

Current action: Add Particle

6.94 P.SIZE

18.50 P.MASS

FIXED

ADD

1000.00 A.STRENGTH

ATTRACT

REPEL

0.54 ELASTICITY

0.75 GRAVITY

1.00 PREPULSE STRENGTH

0.00 FRICTION

TRAIL

CLEAR

创意编程

课后作业 (06-01) 粒子系统

孔安梨 519430990004



Current action: Add Particle

15.84

PSIZE

100.00

PMASS

FIXED

ADD

1000.00

ARSTRENGTH

ATTRACT

REPEL

0.56

ELASTICITY

38.50

GRAVITY

0.00

PREPELSTRENGTH

0.00

FRICTION

TRAIL

CLEAR

可调

//粒子大小

//粒子质量

//动态/固态

//粒子轨道开关

功能介绍

粒子添加



Current action: Add Particle

33.17 P.SIZE

24.49 P.MASS

FIXED

ADD

1000.00 A.STRENGTH

ATTRACT

REPEL

1.00 ELASTICITY

1.00 GRAVITY

1.00 PREPULSE STRENGTH

0.00 FRICTION

TRAIL

CLEAR

可调

//粒子弹性

//粒子间引力

//粒子间排斥力

//空气阻力

功能介绍

参数模式



Current action: Add Particle

33.17 P.SIZE

24.49 P.MASS

FIXED

ADD

1000.00 A.STRENGTH

ATTRACT

REPEL

1.00 ELASTICITY

1.00 GRAVITY

1.00 PREPULSTRENGTH

0.00 FRICTION

TRAIL

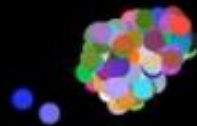
CLEAR

鼠标点击开关

//力的大小

//吸引力按钮

//排斥力按钮



功能介绍

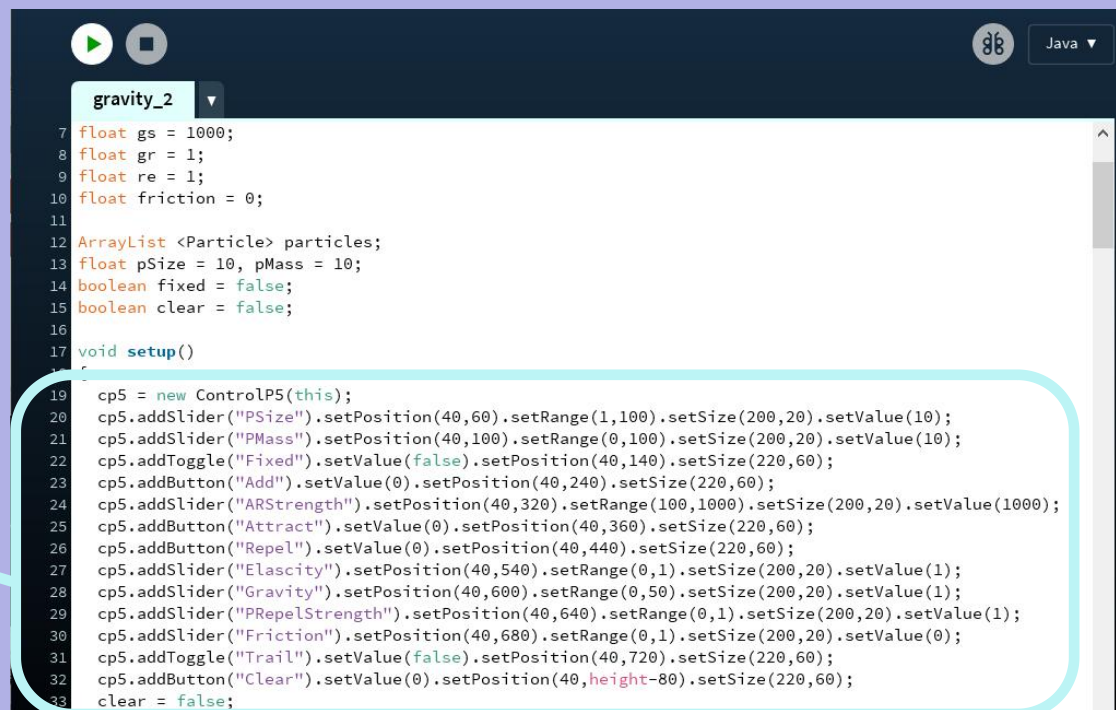
玩耍模式

结构设计说明

各类参数的控制基于由Andreas Schlegel所编写的controlP5库

<http://www.sojamo.de/libraries/controlP5/>

初始化各种slider和button.



```
gravity_2
7 float gs = 1000;
8 float gr = 1;
9 float re = 1;
10 float friction = 0;
11
12 ArrayList <Particle> particles;
13 float pSize = 10, pMass = 10;
14 boolean fixed = false;
15 boolean clear = false;
16
17 void setup()
18 {
19     cp5 = new ControlP5(this);
20     cp5.addSlider("PSize").setPosition(40,60).setRange(1,100).setSize(200,20).setValue(10);
21     cp5.addSlider("PMass").setPosition(40,100).setRange(0,100).setSize(200,20).setValue(10);
22     cp5.addToggle("Fixed").setValue(false).setPosition(40,140).setSize(220,60);
23     cp5.addButton("Add").setValue(0).setPosition(40,240).setSize(220,60);
24     cp5.addSlider("ARStrength").setPosition(40,320).setRange(100,1000).setSize(200,20).setValue(1000);
25     cp5.addButton("Attract").setValue(0).setPosition(40,360).setSize(220,60);
26     cp5.addButton("Repel").setValue(0).setPosition(40,440).setSize(220,60);
27     cp5.addSlider("Elasticity").setPosition(40,540).setRange(0,1).setSize(200,20).setValue(1);
28     cp5.addSlider("Gravity").setPosition(40,600).setRange(0,50).setSize(200,20).setValue(1);
29     cp5.addSlider("PRepelStrength").setPosition(40,640).setRange(0,1).setSize(200,20).setValue(1);
30     cp5.addSlider("Friction").setPosition(40,680).setRange(0,1).setSize(200,20).setValue(0);
31     cp5.addToggle("Trail").setValue(false).setPosition(40,720).setSize(220,60);
32     cp5.addButton("Clear").setValue(0).setPosition(40,height-80).setSize(220,60);
33     clear = false;
```

定义粒子:

定义粒子的质量 (影响粒子对其他粒子的重力); 半径; 颜色;

储存位置、速度和加速度.

储存粒子在三十帧之内的位置路径, 形成好看的移动效果.
构造函数.
设置粒子属性.

每一帧粒子的显示和位置更新
(加速度决定速度 速度决定位置)
加速度在draw里更新.

```
127
128 class Particle
129 {
130     float mass = 1;
131     float rad = 10;
132     color col= color(255);
133     PVector loc;
134     PVector vel = new PVector(0,0);
135     PVector acc = new PVector(0,0);
136     boolean fixed = false;
137     PVector[] trail = new PVector[30];
138     Particle(float x, float y)
139     {
140         loc = new PVector(x,y);
141         for (int i=0; i<30; i++) trail[i] = new PVector(x,y);
142     }
143     Particle copy() {return this;}
144     Particle setMass(float m) { mass = m; return this;}
145     Particle setRad(float r) { rad = r; return this;}
146     Particle setVel(float x, float y) { vel = new PVector(x,y); return this;}
147     Particle setAcc(float x, float y) { acc = new PVector(x,y); return this;}
148     Particle setFixed(boolean f) { fixed = f; return this;}
149     Particle setColor(color c) { col = c; return this;}
150     void next()
151     {
152         display();
153         if (!fixed)
154         {
155             if(loc.x<=300 && vel.x<0) vel.x*=-1;
156             if(loc.x>=width && vel.x>0) vel.x*=-1;
157             if(loc.y<=0 && vel.y<0) vel.y*=-1;
158             if(loc.y>=height && vel.y>0) vel.y*=-1;
159
160             if (trailOn) {
161                 trail[0] = loc;
162                 for(int i = 29; i > 0; i--) trail[i] = trail[i-1].copy();
```

```
11  
12 ArrayList <Particle> particles;  
13 float pSize = 10, pMass = 10;  
14 boolean fixed = false;  
15 boolean clear = false;  
16
```

粒子信息用Arraylist储存.

```
54  
55 //display  
56 for (int i = 0; i < particles.size(); i++)  
57 {  
58     particles.get(i).next();  
59 }
```

对Particles里的每个粒子调用Next().
更新位置以及把它们画出来.

计算每个粒子的加速度:

计算每一个粒子所受另外所有粒子的影响的力所导致的加速度并计算其总和。

得出两个例子之间的向量以及距离

若两个粒子相撞，确保距离有一个最低限度，以保证加速度不会过大。计算其他的排斥力、阻力等等。

根据距离与对方粒子的质量计算重力和加速度的强度，并加入加速度的总和

```
60 //update force
61 for (int i = 0; i < particles.size(); i++)
62 {
63     PVector a = new PVector(0,0); //initialize acceleration
64     Particle pi = particles.get(i);
65     for (int j = 0; j < particles.size(); j++)
66     {
67         if (i == j) continue;
68         Particle pj = particles.get(j);
69         PVector r = pj.loc.copy().sub(pi.loc);
70         float d = r.mag();
71         if (d < pi.rad + pj.rad) //particles are touching/colliding
72         {
73             d = pi.rad + pj.rad; //ensure the gforce is not too large
74             particles.get(i).vel.mult(elascity); //reduce velocity to create 'sticking' effect
75             //repel force between particles to create 'bouncing' effect
76             float mult2 = -1*r.x/ ((particles.get(i).mass+1)*d);
77             PVector aj2 = r.normalize().mult(mult2);
78             a.add(aj2);
79         }
80
81         float mult = gr*particles.get(j).mass / (d*d); //magnitude of gforce
82         PVector aj = r.normalize().mult(mult); //force vector
83         a.add(aj);
84     }
85     particles.get(i).acc = a;
86 }
87
```


玩耍模式时用户鼠标点击所产生的吸引力、排斥力.

```
88  if (mousePressed && mouseX > 300)
89  {
90      if (tool != 0)
91      {
92          float g = gs*gs;
93          if (tool == 1) g *= -1;
94
95          PVector loc = new PVector(mouseX,mouseY);
96
97          for (int i = 0; i < particles.size(); i++)
98          {
99              PVector r = particles.get(i).loc.copy().sub(loc);
100              float d = r.mag();
101              if (d < 5 ) continue;
102              if (d < 20) d = 10;
103              float mult = g / (d*d*particles.get(i).mass);
104              particles.get(i).acc.add(r.normalize().mult(mult));
105          }
106      }
107  }
```