**[118] Programming Lab 4**

Haoyu Wang

Monday

October 09, 2017

**Lab 4 Code**

1. // Programming Lab 04 - 10/09/2017
2. // written by Haoyu Wang
3. #include "library.h"

6. **const** **double** PI = acos(-1.0);
8. **void** boom(**double** posx, **double** posy, **int** length)
9. {
11. **int** y = length;
12. **int** z = length \* 2;
14. start\_shape(); // SHAPE
15. move\_to(posx,posy);
16. note\_position();  // Shape
17. set\_heading\_degrees(0);
18. turn\_left\_by\_degrees(90); // changing the direction to to 270
19. draw\_distance(y);
20. note\_position();  // Shape
21. set\_heading\_degrees(-60);
22. draw\_distance(y);
23. note\_position();  // Shape
24. set\_heading\_degrees(90);
25. draw\_distance(3/2.0 \*y);
26. note\_position();  // Shape
27. set\_heading\_degrees(-30);
28. draw\_distance(y);
29. note\_position();  // Shape
30. set\_heading\_degrees(120);
31. draw\_distance(y);
32. note\_position();  // Shape
33. set\_heading\_degrees(20);
34. draw\_distance(y);
35. note\_position();  // Shape
36. set\_heading\_degrees(160);
37. draw\_distance(y);
38. note\_position();  // Shape
39. set\_heading\_degrees(30);
40. draw\_distance(y);
41. note\_position();  // Shape
42. set\_heading\_degrees(140);
43. draw\_distance(y);
44. note\_position();  // Shape
45. set\_heading\_degrees(60);
46. draw\_distance(y);
47. note\_position();  // Shape
48. set\_heading\_degrees(180);
49. draw\_distance(2/3.0 \* y);
50. note\_position();  // Shape
51. set\_heading\_degrees(60);
52. draw\_distance(y);
53. note\_position();  // Shape
54. set\_heading\_degrees(200);
55. draw\_distance(y);
56. note\_position();  // Shape
57. set\_heading\_degrees(110);
58. draw\_distance(y);
59. note\_position();  // Shape
60. set\_heading\_degrees(260);
61. draw\_distance(2.5/2.0 \*y);
62. note\_position();  // Shape
63. set\_heading\_degrees(130);
64. draw\_distance(3/2.0 \*y);
65. note\_position();  // Shape
66. set\_heading\_degrees(290);
67. draw\_distance(y);
68. note\_position();  // Shape
69. set\_heading\_degrees(180);
70. draw\_distance(y);
71. note\_position();  // Shape
72. set\_heading\_degrees(-60);
73. draw\_distance(y);
74. note\_position();  // Shape
75. set\_heading\_degrees(200);
76. draw\_distance(y);
77. note\_position();  // Shape
78. set\_heading\_degrees(-30);
79. draw\_distance(y);
80. note\_position();  // Shape
82. set\_heading\_degrees(220);
83. draw\_distance(y);
84. note\_position();  // Shape
85. set\_heading\_degrees(-30);
86. draw\_distance(2.5/2 \* y);
87. note\_position();  // Shape
89. set\_heading\_degrees(250);
90. draw\_distance(y);
91. note\_position();  // Shape
92. set\_heading\_degrees(20);
93. draw\_distance( y);
94. note\_position();  // Shape
96. set\_heading\_degrees(260);
97. draw\_distance(y);
98. note\_position();  // Shape
99. set\_heading\_degrees(30);
100. draw\_distance(0.75 \* y);
101. note\_position();  // Shape
102. set\_pen\_color(color::orange);
103. fill\_shape();
105. //Draw Edges
106. move\_to(posx,posy);
107. set\_pen\_color(color::red);
108. set\_pen\_width(2);
109. set\_heading\_degrees(0);
110. turn\_left\_by\_degrees(90);
111. move\_distance(y);
112. set\_heading\_degrees(-60);
113. draw\_distance(y);
114. set\_heading\_degrees(90);
115. draw\_distance(3/2.0 \*y);
116. set\_heading\_degrees(-30);
117. draw\_distance(y);
118. set\_heading\_degrees(120);
119. draw\_distance(y);
120. set\_heading\_degrees(20);
121. draw\_distance(y);
122. set\_heading\_degrees(160);
123. draw\_distance(y);
124. set\_heading\_degrees(30);
125. draw\_distance(y);
126. set\_heading\_degrees(140);
127. draw\_distance(y);
128. set\_heading\_degrees(60);
129. draw\_distance(y);
130. set\_heading\_degrees(180);
131. draw\_distance(2/3.0 \* y);
132. set\_heading\_degrees(60);
133. draw\_distance(y);
134. set\_heading\_degrees(200);
135. draw\_distance(y);
136. set\_heading\_degrees(110);
137. draw\_distance(y);
138. set\_heading\_degrees(260);
139. draw\_distance(2.5/2.0 \*y);
140. set\_heading\_degrees(130);
141. draw\_distance(3/2.0 \*y);
142. set\_heading\_degrees(290);
143. draw\_distance(y);
144. set\_heading\_degrees(180);
145. draw\_distance(y);
146. set\_heading\_degrees(-60);
147. draw\_distance(y);
148. set\_heading\_degrees(200);
149. draw\_distance(y);
150. set\_heading\_degrees(-30);
151. draw\_distance(y);
152. set\_heading\_degrees(220);
153. draw\_distance(y);
154. set\_heading\_degrees(-30);
155. draw\_distance(2.5/2 \* y);
156. set\_heading\_degrees(250);
157. draw\_distance(y);
158. set\_heading\_degrees(20);
159. draw\_distance( y);
160. set\_heading\_degrees(260);
161. draw\_distance(y);
162. set\_heading\_degrees(30);
163. draw\_distance(0.75 \* y);
165. set\_pen\_color(color::red);
166. move\_to(posx+(length),posy);
167. set\_font\_size(length+ length/4);
168. write\_string("BOOM!",direction::center);
169. set\_font\_size(0);
171. set\_pen\_width(2);
172. set\_pen\_color(color::black);
173. set\_heading\_degrees(0);
175. }
177. **void** cannonWheel(**const** **double** r, **const** **double** N, **int** shaped, **const** **double** size)
178. {
179. **if**(N<=0)
180. {
181. set\_pen\_color(color::black);
182. set\_pen\_width(5);
183. draw\_point();
184. set\_pen\_width(1);
185. }
186. **else**
187. {
188. set\_pen\_width(size);
189. set\_pen\_color(color::black);
190. move\_distance(r);
191. draw\_point();
192. **if**(shaped==1)
193. note\_position();
194. move\_distance(-r);
195. turn\_right\_by\_degrees(1);
196. **if**(shaped==1)
197. cannonWheel(r,N-1,1,size);
198. **else**
199. cannonWheel(r,N-1,0,size);
200. }
202. }
204. **double** cannonBody(**const** **double** r, **const** **double** angle)
205. {
206. set\_pen\_width(2);
207. **const** **double** L1 = 1.4\*r;
208. **const** **double** L2 = 4.0\*r;
209. **const** **double** L3 = L1+L2;
210. **const** **double** W1 = 2.0\*r;
211. **const** **double** W2 = 1.4\*r;
213. **const** **double** b = ( asin((W1-W2)/2/(L1+L2)) ) \* 180 / PI;
215. **const** **double** cX = get\_x\_position(); // the position of the center of wheel C
216. **const** **double** cY = get\_y\_position();
217. **const** **double** gY = get\_y\_position()+r;
219. set\_heading\_degrees(180+90-angle);
221. move\_distance(L1);
222. **const** **double** pX = get\_x\_position();
223. **const** **double** pY = get\_y\_position();
224. move\_distance(-L1);
226. **if** (pY > gY) // to make sure P can't be bellow ground level
227. {
228. print("ERROR! At angle : ");
229. print( angle-b );
230. print(" | gY : ");
231. print( gY );
232. print(" | pY : ");
233. print( pY );
234. print("\t.. trying a lower angle.");
235. new\_line();
236. **return** cannonBody(r,angle-1);
237. }
239. set\_pen\_color(color::white);
240. draw\_distance(r);
241. set\_pen\_color(color::black);
242. draw\_distance(L1-r);
244. start\_shape();
245. note\_position(); //1
247. turn\_right\_by\_degrees(93);
248. draw\_distance(W1/2.0);
250. **const** **double** w1X = get\_x\_position(); // getting the position of the center of W1
251. **const** **double** w1Y = get\_y\_position();
253. draw\_distance(W1/2.0);
255. note\_position(); //2
257. turn\_right\_by\_degrees(93);
258. draw\_distance(L3);
260. note\_position(); //3
262. turn\_right\_by\_degrees(87);
263. draw\_distance(W2/2.0);
265. **const** **double** eX = get\_x\_position(); // getting the position of the center of W2
266. **const** **double** eY = get\_y\_position();
268. draw\_distance(W2/2.0);
270. **const** **double** posX = get\_x\_position();
271. **const** **double** posY = get\_y\_position();
273. note\_position(); //4
274. set\_pen\_color(color::light\_grey);
275. fill\_shape();
277. //Coloring Cannon Head
278. set\_pen\_color(color::black);
279. turn\_right\_by\_degrees(87);
281. start\_shape();
282. note\_position(); //1
283. move\_distance(L3/10);
284. **const** **double** p1bX = get\_x\_position();
285. **const** **double** p1bY = get\_y\_position();
286. move\_distance(L3/10);
287. **const** **double** p2bX = get\_x\_position();
288. **const** **double** p2bY = get\_y\_position();
289. move\_distance(L3/10);
290. **const** **double** p3bX = get\_x\_position();
291. **const** **double** p3bY = get\_y\_position();
292. note\_position(); //2
294. move\_distance( L3 - 3\*(L3/10) );
295. turn\_right\_by\_degrees(93);
296. move\_distance(W1);
297. turn\_right\_by\_degrees(93);
299. move\_distance( L3 - 3\*(L3/10) );
300. note\_position(); //3
301. **const** **double** p3tX = get\_x\_position();
302. **const** **double** p3tY = get\_y\_position();
303. move\_distance(L3/10);
304. **const** **double** p2tX = get\_x\_position();
305. **const** **double** p2tY = get\_y\_position();
306. move\_distance(L3/10);
307. **const** **double** p1tX = get\_x\_position();
308. **const** **double** p1tY = get\_y\_position();
309. move\_distance(L3/10);
310. note\_position(); //4
311. set\_pen\_color(color::grey);
312. fill\_shape();
314. //Lining Cannon Head
315. set\_pen\_width(1);
316. set\_pen\_color(color::black);
317. move\_to(p1tX,p1tY);
318. draw\_to(p1bX,p1bY);
319. move\_to(p2bX,p2bY);
320. draw\_to(p2tX,p2tY);
321. move\_to(p3tX,p3tY);
322. draw\_to(p3bX,p3bY);
324. turn\_right\_by\_degrees(87);
325. move\_to(posX,posY);
327. //Lining the Canon
328. set\_pen\_width(1.5);
329. set\_pen\_color(color::black);
330. turn\_right\_by\_degrees(87);
331. draw\_distance(L3);
332. turn\_right\_by\_degrees(93);
333. draw\_distance(W1);
334. turn\_right\_by\_degrees(93);
335. draw\_distance(L3);
336. turn\_right\_by\_degrees(87);
337. draw\_distance(W2);
338. //----------------
340. turn\_right\_by\_degrees(87);
341. draw\_distance(L2-r);
342. set\_pen\_color(color::white);
343. draw\_distance(r);
344. set\_pen\_color(color::black);
346. move\_to(cX,cY);
347. start\_shape();
348. cannonWheel(r, 360, 1, 2.5);
349. set\_pen\_color(color::grey);
350. fill\_shape();
351. cannonWheel(r, 360, 0, 2.5);
352. cannonWheel(r/2, 360, 0, 1.5);
354. move\_to(eX,eY);
355. set\_heading\_degrees(90-angle);
356. **return** angle;
357. }
359. **int** cannon(**const** **double** v, **const** **double** ground, **const** **double** enemy\_size, **const** **double** enemy\_pos, **const** **double** angle, **const** **double** t = 0)
360. {
361. **if** ( (get\_x\_position() >= enemy\_pos-enemy\_size) && ( get\_x\_position() <= (enemy\_pos - enemy\_size/8) ) && (get\_y\_position() >= ground-enemy\_size) && (get\_y\_position() <= ground) )
362. {
363. print("The cannon hits the enemy after ");
364. print(t);
365. print(" seconds!\n");
366. boom(get\_x\_position()-10,get\_y\_position(),15);
367. set\_pen\_color(color::black);
368. **return** 1;
369. }
370. **else** **if** ( get\_y\_position() >= ground )
371. {
372. print("The cannon touches the ground after ");
373. print(t);
374. print(" seconds.\n");
375. set\_pen\_color(color::black);
376. **return** 0;
377. }
378. **else**
379. {
380. set\_pen\_width(2);
381. **const** **double** h =  ( v \* t \* cos( (90-angle)\*PI/180) ) - ( 0.5 \* 9.807 \* pow( t ,2) );
382. **const** **double** d =  ( v \* t \* sin( (90-angle)\*PI/180) );
383. **const** **double** x = get\_x\_position()+d/60;
384. **const** **double** y = get\_y\_position()-h/60;
385. set\_pen\_color(250+t\*31);
386. wait(1.0/1000);
387. draw\_to(x,y);
388. //cout << "\n \* Time " << t;
389. set\_pen\_width(1);
390. cannon(v, ground, enemy\_size, enemy\_pos, angle, t+1.0/60);
391. }
392. }
394. **void** draw\_ground(**const** **double** gY,**const** **double** originalG, **const** **int** n=0)
395. {
396. **if** ( get\_x\_position() >= get\_window\_width() && get\_y\_position() >= get\_window\_height() )
397. {
398. set\_pen\_color(color::black);
399. move\_to(0.0, originalG+1);
400. draw\_to((**float**)get\_window\_width(), originalG+1);
401. set\_pen\_width(2);
402. **return**;
403. }
404. **else**
405. {
406. set\_pen\_color(color::light\_grey+n\*3.5);
407. set\_pen\_width(1);
408. move\_to(0.0, gY+1);
409. draw\_to((**float**)get\_window\_width(), gY+1);
410. draw\_ground(gY+1,originalG,n+1);
411. }
412. }

415. **double** draw\_enemy(**const** **double** ground\_point\_y)
416. {
417. set\_pen\_width(1);
418. set\_heading\_degrees(0);
419. **const** **double** x = (**double**)random\_in\_range(600,1000);
420. **const** **double** size = (**double**)random\_in\_range(40,100);
421. **double** positionX, positionY;
423. //Drawing Level 1
424. start\_shape();
425. move\_to(x,ground\_point\_y);
426. draw\_distance(size/2);
427. note\_position(); //1
428. positionX = get\_x\_position();
429. positionY = get\_y\_position();
430. set\_heading\_degrees(-90);
431. draw\_distance(size);
432. note\_position(); //2
433. set\_heading\_degrees(180);
434. draw\_distance(size/2);
435. note\_position(); //3
436. set\_heading\_degrees(90);
437. move\_distance(size);
438. note\_position(); //4
439. set\_pen\_color(color::grey);
440. fill\_shape();
441. //Coloring Level 1
442. start\_shape();
443. move\_to(positionX,positionY);
444. set\_heading\_degrees(-90);
445. note\_position(); //1
446. set\_heading\_degrees(-90);
447. draw\_distance(size);
448. note\_position(); //2
449. set\_heading\_degrees(180);
450. draw\_distance(size/2);
451. note\_position(); //3
452. set\_pen\_color(color::dark\_grey);
453. fill\_shape();
454. set\_pen\_color(color::black);
455. move\_to(positionX,positionY);
456. set\_heading\_degrees(-90);
457. draw\_distance(size);
458. set\_heading\_degrees(180);
459. draw\_distance(size/2);
460. set\_heading\_degrees(90);
461. move\_distance(size);
462. set\_heading\_degrees(0);
463. draw\_distance(size/2);
464. //Drawing Level 2
465. start\_shape();
466. move\_to(positionX,positionY);
467. set\_heading\_degrees(-90);
468. draw\_distance(size/4);
469. note\_position(); //1
470. set\_heading\_degrees(0);
471. draw\_distance(size/2);
472. note\_position(); //2
473. set\_heading\_degrees(-90);
474. draw\_distance(size/4);
475. positionX = get\_x\_position();
476. positionY = get\_y\_position();
477. draw\_distance(size/4);
478. note\_position(); //3
479. set\_heading\_degrees(180);
480. draw\_distance(size/2);
481. note\_position(); //4
482. set\_pen\_color(color::grey);
483. fill\_shape();
485. //Coloring Level 2
486. move\_to(positionX+size/4,positionY);
487. start\_shape();
488. note\_position(); //1
489. set\_heading\_degrees(-90);
490. draw\_distance(size/2);
491. note\_position(); //2
492. set\_heading\_degrees(180);
493. draw\_distance(size/2);
494. note\_position(); //3
495. set\_pen\_color(color::dark\_grey);
496. fill\_shape();
498. set\_pen\_color(color::black);
499. move\_to(positionX+size/4,positionY);
500. set\_heading\_degrees(-90);
501. draw\_distance(size/2);
502. set\_heading\_degrees(180);
503. draw\_distance(size/2);
504. set\_heading\_degrees(90);
505. draw\_distance(size/2);
506. set\_heading\_degrees(0);
507. draw\_distance(size/2);
509. //Flag
510. move\_to(positionX,positionY);
511. set\_heading\_degrees(0);
512. draw\_distance(size/2);
514. //Drawing Flag
515. start\_shape();
516. note\_position(); //1
517. turn\_right\_by\_degrees(90);
518. draw\_distance(size/4);
519. note\_position(); //2
520. turn\_right\_by\_degrees(90);
521. draw\_distance(size/8);
522. note\_position(); //3
523. turn\_right\_by\_degrees(90);
524. draw\_distance(size/4);
525. note\_position(); //4
526. set\_pen\_color(color::dark\_red);
527. fill\_shape();
529. set\_pen\_color(color::black);
530. turn\_right\_by\_degrees(90);
531. draw\_distance(size/8);
532. turn\_right\_by\_degrees(90);
533. draw\_distance(size/4);
534. turn\_right\_by\_degrees(90);
535. draw\_distance(size/8);
536. turn\_right\_by\_degrees(90);
537. draw\_distance(size/4);
538. move\_to(x,ground\_point\_y);
539. **return** size;
540. }
542. **void** game\_message(**int** x, **int** y, string message, **int** color=0)
543. {
544. **const** **double** original\_x = get\_x\_position();
545. **const** **double** original\_y = get\_y\_position();
546. move\_to(x,y);
547. **if**(color==1)
548. set\_pen\_color(color::dark\_green);
549. **if**(color==2)
550. set\_pen\_color(color::dark\_red);
551. write\_string(message);
552. move\_to(original\_x,original\_y);
553. set\_pen\_color(color::black);
554. }
556. **void** game\_message2(string message, **int** size, **int** color)
557. {
558. **const** **double** original\_x = get\_x\_position();
559. **const** **double** original\_y = get\_y\_position();
561. start\_shape();
562. move\_to( 200, 110 );
563. note\_position();
564. draw\_to( 800, 110 );
565. note\_position();
566. draw\_to( 800, 310 );
567. note\_position();
568. draw\_to( 200, 310 );
569. note\_position();
570. draw\_to( 200, 110 );
571. note\_position();
573. **if**(color==0)
574. set\_pen\_color(color::light\_grey-90);
576. **if**(color==1)
577. set\_pen\_color(color::green);
579. **if**(color==2)
580. set\_pen\_color(color::dark\_red);
582. fill\_shape();
583. set\_pen\_color(color::black);
585. move\_to( 200, 110 );
586. draw\_to( 800, 110 );
587. draw\_to( 800, 310 );
588. draw\_to( 200, 310 );
589. draw\_to( 200, 110 );
591. move\_to(500,210);
592. set\_font\_size(size);
593. write\_string(message,direction::center);
594. move\_to(original\_x,original\_y);
595. set\_font\_size(0);
596. **return**;
597. }

600. **void** main()
601. {
602. make\_window(1000,600);
603. set\_pen\_width(3);
604. move\_to(50,500);
606. **char** run = 'Y';
607. **double** r = 20;
608. **double** angle, velocity, enemy\_size, enemy\_position, eX, eY;;
609. **double** cX = get\_x\_position();
610. **double** cY = get\_y\_position();
611. **double** ground = cY+r;
613. print("Lab 4 - Cannon Game\nMade By Haoyu Wang");
614. print("\n------------------------------------------------------------------------------\n");
615. print("This game presents an enemy of a random size and a random position in the area.\n");
616. print("Choose the angle of the cannon, and the velocity to hit your target.\n");
617. print("\n==============================================================================\n");
619. **while** ( run == 'Y' || run =='y' )
620. {
621. clear();
622. move\_to(10,20);
623. write\_string("Lab 4 - Cannon Game");
624. move\_to(210,30);
625. draw\_to(10,30);
627. draw\_ground(ground,ground);
628. enemy\_size = draw\_enemy(ground);
629. enemy\_position = get\_x\_position();
630. move\_to(cX,cY);
632. print("Enter the angle (degrees) = ");
633. angle = read\_float();
634. angle = cannonBody(r, angle);
635. eX = get\_x\_position();
636. eY = get\_y\_position();
638. print("\nYou have three chances to hit the enemy!");
639. print("\nATTEMPT NUMBER 1");
640. game\_message(10,50,"Attempt 1:");
641. print("\nEnter the velocity (metres per second) = ");
642. move\_to(eX,eY);
643. velocity = read\_float();
645. **if**( cannon(velocity,ground,enemy\_size,enemy\_position,angle) == 1)
646. {
647. game\_message(110,50,"Congratulations! You hit the target!",1);
648. game\_message2("You Win!",100,1);
649. }
650. **else**
651. {
652. game\_message(110,50,"You missed the target.",2);
653. print("\nATTEMPT NUMBER 2");
654. game\_message(10,70,"Attempt 2:");
655. print("\nEnter the velocity (metres per second) = ");
656. move\_to(eX,eY);
657. velocity = read\_float();
659. **if**( cannon(velocity,ground,enemy\_size,enemy\_position,angle) == 1)
660. {
661. game\_message(110,70,"Congratulations! You hit the target!",1);
662. game\_message2("You Win!",100,1);
663. }
664. **else**
665. {
666. game\_message(110,70,"You missed the target.",2);
667. print("\nATTEMPT NUMBER 3");
668. game\_message(10,90,"Attempt 3:");
669. print("\nEnter the velocity (metres per second) = ");
670. move\_to(eX,eY);
671. velocity = read\_float();
673. **if**( cannon(velocity,ground,enemy\_size,enemy\_position,angle) == 1)
674. {
675. game\_message(110,90,"Congratulations! You hit the target!",1);
676. game\_message2("You Win!",100,1);
677. }
678. **else**
679. {
680. game\_message(110,90,"That's too bad. You missed the last shot.",2);
681. game\_message2("Game Over",100,2);
682. }
683. }
684. }
685. print("\n\nDo you want to play again? (type y or Y to play again, anything else to quit) ");
686. cin >> run;
687. print("\n==============================================================================\n");
688. }
689. clear();
690. game\_message2("Thanks for playing!",70,0);
691. **return**;
693. }

The program produces an enemy of a random shape, and a random location. The user gets to choose the angle of the cannon, then three chances to pick the right velocity to hit the enemy. If at any point the enemy gets hit, the game ends with the user winning, if not: game over. Then the user gets the option to play again.

The console will show the time of each cannon when it hits the ground or the target.

**Result (Win from the first try)**



**Result (Win from the second try)**



**Result (Win from the third try)**



**Result (Loss after three tries)**

