

Oct 04, 11 9:40

CollisionSystem.java

Page 1/4

```

1  /*****
2  *  COS226 - Algorithms 4 booksite, Section 6.1 applications
3  *
4  *  Compilation:  javac CollisionSystem.java
5  *  Execution:    java CollisionSystem N          (N random particles)
6  *                java CollisionSystem < input.txt (from a file)
7  *
8  *  Creates N random particles and simulates their motion according
9  *  to the laws of elastic collisions.
10 *
11 *  Week 4 - handout questions
12 *
13 *  1 - Where is "pq" declared?
14 *
15 *  2 - Where is "pq" created?
16 *
17 *  3 - Look at the method predict(). It inserts on "pq". How many
18 *      different events are created in one call? What different kinds are they?
19 *
20 *  4 - Now look at the method simulate(). How many events are initially
21 *      put on "pq" (before the while loop begins)?
22 *
23 *  5 - When will simulate() end?
24 *
25 *  6 - One iteration of the while loop in simulate() processes how many events?
26 *
27 *  7 - When will an event taken off "pq" cause no collision to happen?
28 *
29 *  *****/
30
31 import java.awt.Color;
32
33 public class CollisionSystem {
34
35     private MinPQ<Event> pq;          // the priority queue
36     private double t = 0.0;          // simulation clock time
37     private double Hz = 0.5;         // number of redraw events per clock tick
38     private Particle[] particles;    // the array of particles
39
40     // create a new collision system with the given set of particles
41     public CollisionSystem(Particle[] particles) {
42         this.particles = particles;
43     }
44
45     // updates priority queue with all new events for particle a
46     private void predict(Particle a, double limit) {
47         if (a == null) return;
48
49         // particle-particle collisions
50         for (int i = 0; i < particles.length; i++) {
51             double dt = a.timeToHit(particles[i]);
52             if (t + dt <= limit)
53                 pq.insert(new Event(t + dt, a, particles[i]));
54         }
55
56         // particle-wall collisions
57         double dtX = a.timeToHitVerticalWall();
58         double dtY = a.timeToHitHorizontalWall();
59         if (t + dtX <= limit) pq.insert(new Event(t + dtX, a, null));
60         if (t + dtY <= limit) pq.insert(new Event(t + dtY, null, a));
61     }
62
63
64
65
66
67
68
69
70
71
72
73

```

Oct 04, 11 9:40

CollisionSystem.java

Page 2/4

```

74 // redraw all particles
75 private void redraw(double limit) {
76     StdDraw.clear();
77     for (int i = 0; i < particles.length; i++) {
78         particles[i].draw();
79     }
80     StdDraw.show(20);
81     if (t < limit) {
82         pq.insert(new Event(t + 1.0 / Hz, null, null));
83     }
84 }
85
86
87
88 /*****
89 *  Event based simulation for limit seconds
90 *****/
91 public void simulate(double limit) {
92
93     // initialize PQ with collision events and redraw event
94     pq = new MinPQ<Event>();
95     for (int i = 0; i < particles.length; i++) {
96         predict(particles[i], limit);
97     }
98     pq.insert(new Event(0, null, null)); // redraw event
99
100    // the main event-driven simulation loop
101    while (!pq.isEmpty()) {
102
103        // get impending event, discard if invalidated
104        Event e = pq.delMin();
105        if (!e.isValid()) continue;
106        Particle a = e.a;
107        Particle b = e.b;
108
109        // physical collision, update positions, and then simulation clock
110        for (int i = 0; i < particles.length; i++)
111            particles[i].move(e.time - t);
112        t = e.time;
113
114        // process event
115        if (a != null && b != null)
116            a.bounceOff(b); // particle-particle collision
117        else if (a != null && b == null)
118            a.bounceOffVerticalWall(); // particle-wall collision
119        else if (a == null && b != null)
120            b.bounceOffHorizontalWall(); // particle-wall collision
121        else if (a == null && b == null)
122            redraw(limit); // redraw event
123
124        // update the priority queue with new collisions involving a or b
125        predict(a, limit);
126        predict(b, limit);
127    }
128 }
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146

```

Oct 04, 11 9:40

CollisionSystem.java

Page 3/4

```

147 /*****
148 * An event during a particle collision simulation. Each event contains
149 * the time at which it will occur (assuming no supervening actions)
150 * and the particles a and b involved.
151 *
152 * - a and b both null:    redraw event
153 * - a null, b not null:  collision with vertical wall
154 * - a not null, b null:  collision with horizontal wall
155 * - a and b both not null: binary collision between a and b
156 *
157 *****/
158 private class Event implements Comparable<Event> {
159     private final double time; // when event is scheduled to occur
160     private final Particle a, b; // particles involved in event,
161                                 // possibly null
162     private final int countA, countB; // collision counts at event creation
163
164
165     // create a new event to occur at time t involving a and b
166     public Event(double t, Particle a, Particle b) {
167         this.time = t;
168         this.a = a;
169         this.b = b;
170         if (a != null) countA = a.count();
171         else countA = -1;
172         if (b != null) countB = b.count();
173         else countB = -1;
174     }
175
176     // compare times when two events will occur
177     public int compareTo(Event that) {
178         if (this.time < that.time) return -1;
179         else if (this.time > that.time) return +1;
180         else return 0;
181     }
182
183     // has any collision occurred between when event was created and now?
184     public boolean isValid() {
185         if (a != null && a.count() != countA) return false;
186         if (b != null && b.count() != countB) return false;
187         return true;
188     }
189 }
190 }
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219

```

Tuesday October 04, 2011

Oct 04, 11 9:40

CollisionSystem.java

Page 4/4

```

220
221 /*****
222 * Sample client
223 *****/
224 public static void main(String[] args) {
225
226     // remove the border
227     StdDraw.setXscale(1.0/22.0, 21.0/22.0);
228     StdDraw.setYscale(1.0/22.0, 21.0/22.0);
229
230     // turn on animation mode
231     StdDraw.show(0);
232
233     // the array of particles
234     Particle[] particles;
235
236     // create N random particles
237     if (args.length == 1) {
238         int N = Integer.parseInt(args[0]);
239         particles = new Particle[N];
240         for (int i = 0; i < N; i++) particles[i] = new Particle();
241     }
242
243     // or read from standard input
244     else {
245         int N = StdIn.readInt();
246         particles = new Particle[N];
247         for (int i = 0; i < N; i++) {
248             double rx = StdIn.readDouble();
249             double ry = StdIn.readDouble();
250             double vx = StdIn.readDouble();
251             double vy = StdIn.readDouble();
252             double radius = StdIn.readDouble();
253             double mass = StdIn.readDouble();
254             int r = StdIn.readInt();
255             int g = StdIn.readInt();
256             int b = StdIn.readInt();
257             Color color = new Color(r, g, b);
258             particles[i] = new Particle(rx, ry, vx, vy, radius,
259                                     mass, color);
260         }
261     }
262
263     // create collision system and simulate
264     CollisionSystem system = new CollisionSystem(particles);
265     system.simulate(10000);
266 }
267
268 }

```

CollisionSystem.java

2/2