

```

1 //Note: file contains compressed program: cat+mouse.cpp & positions.cpp & positions.h
2
3 // positions.cpp begin
4 #include <iostream>
5 #include <cmath>
6 using namespace std;
7
8 class Position {
9 public:
10 Position() {
11     myRadius = 1.0;
12     myAngleInRadians = 0.0;
13 }
14
15 Position(float r) {
16     myRadius = r;
17     myAngleInRadians = 0;
18 }
19
20 Position(float r, float thetaInRadians) {
21     myRadius = r;
22     myAngleInRadians = thetaInRadians;
23 }
24
25 void SetAbsolutePosition(float r, float thetaInRadians) {
26     myRadius = r;
27     myAngleInRadians = thetaInRadians;
28 }
29
30 void IncrementPosition(float rChange, float angularDistChange) {
31     if (rChange != 0 && angularDistChange != 0){
32         return;
33     }
34     myRadius += rChange;
35
36     if (this->IsAtStatue()) {
37         myRadius = 1.0;
38     }
39
40     float angleChange = angularDistChange/myRadius;
41     myAngleInRadians += angleChange;
42 }
43
44 void Print() const {
45     cout << "Radius: " << this->myRadius << endl;
46     cout << "Angle: " << this->myAngleInRadians << endl;
47 }
48
49 bool Sees(Position mouse) const {
50     if (this->myRadius * cos(this->myAngleInRadians - mouse.myAngleInRadians) >= 1.0) {
51         return true;
52     }
53     return false;
54 }
55
56 bool IsAtStatue() const {
57     if (this->myRadius <= 1.0) {
58         return true;
59     }
60     return false;
61 }
62
63 bool IsBetween(Position pos1, Position pos2) const {
64     float mouseAng = myAngleInRadians;
65     if (cos(mouseAng - pos1.myAngleInRadians) >= cos(pos2.myAngleInRadians -
66         pos1.myAngleInRadians)
67         && cos(pos2.myAngleInRadians - mouseAng) >= cos(pos2.myAngleInRadians -
68         pos1.myAngleInRadians)) {
69         return true;

```

```

68     }
69     return false;
70 }
71
72 void tests() {
73
74     Position test1;
75     Position test2 (7.6);
76     Position test3 (3.20, 6.309);
77     cout << "Testing constructor, expected (1,0)\n";
78     test1.Print();
79     cout << "\nTesting overloaded constructor1, expected (rand,7.6)\n";
80     test2.Print();
81     cout << "\nTesting overloaded constructor2, expected (3.20,6.309)\n";
82     test3.Print();
83
84     cout << "\nTesting SetAbsolutePosition, expected (2.22, 4.67)\n";
85     test3.SetAbsolutePosition(2.22, 4.67);
86     test3.Print();
87
88     cout << "\nTesting IncrementPosition, expected (1,-4)\n";
89     test1.IncrementPosition(0, -4);
90     test1.Print();
91
92     cout << "\nexpected (3.3,-4)\n";
93     test1.IncrementPosition(2.3, 0);
94     test1.Print();
95
96     cout << "\nTesting Sees \n";
97     for (int i = 0; i < 5; i++) {
98         float a,b,c;
99         a = rand();
100        b = rand();
101        c = rand();
102        Position testCat (a,b);
103        Position testMouse (c);
104
105        float expected = (a*cos(b - c) >= 1.0);
106        cout << "expected " << expected << endl;
107        cout << testCat.Sees(testMouse) << endl;
108    }
109
110    cout << "\nTesting IsBetween \n";
111    Position testCat (1.0,1.0);
112    cout << "expected 0\n";
113    cout << testCat.IsBetween(0.0,0.9);
114    cout << "\nexpected 1\n";
115    cout << testCat.IsBetween(0.9,1.1);
116    cout << "\nexpected 1\n";
117    cout << testCat.IsBetween(6.78,7.58);
118
119    cout << "\n\nTesting IsAtStatue\n";
120    cout << "expected 1\n";
121    cout << testCat.IsAtStatue();
122    testCat.IncrementPosition(4.3, 0);
123    cout << "\nexpected 0\n";
124    cout << testCat.IsAtStatue();
125 }
126 private:
127     float myRadius;
128
129     float myAngleInRadians;
130 };
131
132 // positions.cpp end
133
134 // cat+mouse.cpp begin
135 // #include positions.h
136 // #include <iostream>

```

```

137 // #include <cmath>
138 // using namespace std;
139 void GetPositions(Position& cat, Position& mouse)
140 {
141     float catR;
142     float catAng;
143     float mouseAng;
144
145     cout << "Please enter cat radius.\n";
146     cin >> catR;
147
148     cout << "Please enter cat angle (degrees)." << endl;
149     cin >> catAng;
150
151     cout << "Please enter mouse angle (degrees)." << endl;
152     cin >> mouseAng;
153
154     catAng = catAng*M_PI/180;
155     mouseAng = mouseAng*M_PI/180;
156
157     cat = Position(catR,catAng);
158     mouse = Position(1,mouseAng);
159 }
160
161 /**
162  * You define the RunChase function.
163  * Given initialized cat and mouse positions,
164  * it should simulate the cat chasing the mouse, printing the
165  * result of each movement of cat and mouse. Either the cat will
166  * catch the mouse, or 30 time units will go by and the cat will
167  * give up.
168  */
169 void RunChase(Position& cat, Position& mouse)
170 {
171     Position newCatPosition = cat;
172     Position oldCatPosition = cat;
173
174     int counter = 0;
175     int const maxTime = 30;
176     bool caught = false;
177
178     if (mouse.IsBetween(cat,cat)) {
179         cout << "\nMouse caught immediately\n";
180         caught = true;
181     }
182
183     while (counter < maxTime && caught == false) {
184
185         if (oldCatPosition.Sees(mouse)) {
186             newCatPosition.IncrementPosition(-1.0, 0.0);
187         } else {
188             newCatPosition.IncrementPosition(0.0, 1.25);
189         }
190
191         counter++;
192
193         if (mouse.IsBetween(oldCatPosition, newCatPosition)
194             && newCatPosition.IsAtStatue()) {
195             caught = true;
196             cout << "\nMouse caught at time: " << counter << endl;
197         } else {
198             mouse.IncrementPosition(0.0, 1);
199         }
200         cout << endl << counter;
201         cout << "\nMouse\n";
202         mouse.Print();
203         cout << "\nCat\n";
204         newCatPosition.Print();
205         oldCatPosition = newCatPosition;

```

```
206     }
207
208     if (!caught) {
209         cout << "Cat wandered off.\n";
210     }
211 }
212
213 //
214
215 int main()
216 {
217     Position cat, mouse;
218     GetPositions(cat, mouse);
219     RunChase(cat, mouse);
220     return 0;
221 }
```