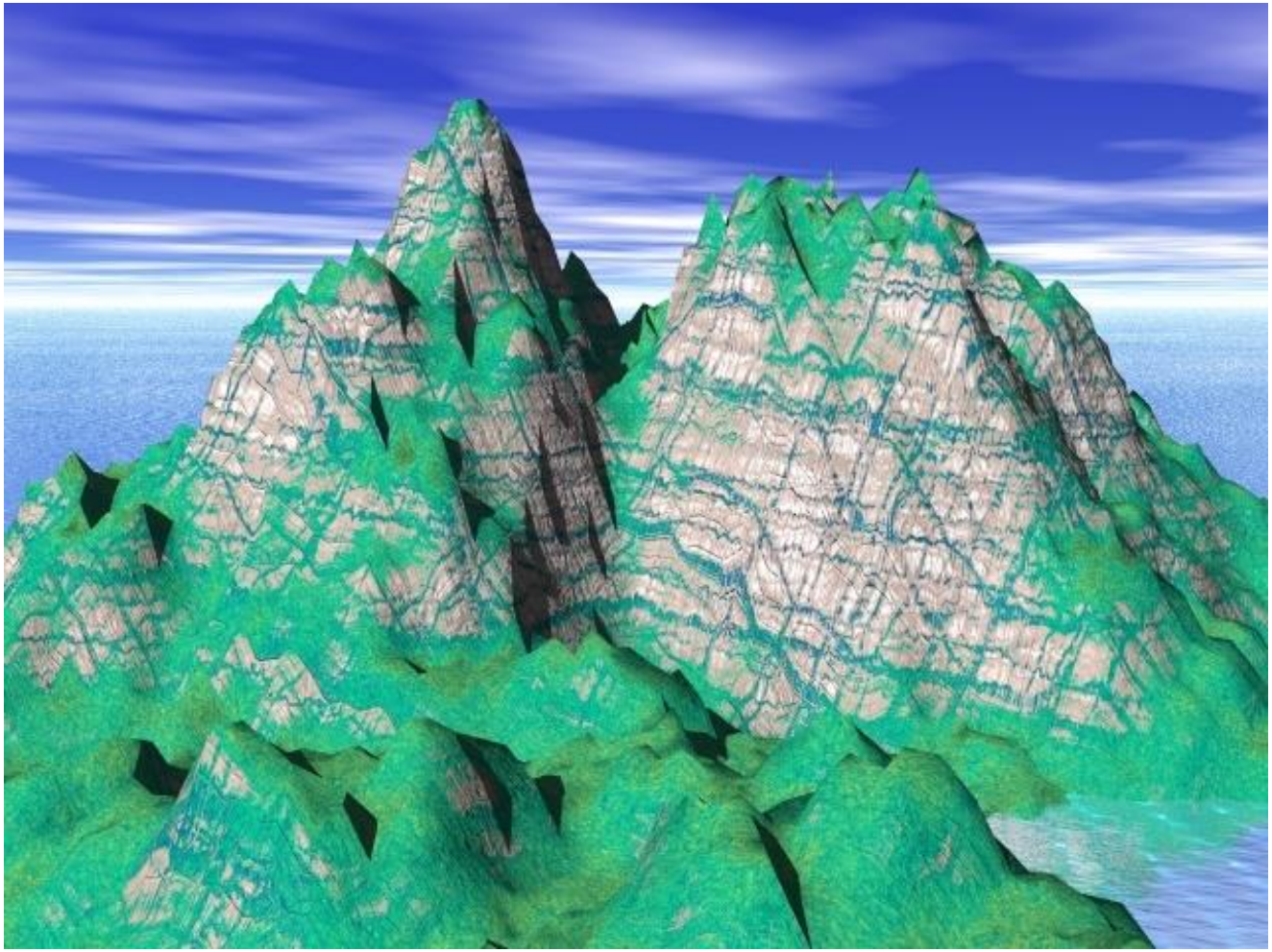
Terrain(Random Mid-point Displacement)

```
// given random numbers x1 and x2 with equal distribution from -1 to 1
// generate numbers y1 and y2 with normal distribution centered at 0.0
// and with standard deviation 1.0.
void Gaussian(float &y1, float &y2) {

        float x1, x2, w;

        do {
                x1 = 2.0 * 0.001*(float)(rand()%1000) - 1.0;
                x2 = 2.0 * 0.001*(float)(rand()%1000) - 1.0;
                w = x1 * x1 + x2 * x2;
        } while ( w >= 1.0 );

        w = sqrt( (-2.0 * log( w ) ) / w );
        y1 = x1 * w;
        y2 = x2 * w;
}
```

Terrain(Random Mid-point Displacement)



Mountain surface: example



Outline

- Representation of natural scenes
  - Fractal(分形)
  - L-System (L系统*)
  - Particle system (粒子系统*)
  - Cloth simulation

L-systems (Lindenmayer system)： Plant modeling

- Aristid Lindenmayer（林登麦伊尔) ,17/11/1925 –30/10/1989, Hungarian biologist.
- 1968 developed L-system to model the behaviour of cells of plants.
- L-systems nowadays are also used to model whole plants.

http://algorithmicbotany.org/papers/

L-system : Plant modeling

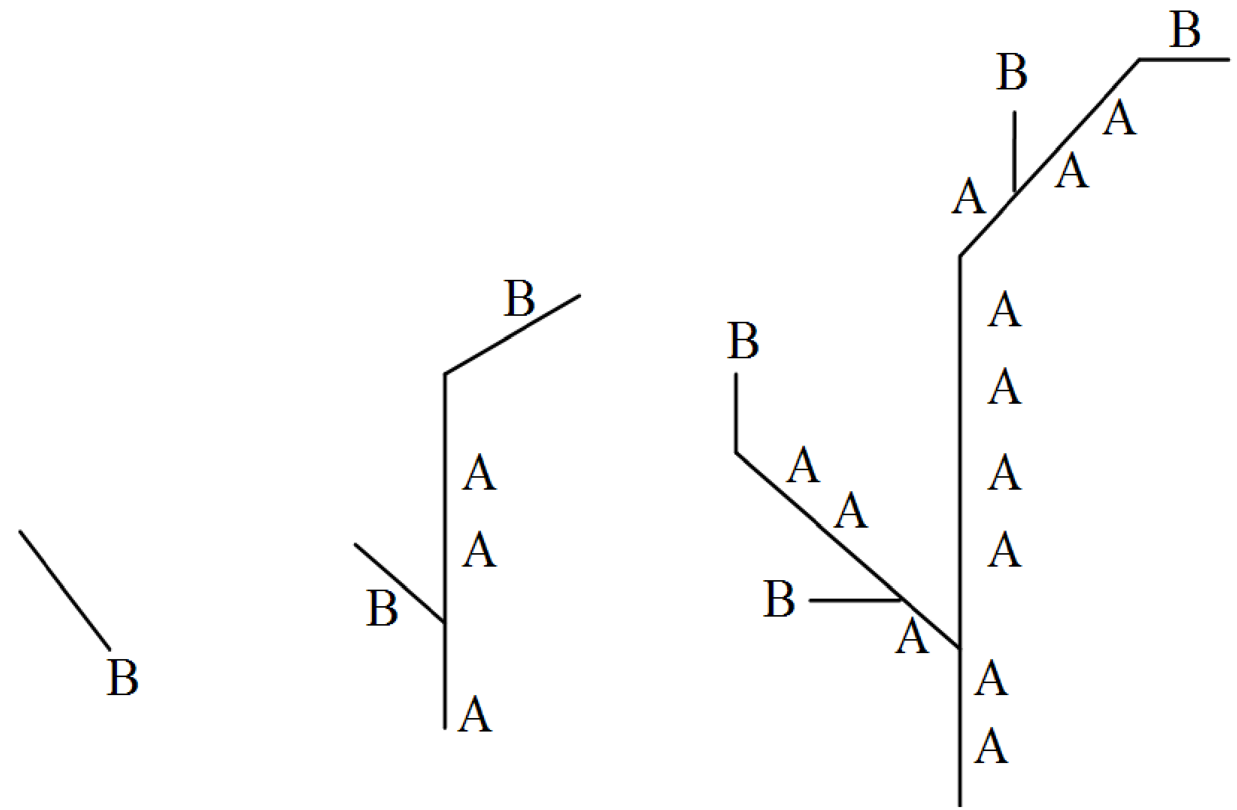
- Grammar rules： a grammar rule replaces a string using a new one
  - The iteration result is called a generation
  - Character explanation： use geometric primitives to replace the characters

Tree: example

- Characters： “A”, “B”, “[” “]”, “(”, “)”
- Grammar rule：
  - A→AA; B→A[B]AA(B)
  - 2 Iterations
    - B→ A[B]AA(B) → AA[A[B]AA(B)]AAAA(A[B]AA(B))
- Character explanation(字符解释)：
  - “A” stands for a stem; “B” is a branch
  - “[]” : the branch turns left with 45°
  - “()” : turn right, 45°

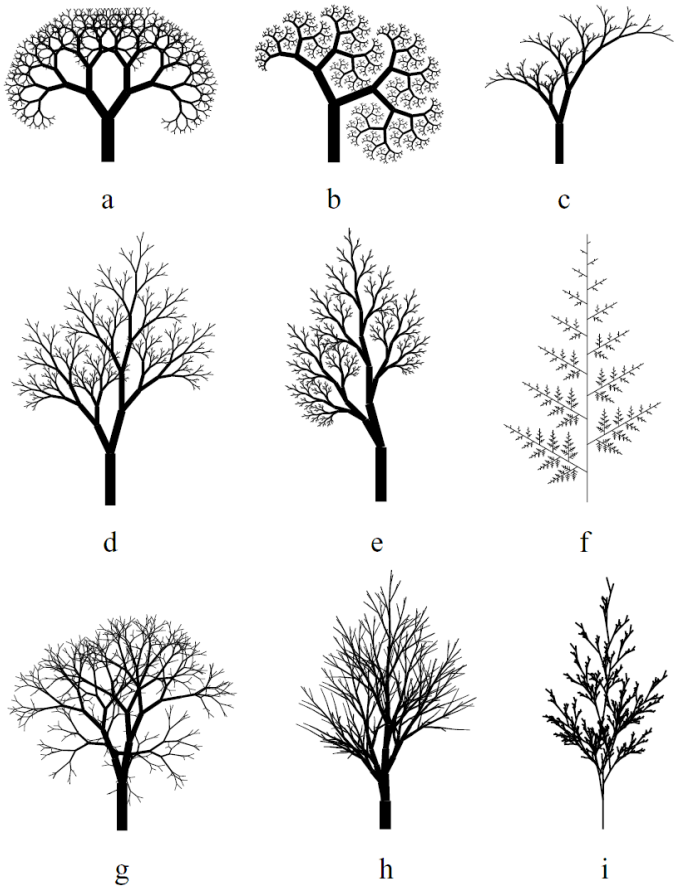
Background: http://mathworld.wolfram.com/LindenmayerSystem.html

(continued)



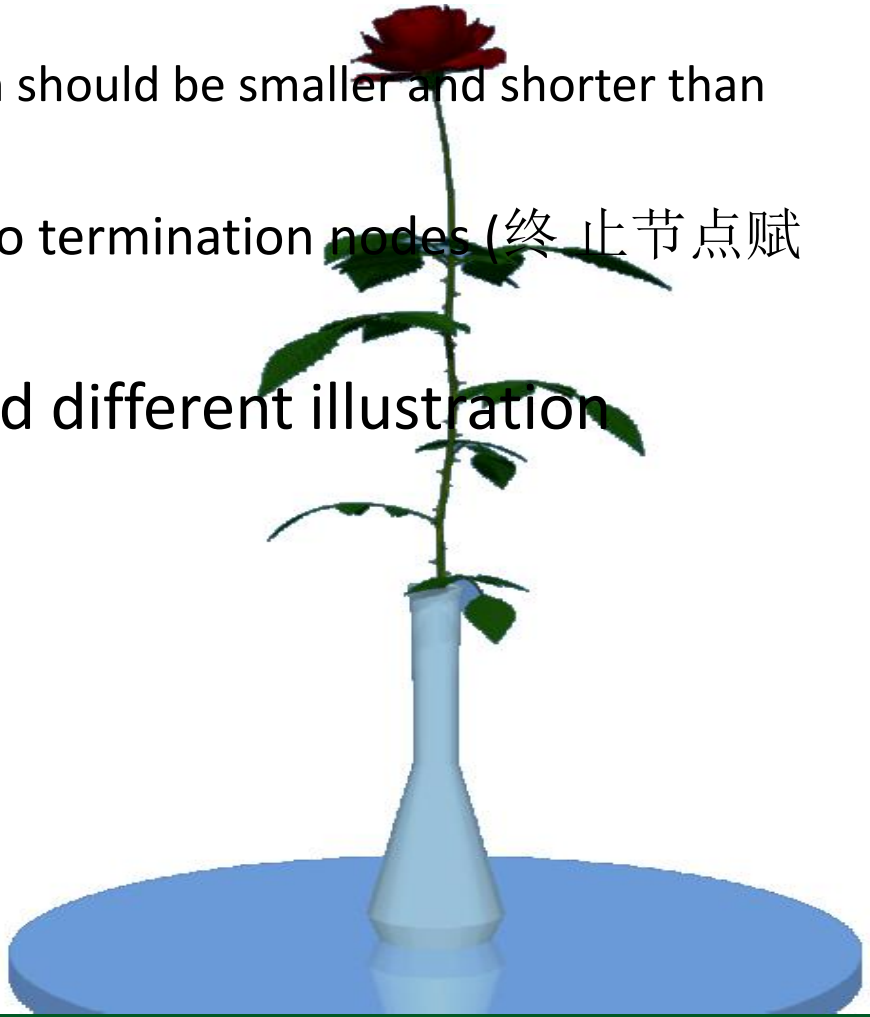
(a) Initial (b) 1st generation (c) 2st generation

L-system: examples for plant modeling

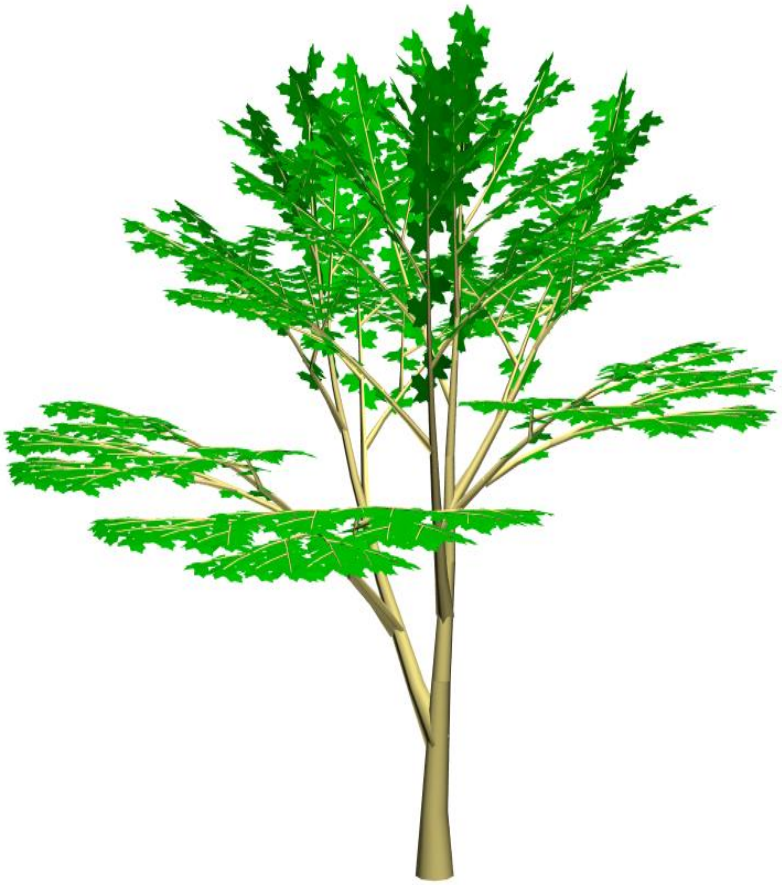


L-system： Plant modeling

- Introduce more control in L-system
  - Illustrate the character of different generation with a different primitive
    - Branches of $(n+1)$-th generation should be smaller and shorter than that of the n-th generation
  - Flowers and leaves are assigned to termination nodes (终止节点赋予树叶和花)
- Use different grammar rules and different illustration



Trees based on L-system: example



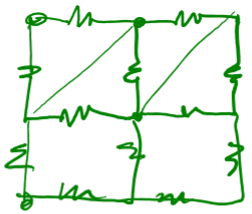
Further reading: The Algorithmic Beauty of Plants

Outline

- Representation of natural scenes
  - Fractal(分形)
  - L-System (L系统*)
  - Particle system (粒子系统*)
  - Cloth simulation

Particle system

- A particle system is a collection of point masses that obeys some physical laws(e.g, gravity, spring behaviors,…)
- Particle systems can be used to simulate all sorts of physical phenomena(fire, fog, smoke, fluid, cloth, and hair etc.)



Components of Particle systems

- Particle sets is changing over time
- Particle movement is controlled by physical methods
- Particles are given life：
  - generation
  - development
  - die

General description of particle system

- Particle system is a dynamic system
- The steps to generate a frame are

1. Create new particles and add them into the system
2. Assign attributes
3. Delete particles who have died (删除超出生命周期者)
4. Move particle according to their motion
5. Render all particles

1 Particle generation

1) Specify the new particle number of each frame

$$Nparts_f = MeanParts_f + Rand() \text{ X } VarianceParts_f$$

MeanPartf ---- 平均值； VarianceParts----变化范围; -1≤rand() ≤ 1

2) Determine the new particle number according to screen size

$$Nparts_f = (MeanParts_{SAf} + Rand() \text{ X } VarianceParts_{SAf}) \text{ X ScreenArea}$$

(按照这个方法新粒子数个数取决于屏幕面积)

2 Particle attributes

- Initial position 初始位置p
- Initial speed 初始速度v
- Initial size 初始大小

  InitialSize = MeanSize + Rand() X VarSize
- Initial color 初始颜色c
- Initial transparency 初始透明度t
- Initial shape 初始形状s
- Lifttime 生命周期
- …

3 Particle Dynamics

- Force field: gravity, wind, tension(压力）
- Viscosity/damp(粘性/阻尼): fluid
- Collision(碰撞)
- Spring(弹力): Springs between neighboring particles (mesh)
- …

Dynamics （动力学）

- Newton law: f = ma
- State change:只有重力作用的时候

  Gravity(加速度): a(t + △t) = g

  Speed(速度): v(t+ △ t) = v(t) + a(t)*△ t

  Position(位置):

  p(t+△t) = p(t) + v(t)*△t + a(t)2*△t/2

4 Particle extinction （粒子死亡）

- For particles, if its lifetime exceeds its life circle, we need to remove it from the system.

5. Particle rendering

- Render all particles as its specified shape
  - Points,
  - Spheres,
  - ellipsoids,
  - ……

A simple example

- Displaying the particles (显示粒子)
- Updating particle positions (更新位置)
- Initialization (初始化)
- Collisions (碰撞检测)
- Flocking(群集)

Particle structure: attributes

```
typedef struct particle {
    int color;          //颜色
    float position[3]; //位置
    float velocity[3]; //速度
    float mass;        //质量
}
Glfloat colors[8][3]={{0.0,0.0,0.0}, {1.0,0.0,0.0}, {0.0,1.0,0.0},
  {0.0,0.0,1.0}, {0.0,1.0,1.0}, {1.0,0.0,1.0}, {1.0,1.0,0.0},
  {1.1,0.1,1.0}};
```

Displaying particle

```
Void myDisplay(){
   int i;
   glClear(GL_COLOR_BUFFER_BIT);
   glBegin(GL_POINTS);
   for (i =0; i <num_particles; i++){
       glColor3fv(colors[particles[i].color]);
       glVertex3fv(particles[i].position);
   }
   glEnd();
}
```

Updating particle positions

```
float last_time, present_time, num_particles;
myIdle(){
   int i,j;   float dt;
   present_time = glutGet(GLUT_ELAPSED_TIME);//mili
   dt = 0,001*(present_time - last_time);
    for (i=0 i<num_particles; i++){
         for (j = 0; j < 3; j++){
              particles[i].position[j]+=dt * particles[i].velocity[j];
              particles[i].velocity[j]+=dt * force(i,j)/particles[i].mass;
          }
          collision(i);
     }
     last_time = present_time;
     glutPostRedisplay();
}
```

Initialization

```
float  num_particles = INITIAL_NUM_PARTICLES; // 自定义
float  speed = INITIAL_SPEED;
Void myinit(){
    int i,j;
    for (i=0 i<num_particles; i++){
        particles[i].mass = 1.0;  //default mass
        particles[i].color = i%8;  //color index
        for (j=0; j<3;j++){
            particles[i].position[j] =2.0*((float)random()/RAND_MAX)-1.0;
      particles[i].velocity[j]=speed*2.0*((float)random()/RAND_MAX)-1.0; }
     }
}
```

Collisions

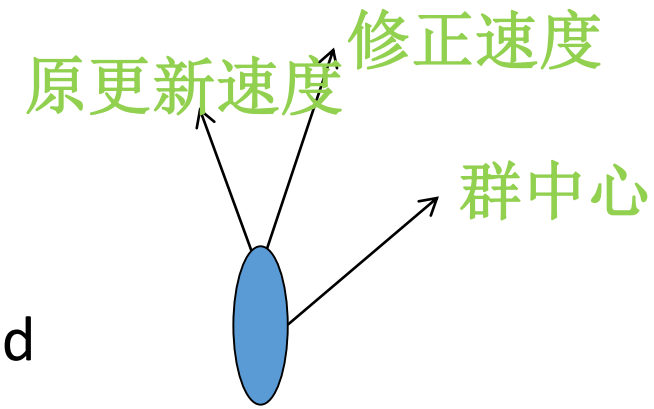
```
float coef;  // coefficient of restitution
Void collision(int n){
    int j;
    for (j=0; j<3;j++){
       if (particles[n].position[j]>=1.0){    //碰壁
          particles[n].velocity[j]=-coef* particles[n].velocity[j];
          particles[n].position[j] =1.0-coef*(particles[n].position[j]-1 );
       }
      if (particles[n].position[j]<=-1.0){    //碰壁
          particles[n].velocity[j]=-coef* particles[n].velocity[j];
          particles[n].position[j] =-1.0-coef*(particles[n].position[j]+1) ; }
    }
}
```

Forces

```
Bool gravity = TRUE;
float forces(int I, int j){
   if (!gravity) return(0.0);
   else if(j==1) return (-1); // y轴是重力方向
   else return(0,0);
}
```

Flocking

```
float cm[3];
Void flock(){  // compute the center of the particles
    for (int k=0; k<3; k++){
        cm[k] = 0;
        for (j=0; j < num_particles; j++){
            cm[k]+=particles[j].position[k];
        cm[k]/=num_particles;
    }
}
// we can now modify the particle speed
//according to the right figure.
```



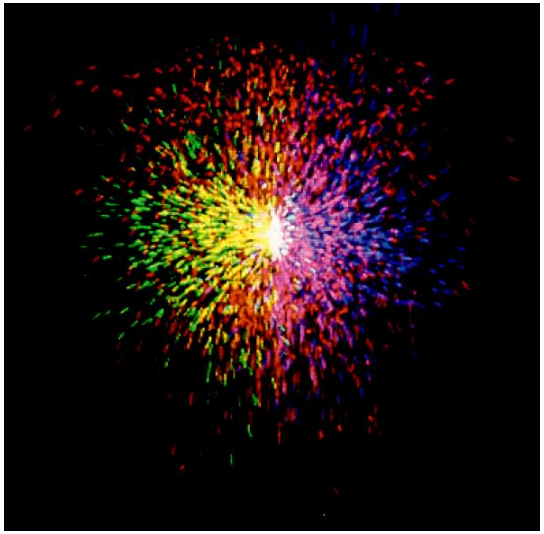
Flame and waterfall: example

粒子系统生成的火焰

粒子系统生成的瀑布

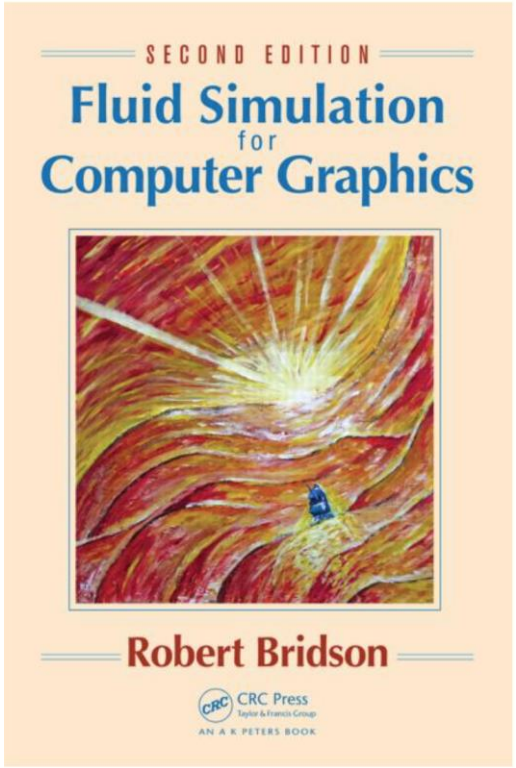
Reference

- WILLIAM T. REEVES. Particle Systemsm： A Technique for Modeling a Class of Fuzzy Objects. ACM TOG, 2(2), 1983:91-108



Books

[1].Bridson R. Fluid simulation for computer graphics[M]. CRC Press, 2015.



Websites

[1].http://www.kesen.realtimerendering.com

[2].http://research.nii.ac.jp/~rand/

[3].http://blog.yiningkarlli.com

[4].http://www.cs.ubc.ca/~rbridson/

[5].http://matthias-mueller-fischer.ch

[6].http://mantaflow.com/doxy/group__Plugins.html

[7].http://rlguy.com/index.html

Websites

[9].http://lgg.epfl.ch/research_physicsbased_animation.php

[10].http://sealab.cs.utah.edu/Courses/CS6967-F08/

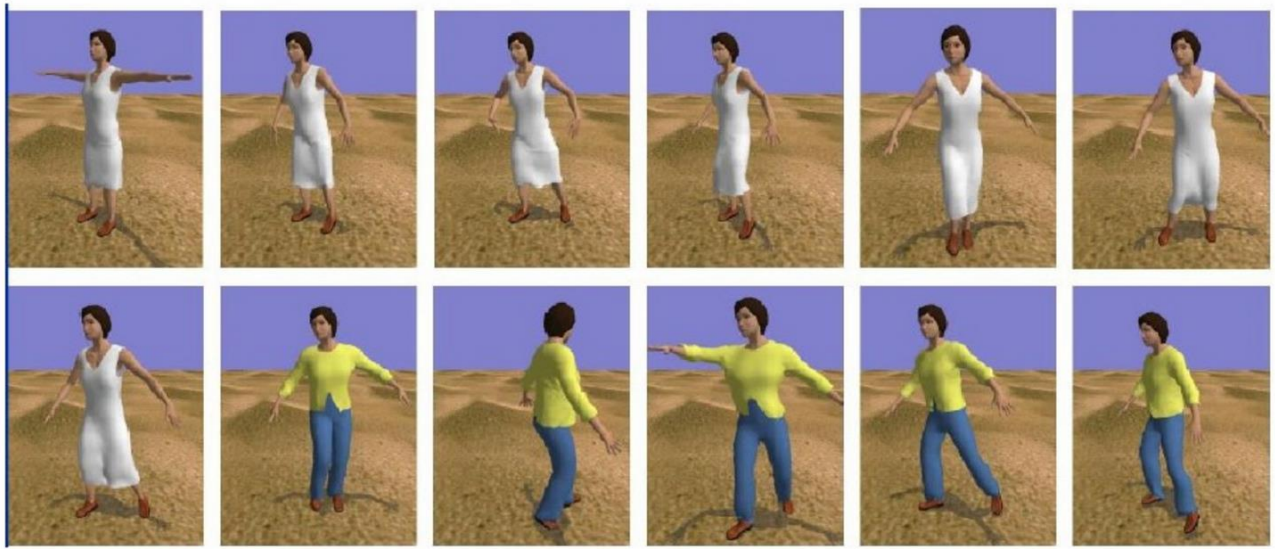
[11]. http://thecodeway.com/blog/?p=83

Outline

- Representation of natural scenes
  - Fractal(分形)
  - L-System (L系统*)
  - Particle system (粒子系统*)
  - Cloth simulation

Cloth Simulation

- **Cloth simulation has been an important topic in computer animation since the early 1980's**
- **It has been extensively researched, and has reached a point where it is basically a solved problem**
- **Today, we will look at a very basic method of cloth simulation.**

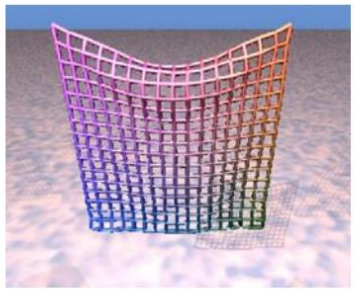


Cloth Simulation

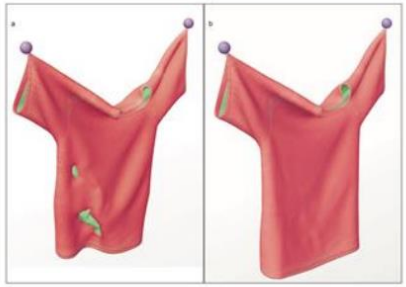
- We will treat the cloth as a system of particles interconnected with spring-dampers
- Each spring-damper connects two particles, and generates a force based on their positions and velocities
- Each particle is also influenced by the force of gravity
- With those three simple forces (gravity, spring, & damping), we form the foundation of the cloth system
- Then, we can add some fancier forces such as aerodynamics, bending resistance, and collisions, plus additional features such as plastic deformation and tearing

Cloth Simulation

- Elastic sheet model



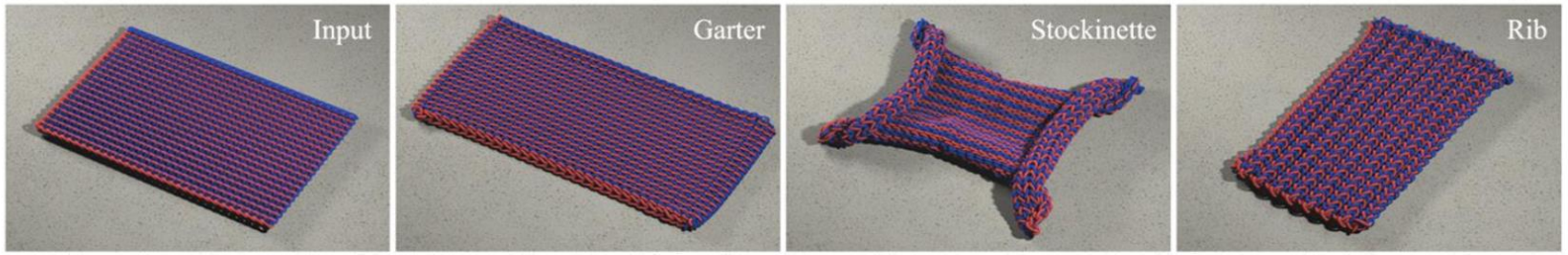
[Provot 1995]



[Baraff et al. 2003]

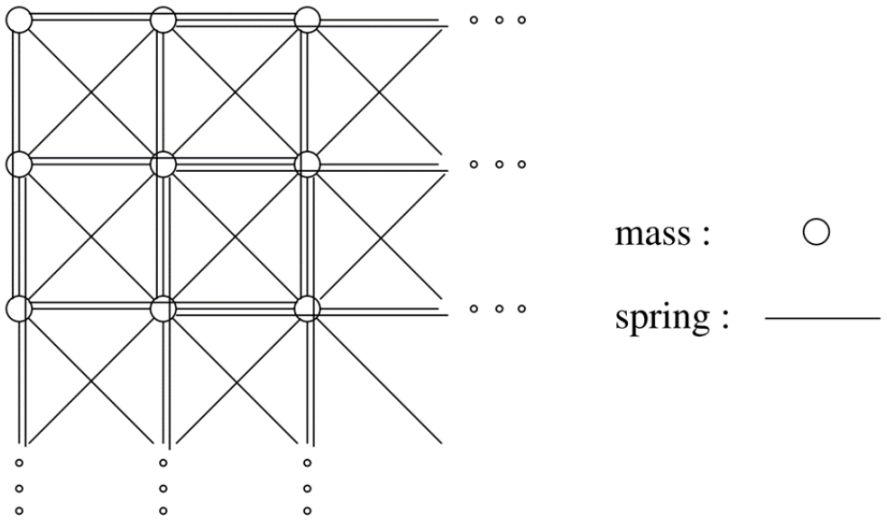
… …

- Yarn-based model

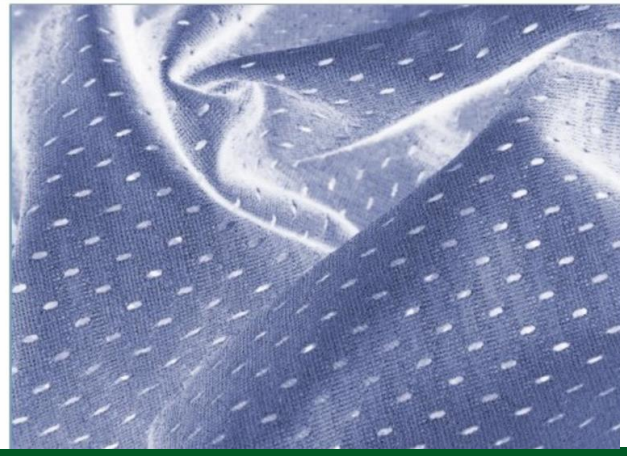


[Kaldor et al. 2008]

Mass-Spring System for Cloth Simulation

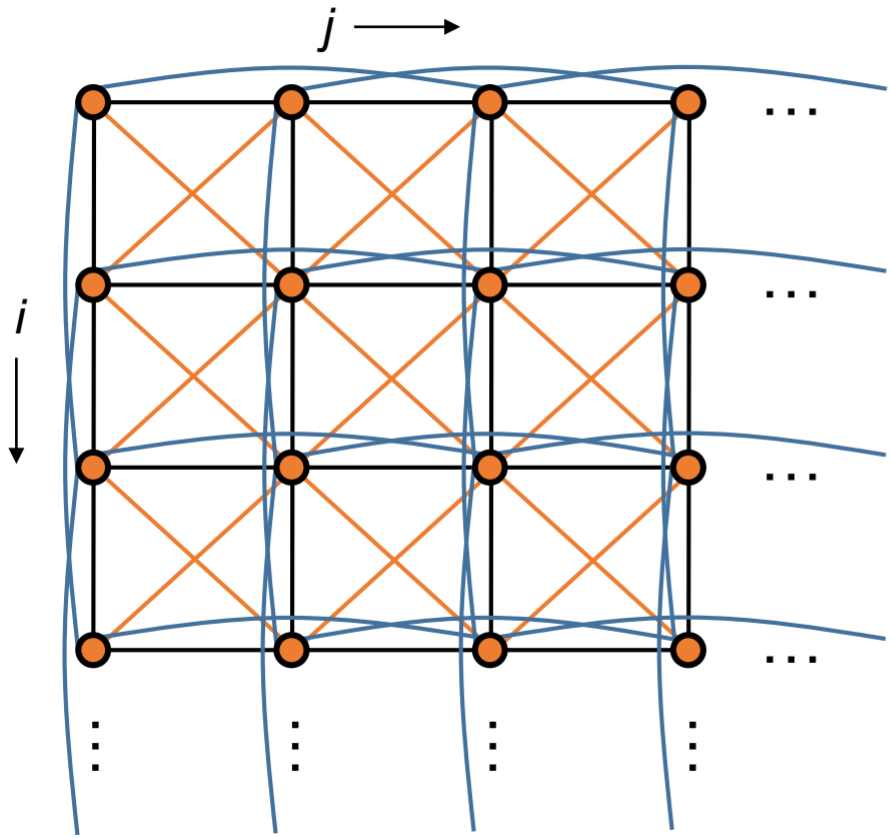


- Cloth = a “web” of particles and springs



Mass-Spring System for Cloth Simulation

- Consider a rectangular cloth with $m \times n$ particles



Types of springs

Structural ——

$[i, j]$—$[i, j + 1]$; $[i, j]$—$[i + 1, j]$

Shear ——

$[i, j]$—$[i + 1, j + 1]$; $[i + 1, j]$—$[i, j + 1]$

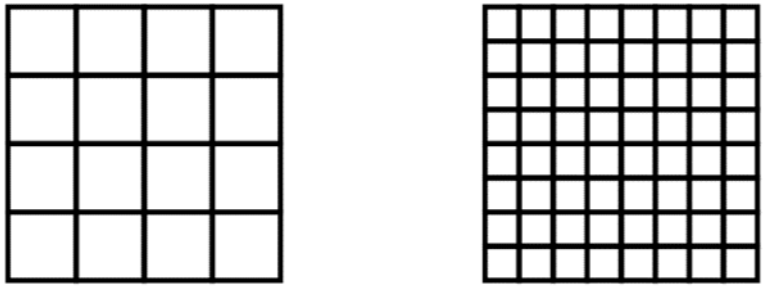
Flexion (bend) ——

$[i, j]$—$[i, j + 2]$; $[i, j]$—$[i + 2, j]$

Discretization

- What happens if we discretize our cloth more or less finely?
- Do we get the same behavior?

- Usually NOT! It takes a lot of effort to design discretization-independent schemes.



Particles

$\mathbf{r}$: *position*

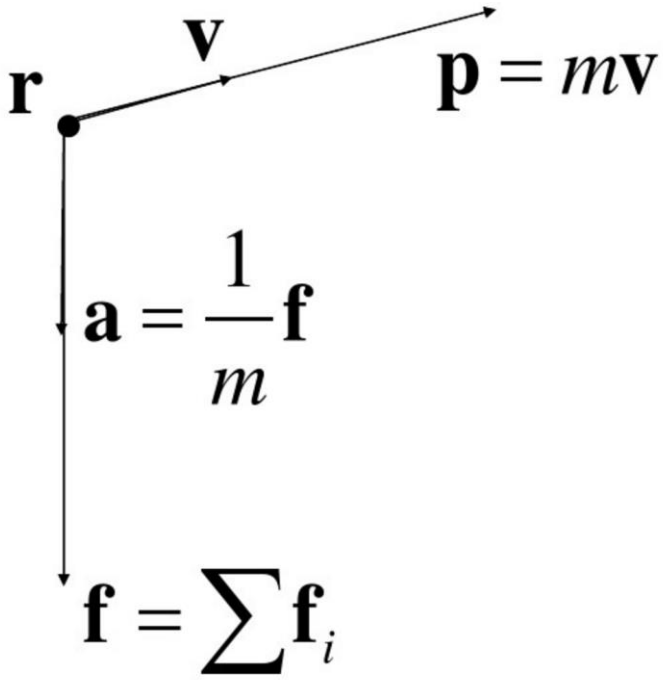
$\mathbf{v}$: *velocity*

$\mathbf{a}$: *acceleration*

m : *mass*

$\mathbf{p}$: *momentum*

$\mathbf{f}$: *force*



Euler Integration

- **Once we've computed all of the forces in the system, we can use Newton's Second Law (f=ma) to compute the acceleration**

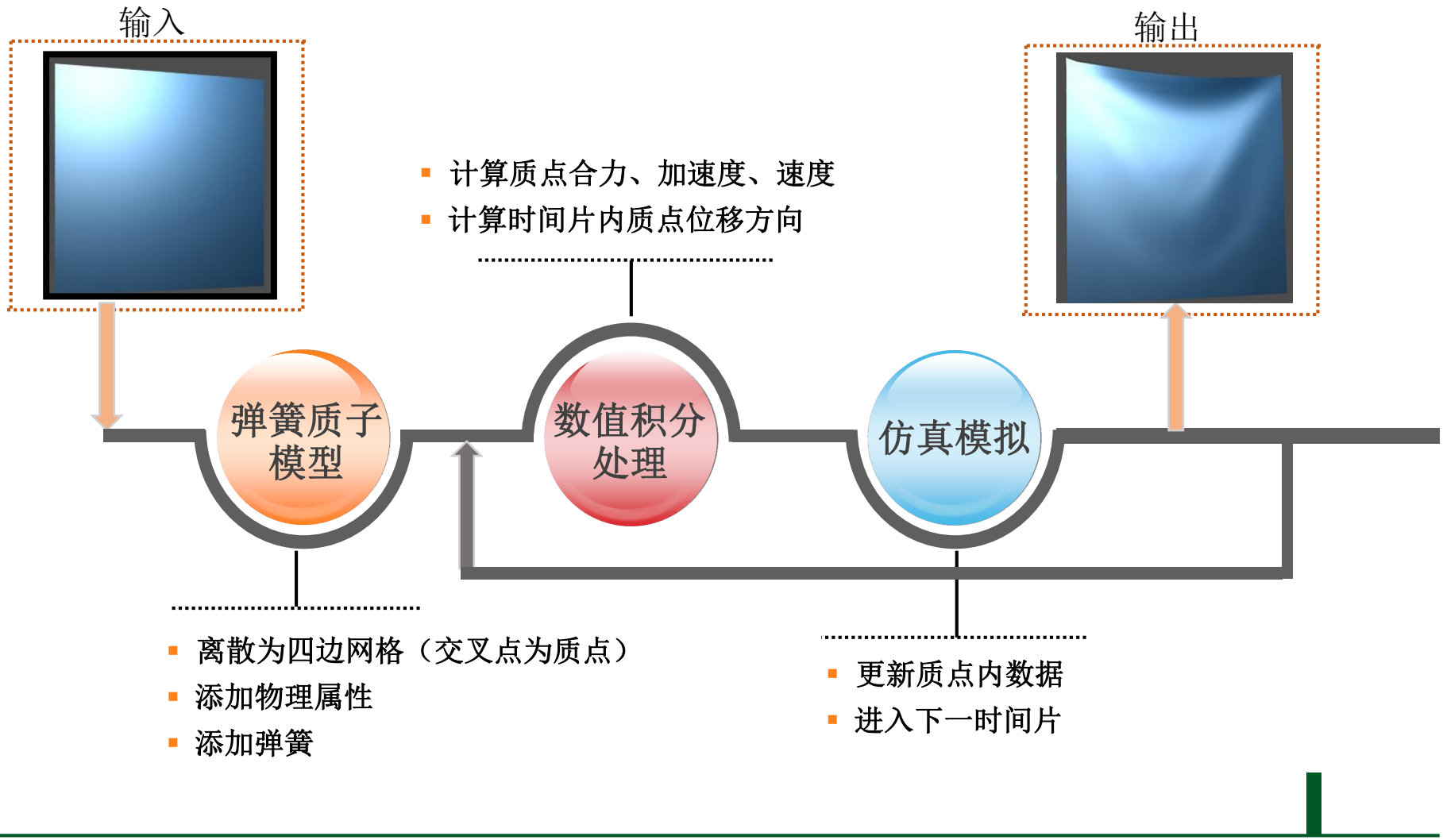
$$\mathbf{a}_n = \frac{1}{m}\mathbf{f}_n$$

- **Then, we use the acceleration to advance the simulation forward by some time step Δt, using the simple Euler integration scheme**

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{a}_n \Delta t$$

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_{n+1} \Delta t$$

算法框架

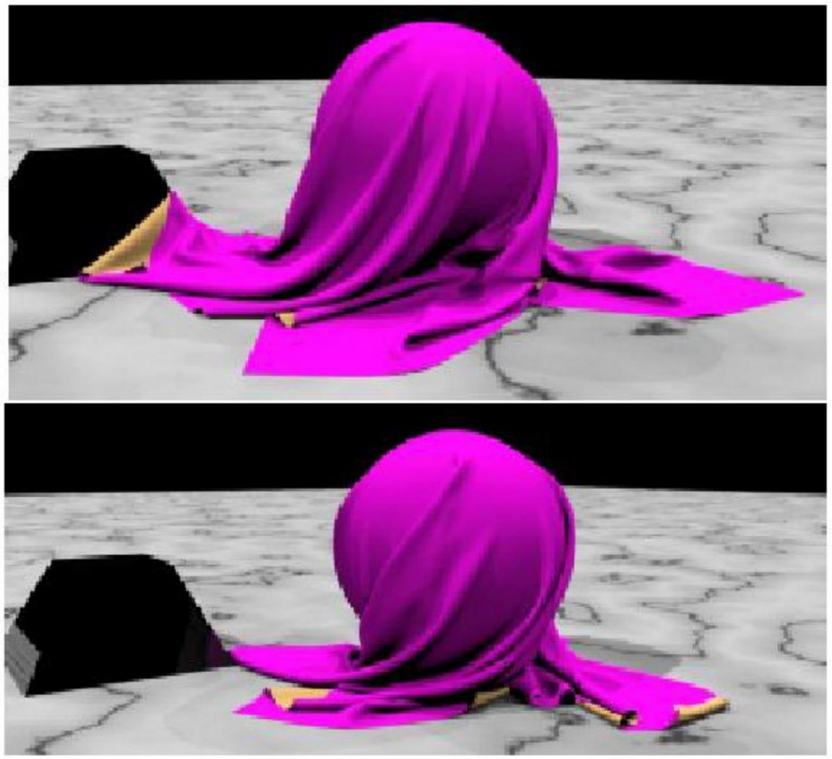

输入
输出
弹簧质子模型
数值积分处理
仿真模拟
计算质点合力、加速度、速度
计算时间片内质点位移方向
离散为四边网格（交叉点为质点）
添加物理属性
添加弹簧
更新质点内数据
进入下一时间片

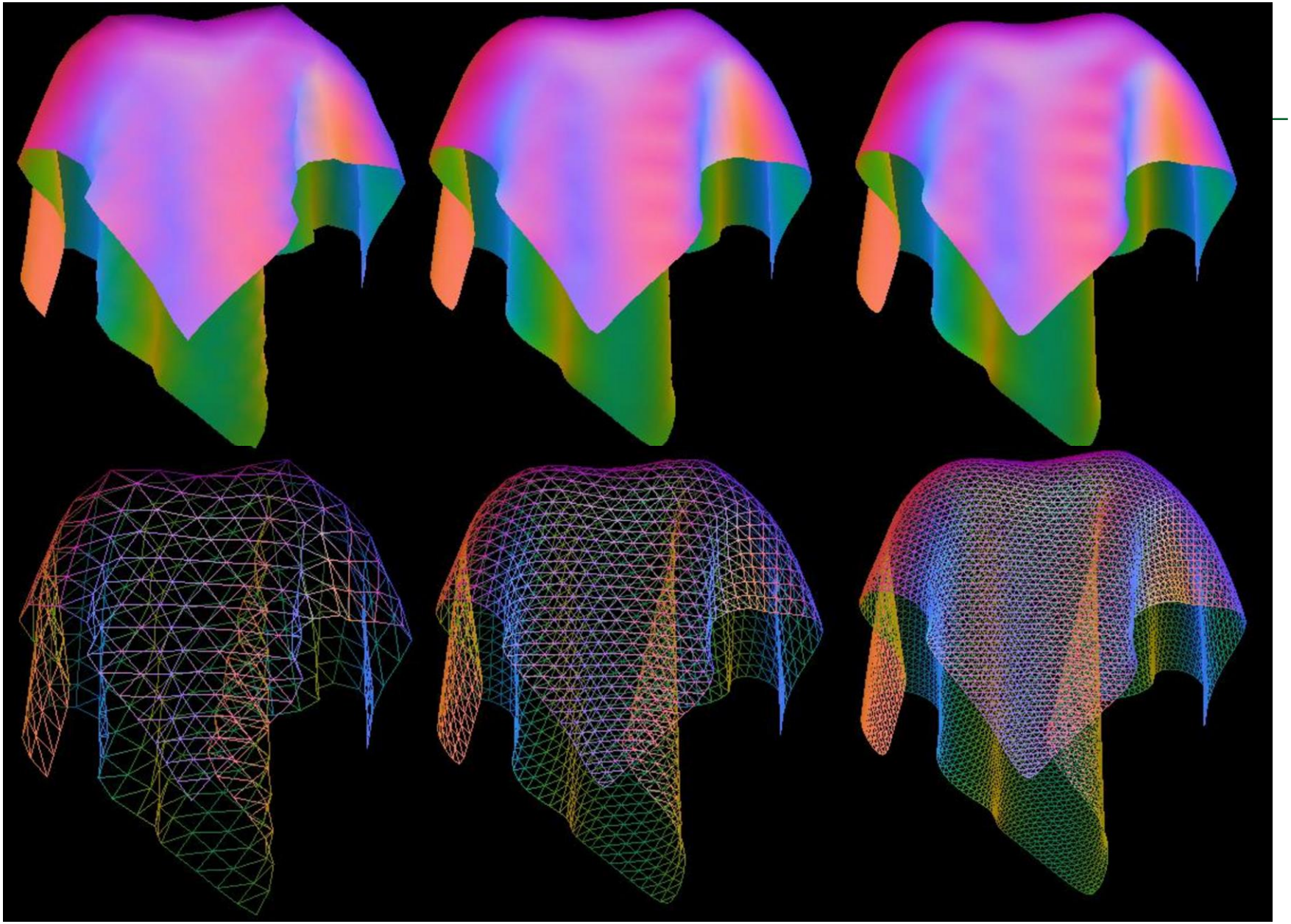
Advanced Cloth Simulation

- Real cloth simulation rarely uses springs
- Instead, forces are generated based on the the deformation of a triangular element
- This way, one can properly account for internal forces within the piece of cloth based on the theory of continuum mechanics
- The basic process is still very similar. Instead of looping through springs computing forces, one loops through the triangles and computes the forces
- Continuum models account for various properties such as elastic deformation, plastic deformation, bending forces, anisotropy, and more

Collision Detection & response

- Cloth colliding with rigid objects is tricky
- Cloth colliding with itself is even trickier
- There have been several published papers on robust cloth collision detection and response methods





20x20 grid with two fixed vertices. Left = No subdivision, Mid = 1 level of subdivision, Right = 2 levels of subdivision