2020년도 2학기 컴퓨터공학설계및실험Ⅰ

10주차 WaterFall-1 week 결과 보고서

20161663 허재성

1. 실습 목적

openFrameWokrks에 대해 이해하고, 여러 예제와 Waterfall 프로그램을 구현하며 그 사용법을 익힌다.

2. 실습 구현 내용

ofApp.h 파일에 다음과 같은 클래스들을 추가했다.

Point 클래스는 점을 표현하는 클래스이다.

|  |
| --- |
| class Point {  private:  int xpos, ypos;  public:  Point();  Point(int \_x, int \_y);  void setX(int \_x);  void setY(int \_y);  int getX() const;  int getY() const;  }; |

멤버 변수

(1) int xpos, ypos;

스크린 위의 점의 x 좌표와 y 좌표를 나타내는 멤버 변수이다.

생성자 및 멤버 함수

(1) Point()

Point 인스턴스를 생성하는 기본 생성자로 xpos와 ypos의 값을 0으로 초기화한다.

input : 없음

output : 없음

(2) Point(int \_x, int \_y)

Point 인스턴스를 생성하는 생성자로 정수형 변수 \_x, \_y를 인자로 받아 xpos, ypos를 각각의 값으로 초기화한다.

input : xpos를 초기화할 값 \_x와 ypos를 초기화할 값 \_y

output : 없음

(3) void setX(int \_x), void setY(int \_y)

각각 Point 인스턴스의 xpos, ypos의 값을 설정 또는 변경하는 함수이다. setX 함수로 xpos의 값을 \_x로, ypos의 값을 \_y로 설정한다.

input : 각각 변경할 xpos의 값 \_x, 변경할 ypos의 값 \_y

output : 없음

(4) int getX() const, int getY() const

각각 Point 인스턴스의 xpos, ypos 값을 반환하는 함수이다.

input : 없음

output : 각각 xpos, ypos

LineSegment 클래스는 두 점을 잇는 선분을 표현하는 클래스이다.

|  |
| --- |
| class LineSegment {  private:  Point leftP, rightP;  public:  LineSegment(int xl, int yl, int xr, int yr);  int getLX() const;  int getLY() const;  int getRX() const;  int getRY() const;  }; |

멤버 변수

(1) Point leftP, rightP;

각각 선분의 양 끝 점인 왼쪽 끝 점과 오른쪽 끝 점을 나타내는 Point 객체 변수이다

생성자 및 멤버 함수

(1) LineSegment(int xl, int yl, int xr, int yr)

선분 LineSegment 인스턴스의 양 끝 두 점 leftP, rightP의 좌표를 설정하는 생성자이다. 왼쪽 끝 점 leftP의 x좌표, y좌표를 각각 xl, rl로 설정, 오른쪽 끝 점 rightP의 x좌표, y좌표를 각각 xr, yr로 설정한다. 설정할 때, Point 클래스의 멤버 함수인 setX, setY를 이용한다.

input : 왼쪽 끝 점의 x좌표와 y좌표, 오른쪽 끝 점의 x좌표와 y좌표인 xl, yl, xr, yr

output : 없음

(2) int getLX() const, int getLY() const

각각 선분 LineSegment 인스턴스의 왼쪽 끝 점 leftP의 x 좌표와 y 좌표를 반환하는 함수이다. 반환 받을 때 Point 클래스의 멤버 함수인 getX, getY를 이용한다.

input : 없음

output : 각각 선분의 왼쪽 끝 점 leftP의 x 좌표, y 좌표

(3) int getRX() const, int getRY() const

각각 선분 LineSegment 인스턴스의 오른쪽 끝 점 rightP의 x 좌표와 y 좌표를 반환하는 함수이다. 반환 받을 때 Point 클래스의 멤버 함수인 getX, getY를 이용한다.

input : 없음

output : 각각 선분의 오른쪽 끝 점 rightP의 x 좌표, y 좌표

Waterfall 클래스는 Waterfall을 구현하기 위해 필요한 물이 나오는 구멍(점)과 선반(선분)을 묶에서 관리하기 위한 클래스이다. 다음과 같은 멤버 변수와 멤버 함수가 있다.

|  |
| --- |
| Class Waterfall {  private:  int wh;  int sh;  int pos; // 선택한 점의 index  public:  Point\*\* waterhole;  LineSegment\*\* shelf;  Waterfall();  ~Waterfall();  void setWH(int \_wh);  void setSH(int \_sh);  void setPos(int \_pos);  int getWH() const;  int getSH() const;  int getPos() const;  }; |

멤버 변수

(1) int wh, sh;

각각 물이 나오는 구멍(점)의 개수와 선반(선분)의 개수를 나타내는 정수형 변수이다.

(2) int pos;

뒤에서 구현할 물이 나올 점을 선택했을 때 해당 점의 스크린 상에서의 위치(인덱스)이다. 점의 개수가 wh이므로 인덱스 pos는 0~wh-1의 값을 가진다. 첫 번째 점은 인덱스 pos = 0, 마지막 점은 pos = wh -1이 된다.

(3) Point\*\* waterhole;

Point 객체의 주소(Point\*)를 저장하는 포인터 배열을 가리키기 위한 Point\*\*형 변수이다. 물이 나오는 점의 개수 wh개의 크기를 가지는 배열이 동적 할당된다.

(4) LineSegment\*\* shelf;

LineSegment 객체의 주소(LineSegment\*)를 저장하는 포인터 배열을 가리키기 위한 LineSegment\*\*형 변수이다. 선반의 개수 sh개의 크기를 가지는 배열이 동적 할당된다.

생성자, 소멸자 및 멤버 함수

(1) Waterfall()

기본 생성자로 객체의 이중 포인터 변수 waterhole과 shelf를 NULL로 초기화하고, wh, sh, pos를 모두 0으로 초기화한다.

input : 없음

output : 없음

(2) ~Waterfall()

동적 할당된 포인터 배열 waterhole과 shelf의 각 인덱스를 돌며 각 인덱스에 주소가 저장된 동적 할당된 Point 객체와 LineSegement 객체를 해제해준다. 모든 동적 할당된 객체가 해제된 후에 동적 할당된 포인터 배열 waterhole과 shelf를 해제해준다.

input : 없음

output : 없음

(3) void setWH(int \_wh), void setSH(int \_sh), void setPos(int \_pos)

각각 멤버 변수 wh, sh, pos의 값을 \_wh, \_sh, \_pos로 설정한다.

input : 각각 wh, sh, pos를 초기화할 값 \_wh, \_sh, \_pos

output : 없음

(4) int getWH() const, int getSH() const, int getPos() const

각각 멤버 변수 wh, sh, pos의 값을 반환한다.

input : 없음

output : 각각 wh, sh, pos의 값

추가, 변경된 변수, 함수

텍스트 파일을 읽었을 때, 해당 데이터가 스크린의 범위를 넘어가는 지 확인하기 위해 헤더파일ofApp.h에 선언된 클래스 ofApp에 멤버 변수로 range\_flag를 추가한다. 특정 텍스트 파일을 읽어서 범위를 벗어나는 점이 있으면 range\_flag를 1로 설정한다.

선분과 점을 구현하고 화면에 표시하기 위해 소스 파일 ofApp.cpp을 수정한다.

ofApp.cpp 파일에 전역적으로 Waterfall 포인터 wf를 추가한다.

|  |
| --- |
| Waterfall\* wf = NULL; |

화면에 점을 그리는 것은 원을 그리는 것으로 구현할 수 있다. 점의 좌표는 원의 중심의 좌표가 되고 중심을 기준으로 지름을 설정한다. 지름의 길이가 적당하면 원을 점으로 간주할 수 있다. 이번 문제에서는 지름 dot\_diameter를 실수 10.0f으로 설정한다. 점의 지름은 ofApp.cpp에 정의되어 있는 setup 함수에서 설정할 수 있다. 추가적으로 setup 함수에서 range\_flag의 초기값을 0으로 설정한다.

|  |
| --- |
| void ofApp::setup() {  /\* 생략 \*/  range\_flag = 0;  dot\_diameter = 10.0f;  } |

TODO – 1 구현

기본적으로 주어진 소스 파일 main.cpp, ofApp.cpp를 컴파일 후 실행하면 갈색 천장과 바닥이 있는 하얀 색 스크린이 뜬다. ‘l’ 키를 누르고 입력 텍스트 파일을 설정했을 때, 해당 텍스트 파일에 존재하는 선분의 개수와 정보(선분 양 끝 점의 좌표), 점의 개수와 좌표를 읽어와 저장해야 한다. 이를 위해 ofApp.cpp의 processOpenFileSection 함수를 수정한다.

파일을 열고 wf에 Waterfall 객체를 동적 할당한다. 동적 할당한 wf의 포인터 배열 waterhole과 shelf에 텍스트 파일에서 읽어온 점의 정보와 선분의 정보를 저장해야 한다. 이 때, wf가 NULL이 아니라면, 즉 이전에 텍스트 파일을 열어서 동적 할당한 wf가 존재한다면 해제 후 다시 동적 할당한다.

wf를 동적할당한 후 텍스트 파일을 줄 단위로 읽으며 데이터를 저장한다. 줄 단위로 읽을 때, 띄어쓰기를 기준으로 단어 수를 계산하고 단어 수가 1개인 경우, 선분 또는 점의 개수이다.

input\_type이 0일 때, 처음 나오는 1개의 단어는 선분 개수이므로 해당 수를 num\_of\_line에 저장 후 wf의 멤버 변수 sh를 num\_of\_line으로 설정한다. 설정 후, sh 크기를 가지는 LineSegement 포인터 배열을 동적 할당한 후에 wf의 shelf가 가리킨다. 선분의 개수를 읽은 후, 선분의 개수만큼 4개의 단어로 이루어진 줄들이 나온다. 4개의 단어들은 선분의 왼쪽 끝 점, 오른쪽 끝 점의 좌표로 해당 점의 좌표들이 스크린의 범위를 넘어가지 않는지 확인한다. 스크린의 범위(0<= x <= 1,024, 40 <= y <= 728)를 넘어갈 경우, 오류 메시지를 출력하고 range\_flag를 1로 설정 후 함수를 종료한다. 범위를 넘어가지 않을 경우 해당 선분을 shelf에 저장한다. 선분을 모두 저장하면 input\_type을 1로 설정해 점에 대한 입력을 받도록 한다.

input\_type이 1일 때 한 단어는 점의 개수 num\_of\_dot로 입력을 받고 wf의 멤버 변수 wf를 num\_of\_dot의로 설정한다. 설정 후, wh 크기를 가지는 Point 포인터 배열을 동적 할당한 후에 wf의 waterhole이 가리킨다. 점의 개수를 읽은 후, 점의 개수만큼 2개의 단어로 이루어진 줄들이 나오는데 2개의 단어들은 점의 좌표에 해당한다. 해당 점의 좌표들이 스크린의 범위를 넘어가지 확인 후, 위와 마찬가지로 넘어갈 경우 오류 메시지를 출력 후, range\_flag를 1로 설정 후 함수를 종료한다. 범위를 넘어가지 않을 경우 해당 점을 waterhole에 저장한다. 점을 모두 저장했다면 입력파일이 정상적이라는 가정 하에 반복문이 종료되어 함수도 종료된다.

범위를 넘어가는 점이 있을 경우, range\_flag가 1이 된 채로 함수가 종료되고, 그렇지 않을 경우 range\_flag는 0인 상태에서 wf의 shelf, waterhole에 선반, 물이 나오는 점들의 정보가 저장된다.

다음으로 수정할 함수는 keyReleased 함수이다. ‘l’ 키를 눌렀을 때, 텍스트 파일을 선택하는 다이얼로그 창을 띄우고 파일을 선택하면 processOpenFileSelection 함수를 호출하는 함수이다. 다이얼로그 창을 띄우기 전에 이전에 변경되었을 수 있는 draw\_flag와 load\_flag를 0으로 설정한다. processOpenFileSelection을 호출하여 range\_flag가 1일 경우, 범위를 초과하였다는 의미이므로 load\_flag를 0으로 설정하여 해당 파일이 로드 되지 않았음을 알리고 다음 파일을 위해 range\_flag는 다시 0으로 설정한다. 반대로 range\_flag가 0일 경우, 범위를 초과하지 않았다는 의미이므로 load\_flag를 1로 설정하여 파일이 정상적으로 로드되었음을 알린다. 방향키 입력 관련 코드의 수정은 뒤에서 보인다.

의사 코드는 다음과 같다.

|  |
| --- |
| void ofApp::processOpenFileSelection(ofFileDialogResult openFileResult) {  파일 입력;  if(파일이 존재하지 않음) {  파일이 존재하지 않는다는 메시지 출력;  return;  }  파일을 찾았다는 메시지 출력;  int input\_type = 0;  int sh = 0, wh = 0;  if (wf가 이미 할당되어 있는 경우) wf를 해제;  wf를 (재)할당;  for (텍스트 파일의 줄을 한 줄씩 읽음) {  단어로 구분;  if (한 단어일 경우) {  if (input\_type == 0) { // 선분 정보일 경우  num\_of\_line = 선분의 개수;  선분의 개수 출력;  wf의 sh를 num\_of\_line으로 설정;  wf의 shelf에 sh 크기의 LineSegment 포인터 배열 동적 할당;  }  else { // 점 정보일 경우  num\_of\_dot = 점의 개수;  점 개수 출력;  wf의 wh를 num\_of\_dot으로 설정;  wf의 waterhole에 wh 크기의 Point 포인터 배열 동적 할당;  }  }  else if (한 단어가 아닐 경우) {  int x1, y1, x2, y2;  if (input\_type == 0) { // 선분 정보일 경우  x1, y1, x2, y2를 각각 선분의 왼쪽 끝 점 좌표, 오른쪽 끝 점 좌표로 함;  if (범위를 벗어날 경우) {  cout << "범위를 벗어난 입력" << endl;  range\_flag = 1; // 범위 초과 플래그 설정  return;  }  wf의 LineSegment 포인터 배열 shelf의 sh번째 원소로 선분 저장;  sh++;  if (선분을 모두 저장했을 경우) input\_type = 1;  }  else { // Input for actual information of dots.  x1, y1을 점의 좌표로 함;    if (범위를 벗어날 경우) {  cout << "범위를 벗어난 입력" << endl;  range\_flag = 1;  return;  }  wf의 Point 포인터 배열 waterhole의 wh번째 원소로 선분 저장;  wh++;  }  } // End of else if.  } // End of for-loop (Read file line by line).  //initializeWaterLines();  } |

TODO – 2 구현

다음으로 수정할 함수는 keyPressed 함수이다. ‘d’ 키를 입력 받았을 경우의 처리를 구현한다. 만약 ‘d’ 키를 입력받았는데, load\_flag가 0이라면 해당 파일이 로드되지 않았다는 의미이므로 함수를 종료한다. load\_flag가 1일 경우, 파일이 로드되었고 선반과 점의 정보가 저장되었다는 의미이다. 따라서 선반과 점을 그릴 수 있도록 draw\_flag를 1로 설정한다. ‘d’를 누르면 선반과 점이 그려져야 하므로 draw 함수를 호출한다. draw 함수를 호출하기 전에 ‘d’ 키를 누르면 물이 나오는 점(빨간 색으로 표시된 점)이 첫 번째 점(가장 왼쪽의 점)이 되어야 하므로 wf의 멤버 변수 pos를 0으로 설정한 후 draw 함수를 호출한다. ‘q’ 키를 입력받았을 때에는 동적 할당한 객체들을 모두 해제 후 종료해야 한다. draw\_flag를 0으로 리셋 후 wf를 해제한다. 해제 후 메시지를 출력 후 프로그램을 종료한다.

의사 코드는 다음과 같다.

|  |
| --- |
| void ofApp::keyPressed(int key) {  if (입력 키가 ‘q’일 경우) {  draw\_flag를 0으로 리셋  동적 할당된 wf 해제;  메모리 해제 완료 메시지 출력;  프로그램 종료;  }  if (입력 키가 ‘d’일 경우) {  if (load\_flag가 0일 경우) { // 파일이 로드되지 않은 경우  return; // 함수 종료  }  선분과 점을 그리기 위해 draw\_flag를 1로 설정;  물이 나오는 점을 첫 번째 점으로 설정; // 위치를 0으로 설정  draw();  }  } |

TODO – 3 구현

다음으로 수정할 함수는 draw 함수이다. 기본적으로 draw 함수에는 천장과 바닥을 갈색으로 그리는 코드가 작성되어 있다. 프로그램이 처음 실행되면 draw\_flag가 0인 상태에서 draw 함수가 호출되어 천장과 바닥만이 그려진다. 추가적으로 ‘d’ 키를 눌러 draw\_flag가 1이 된 상태에서 draw 함수가 호출될 경우 점과 선분을 그리기 위한 코드를 추가한다. if문으로 draw\_flag가 1인지 확인 후, 1일 경우 shelf에 저장된 선반 정보를 이용해 선반을 그리고 waterhole에 저장된 점들을 그린다. 선반을 그릴 때에는 선분을 그리는 함수 ofDrawLine을 이용해 그린다. 점을 그릴 때에는 원을 그리는 함수 ofDrawCircle 함수를 이용해 그린다. 점을 그릴 때 해당 점의 인덱스가 pos와 같을 경우, 해당 점이 물이 나오는 점이므로 ofSetColor(255, 0, 0)을 호출해 빨갛게 그린다. 그렇지 않을 경우, ofSetColor(0, 0, 0)을 호출해 까맣게 그린다.

의사 코드는 다음과 같다.

|  |
| --- |
| void ofApp::draw() {  천장과 바닥을 갈색으로 그림;  선의 굵기를 5로 설정;  if (draw\_flag) { // ‘d’ 키가 눌렸을 경우  for (모든 선분에 대하여) {  ofDrawLine()으로 선분을 그린다;  }  for (모든 점에 대하여) {  if (점의 좌표가 wf의 pos와 같을 경우)  ofSetColor(255, 0, 0)으로 빨간 색으로 설정;  else ofSetColor(0, 0, 0)으로 검은 색으로 설정;  ofDrawCircle()로 점을 그림;  }  }  } |

TODO – 4 구현

마지막으로 선분과 점이 그려진 상태에서 방향키 좌우를 눌러 물이 나오는 점의 위치를 바꾸는 기능을 구현해야 한다. 위의 함수 keyReleased 함수를 수정한다. 입력받은 키가 OF\_KEY\_RIGHT(오른쪽 방향키)일 경우와 OF\_KEY\_LEFT(왼쪽 방향키)인 경우로 나누어 생각한다. 두 경우 모두 draw\_flag가 0일 경우, 선반과 점이 그려지지 않았으므로 아무런 일도 하지 않는다. draw\_flag가 1일 경우에만 동작한다.

두 경우 모두 공통적으로 현재 물이 나오는 점의 위치 pos를 이용한다. 우측 방향키를 입력 받았을 경우, pos가 가장 오른쪽 점의 위치(wh – 1)일 경우, 우측 방향키를 입력 시 pos를 0으로 설정하여 가장 왼쪽 점이 선택되도록 한다. 그렇지 않을 경우, pos의 값을 1 증가시켜서 현재 점의 바로 오른쪽 점이 선택되도록 한다.

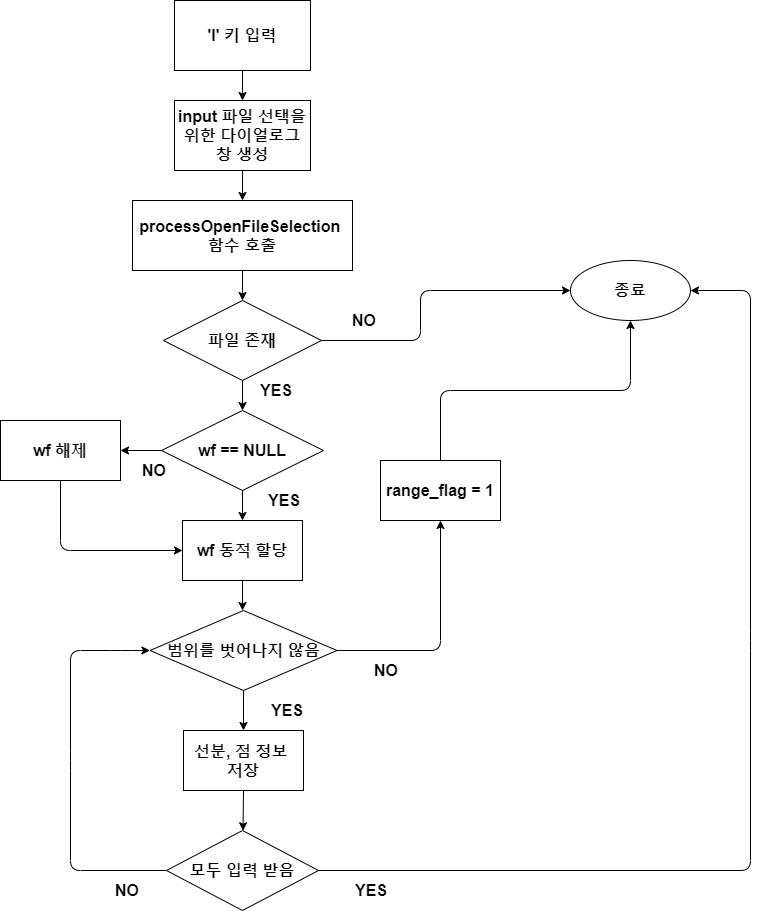
반대로 왼쪽 방향키를 입력 받았을 경우, pos가 가장 왼쪽 점의 위치(0)일 경우, 좌측 방향키를 입력 시 pos를 wh – 1로 설정하여 가장 오른쪽 점이 선택되도록 한다. 그렇지 않을 경우, pos의 값을 1 증가시켜서 현재 점의 바로 왼쪽 점이 선택되도록 한다.

위의 두 경우 모두 점의 위치를 재조정 후 draw 함수를 호출하여 선택된 점을 다시 그려준다. 그리고 새로 선택된 .점의 좌표를 출력한다.

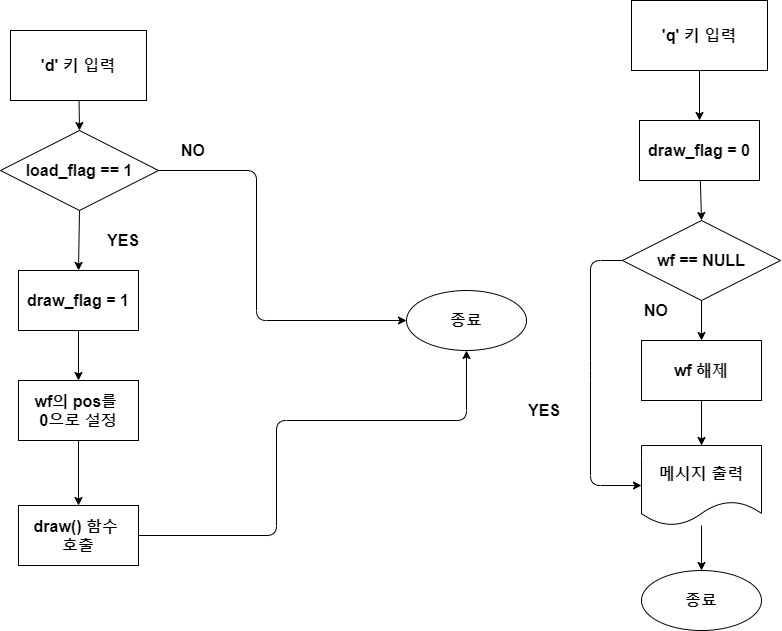
의사 코드는 다음과 같다. TODO-1의 내용도 일부 포함되어 있다.

|  |
| --- |
| void ofApp::keyReleased(int key) {  if (‘l’ 키를 입력했을 경우) {  이전에 변경되었을 수 있는 draw\_flag, load\_flag를 0으로 리셋함.  파일 선택을 위한 다이얼 로그창 띄움;  if (파일이 성공적으로 열린 경우) {  processOpenFileSelection 함수 호출;  if (!range\_flag) load\_flag = 1; // 범위 초과 없을 때 load\_flag를 1로 설정  else range\_flag = 0; // load\_flag는 0으로 유지하고 range\_flag 다시 0으로 설정  }  }  if (오른쪽 키를 입력했을 경우) {  if (draw\_flag) {  if (물이 나오는 점의 위치가 가장 오른쪽일 경우)  점의 위치를 0으로 설정;  else 점의 위치를 1 증가;  draw();  변경된 점의 좌표 출력;  }  }  if (왼쪽 키를 입력했을 경우) {  if (draw\_flag) {  if (물이 나오는 점의 위치가 가장 오른쪽일 경우)  점의 위치를 wh - 1으로 설정;  else 점의 위치를 1 감소;  draw();  변경된 점의 좌표 출력;  wf->waterhole[wf->getPos()]->getY() << ")" << endl;  }  }  } |

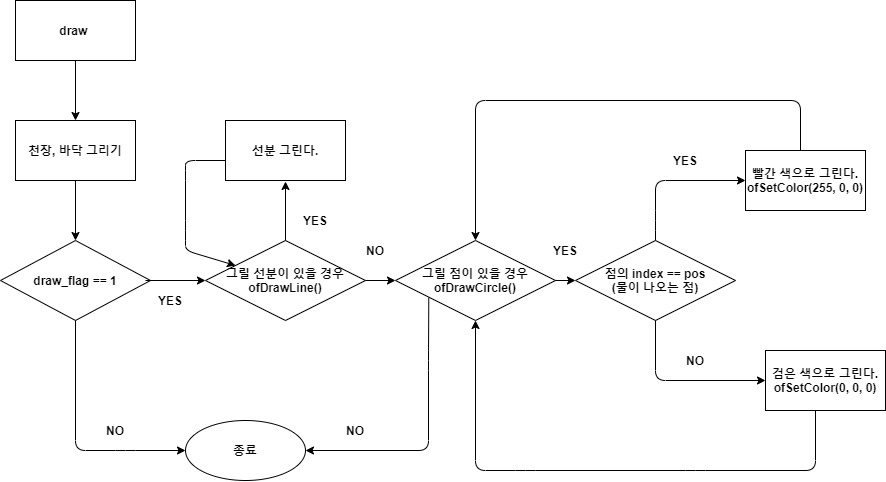
TODO-1의 순서도이다.



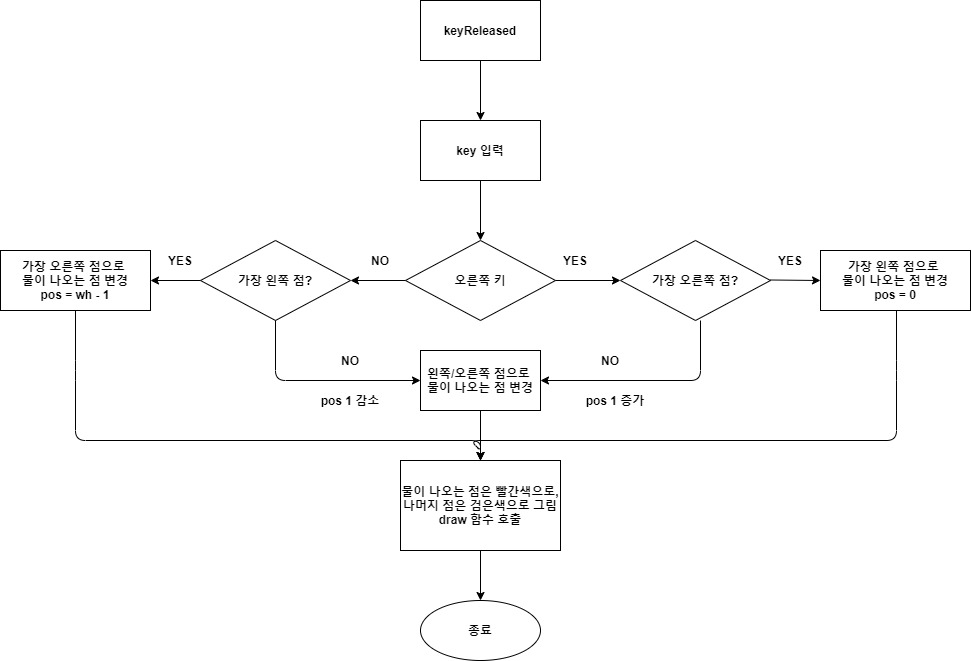
TODO-2의 순서도이다.



TODO-3의 순서도이다.



TODO-4의 순서도이다.



3. 실습 환경

OS : Windows 10

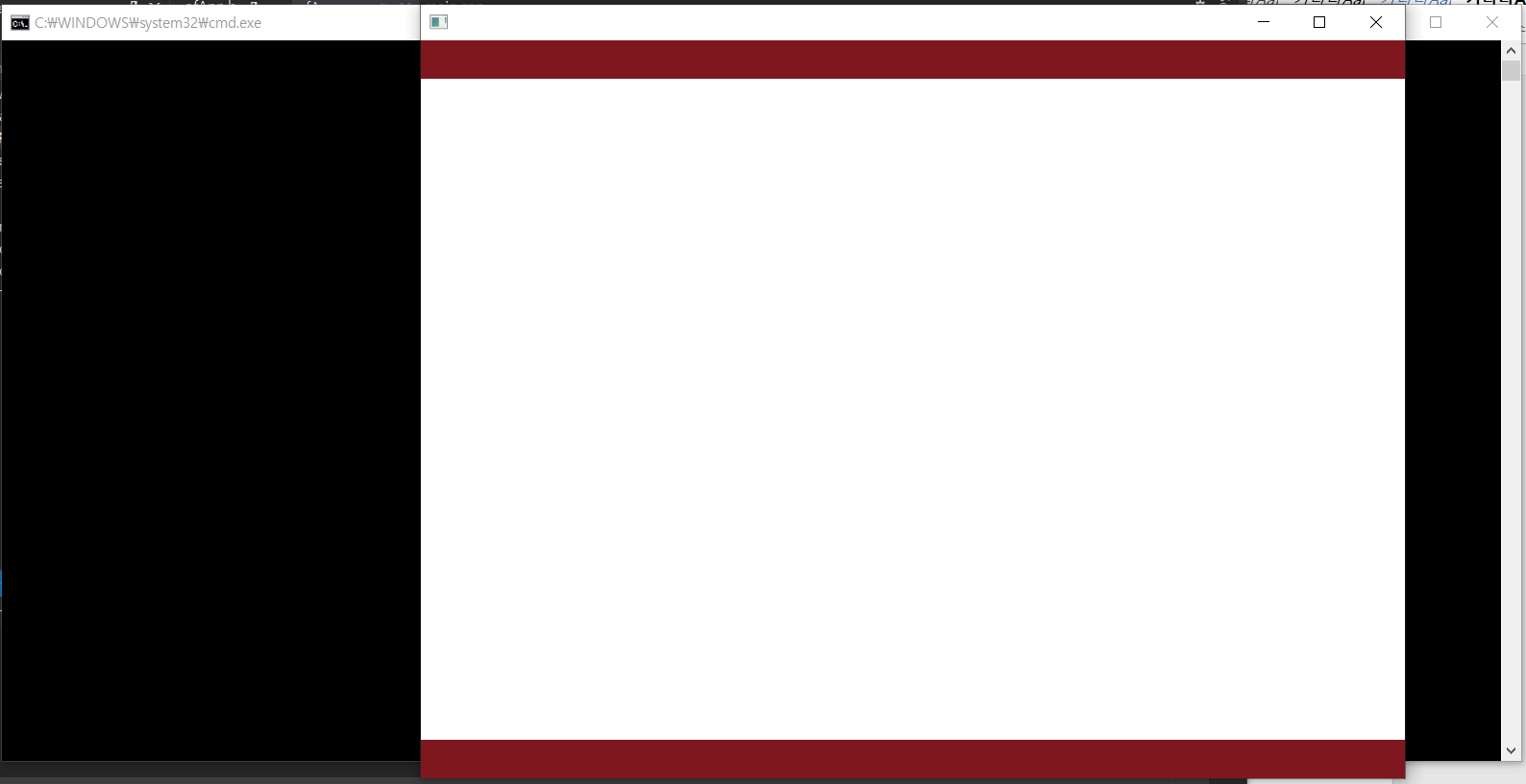
CPU : Intel® Core(TM) i3-5005U CPU @ 2.00GHz

RAM : 8.00GB

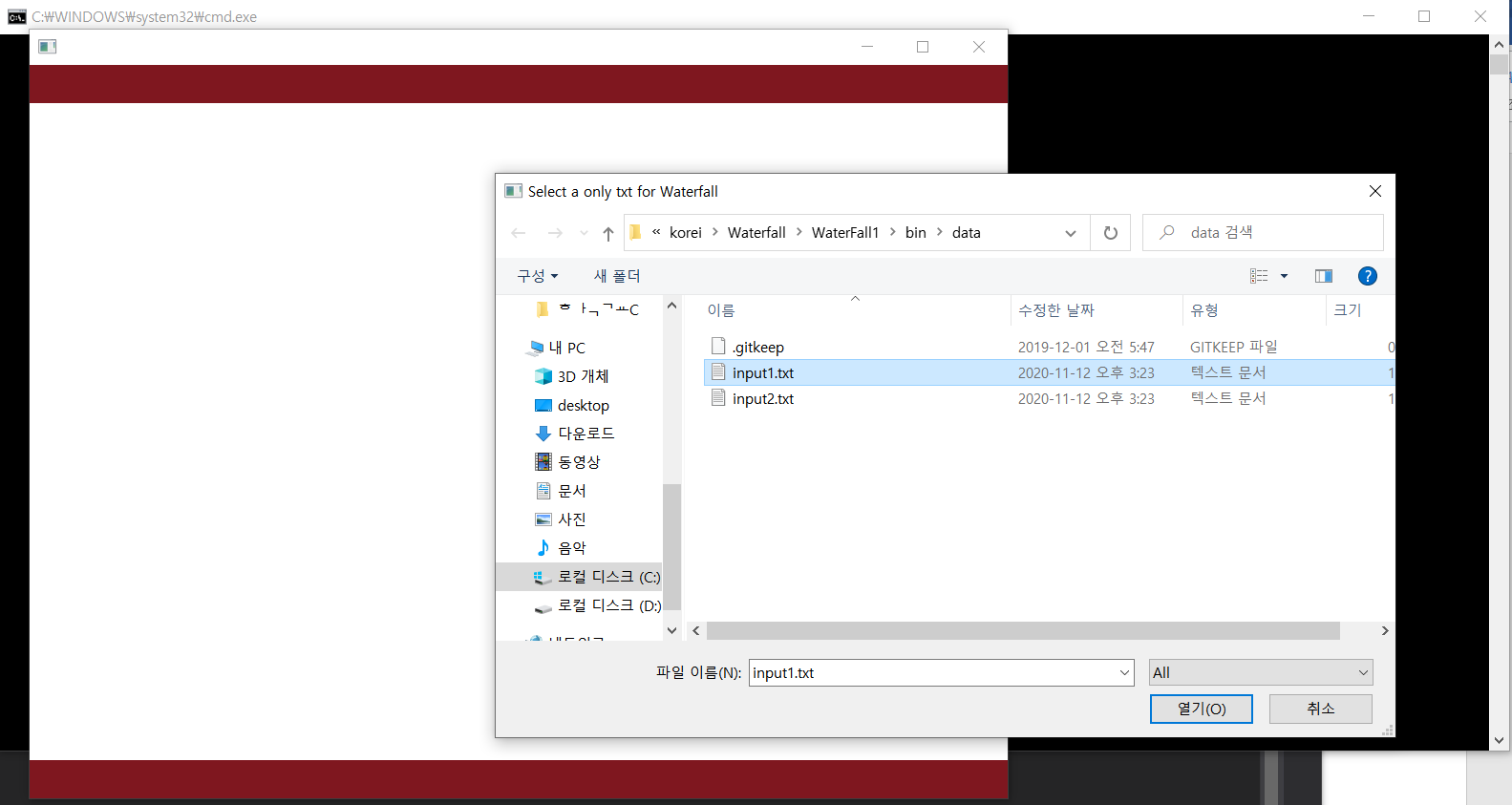
IDE : Visual Studio 2019 community

4. 실습 결과 및 분석

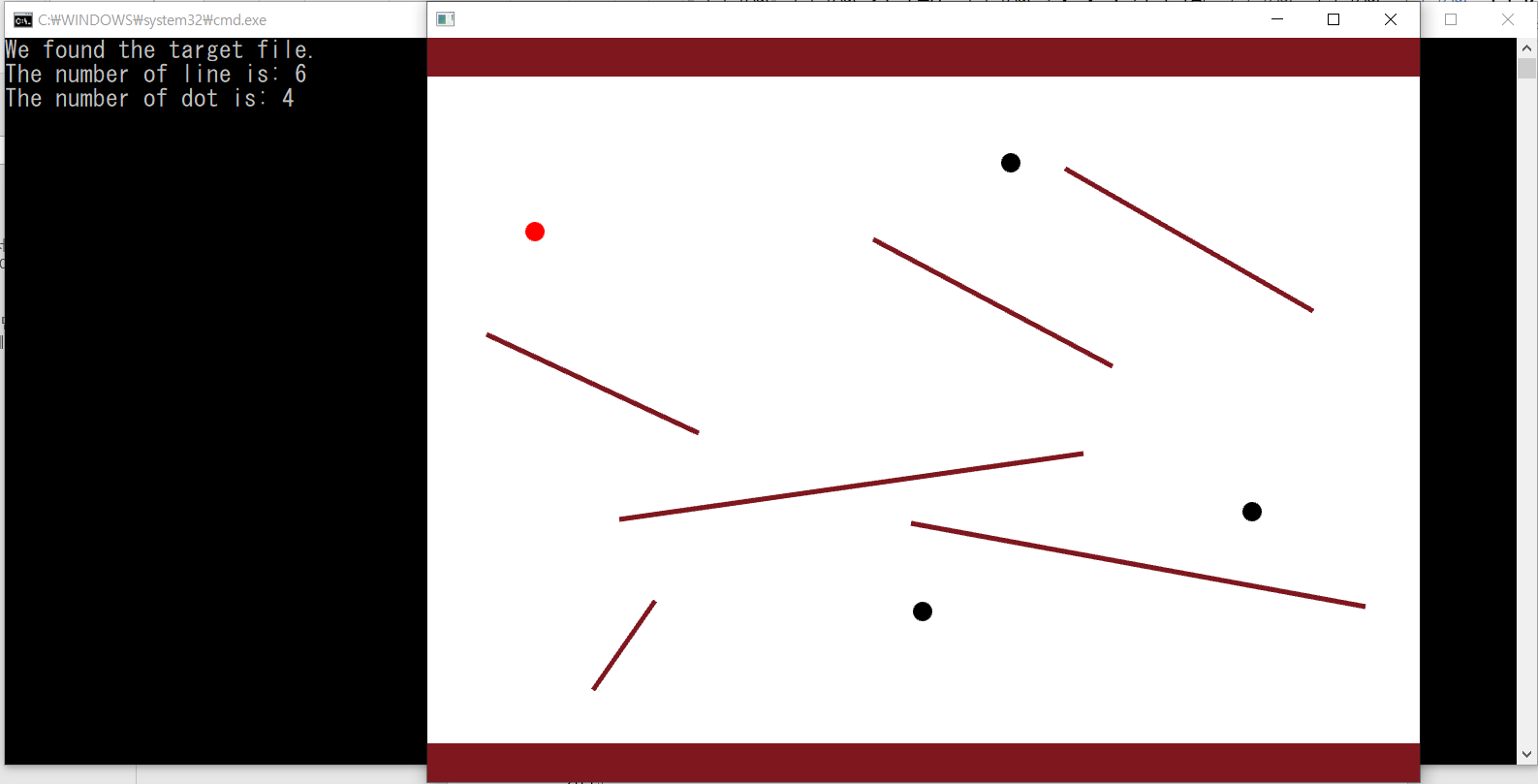
실습에 사용할 2개의 텍스트 파일로 input1, input2 파일이 있다. 1은 정상적인 텍스트 파일, 2는 한 점의 좌표가 스크린을 넘어가는 파일이다.



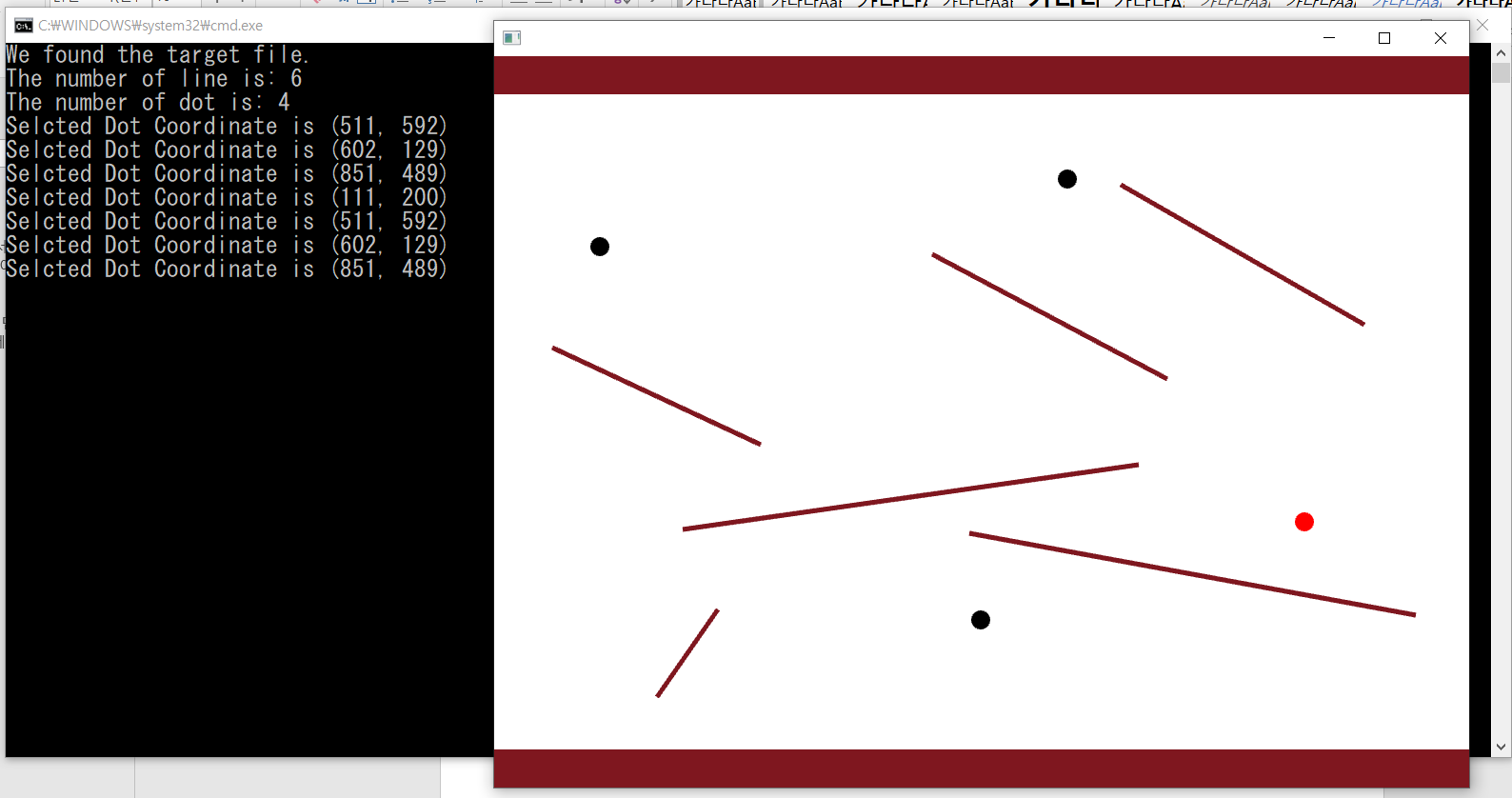
처음 프로그램을 실행시키면 콘솔 창과 함께 스크린이 화면에 뜬다. 이 상태에서 ‘l’ 키를 누른다.



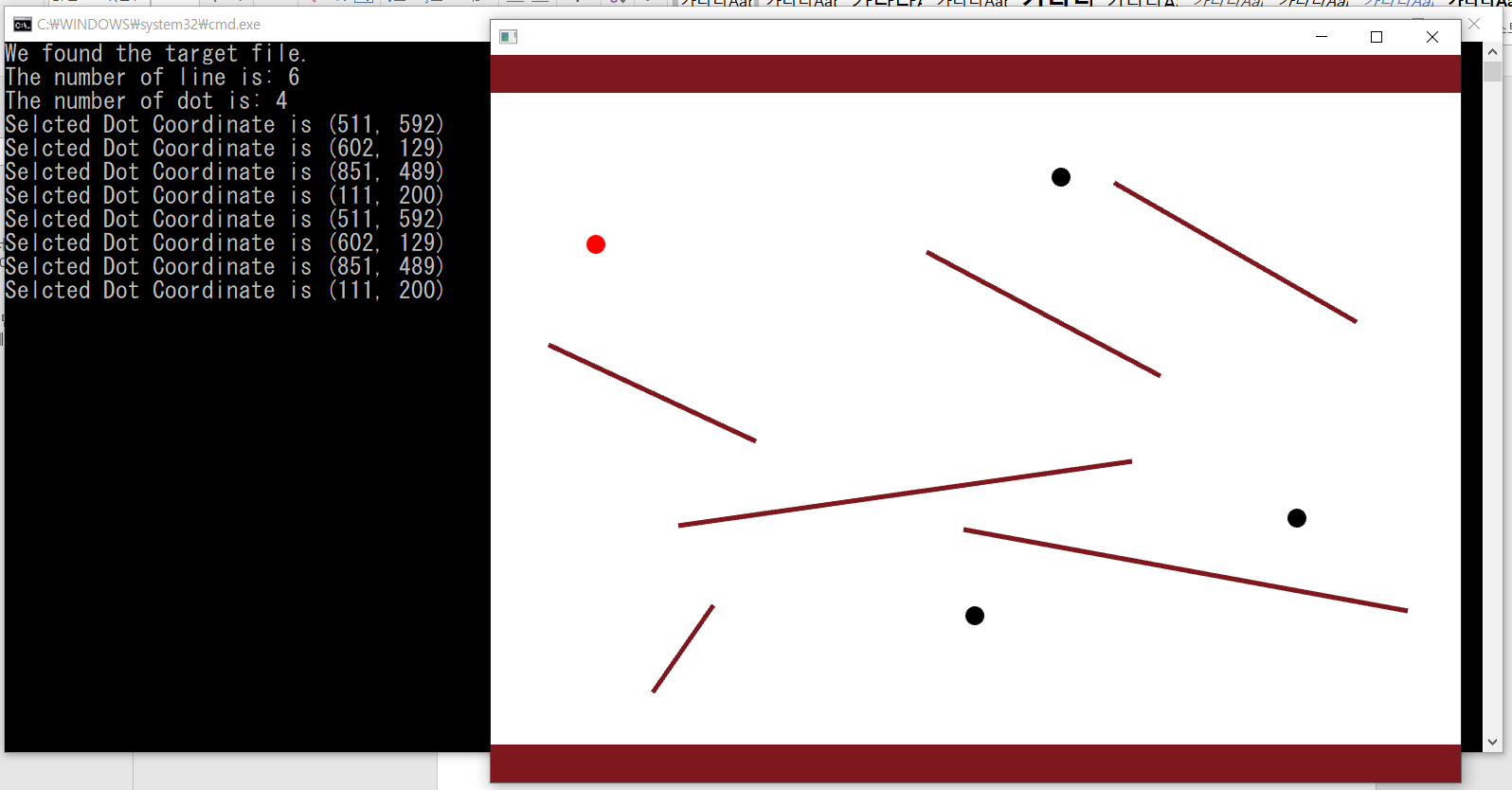
텍스트 파일을 선택하는 다이얼로그 창이 뜬다. input1.txt 파일을 선택한다.



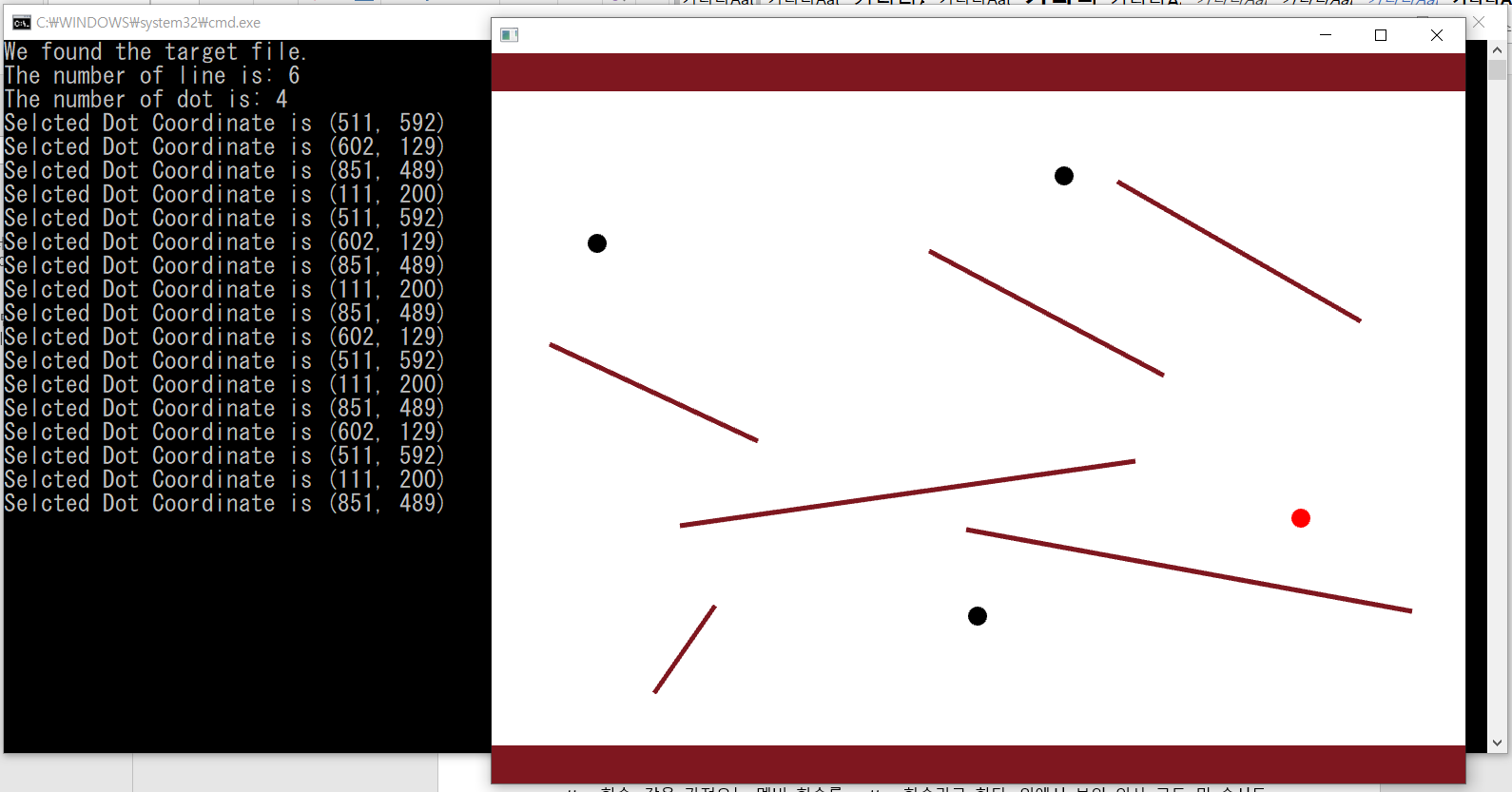
파일을 선택하면 파일을 찾았다는 메시지와 함께 선분과 점의 개수가 콘솔에 뜬다. ‘d’ 키를 누르면 위와 같이 선분과 점이 그려진다. 가장 왼쪽 점이 빨간색으로 칠해져 있다.



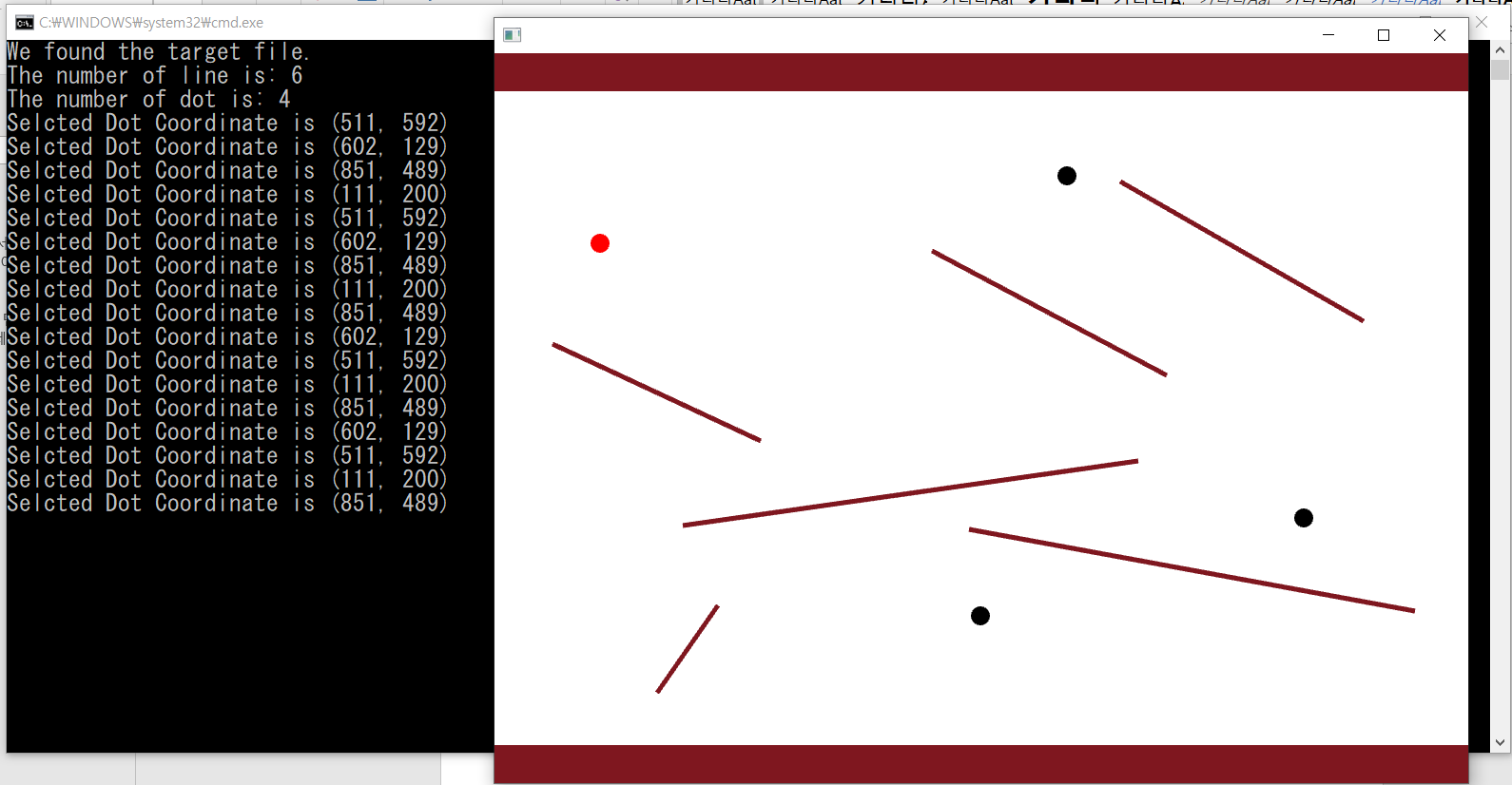
오른쪽 키를 계속 눌러서 가장 오른쪽 점으로 이동했다. 콘솔을 보면 오른쪽 키를 누를 때마다 이동한 점의 좌표를 보여준다.



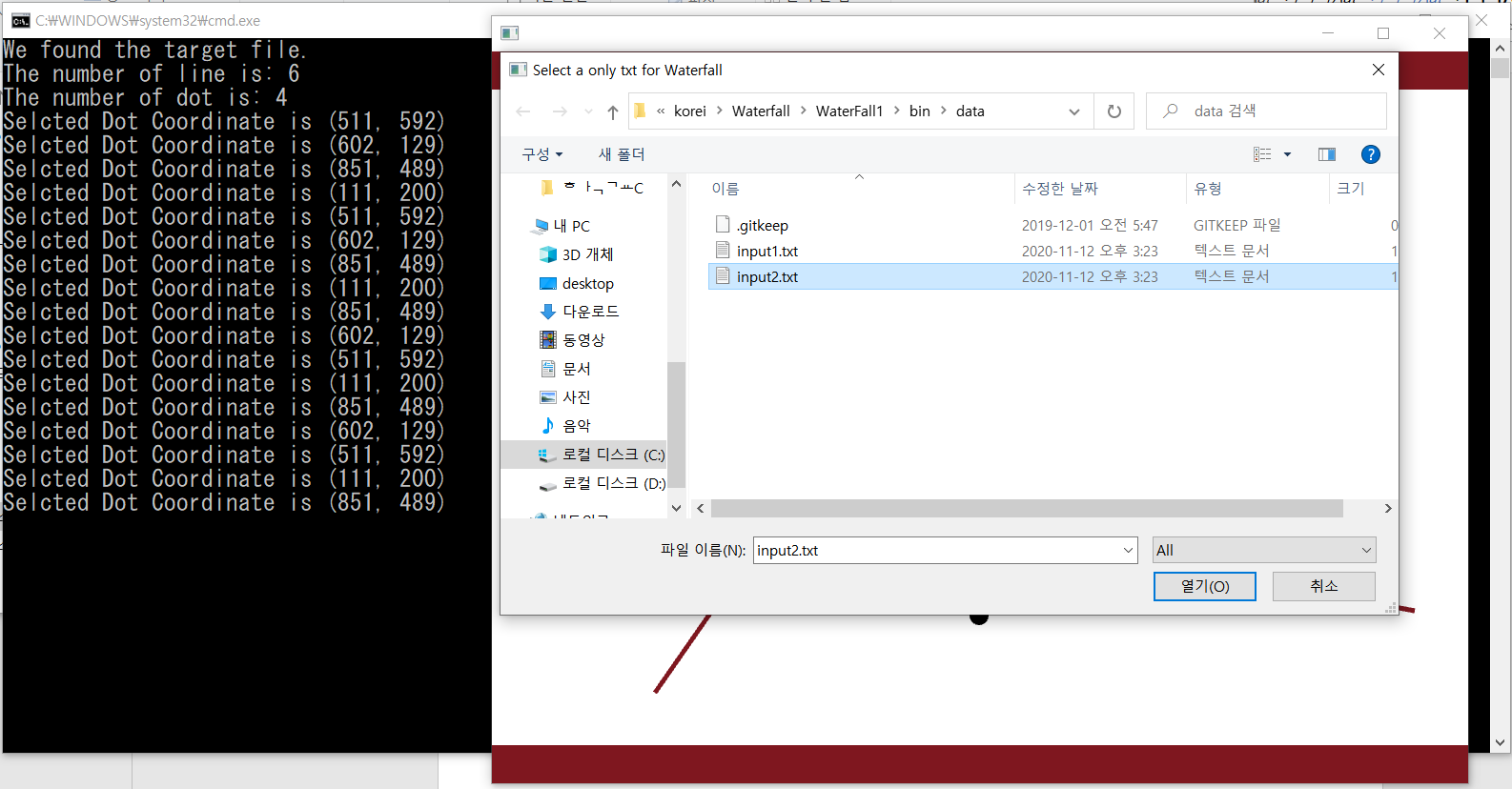
가장 오른쪽 점에 위치한 상태에서 오른쪽 방향키를 한 번 더 누르면 가장 왼쪽 점으로 오는 것을 알 수 있다.



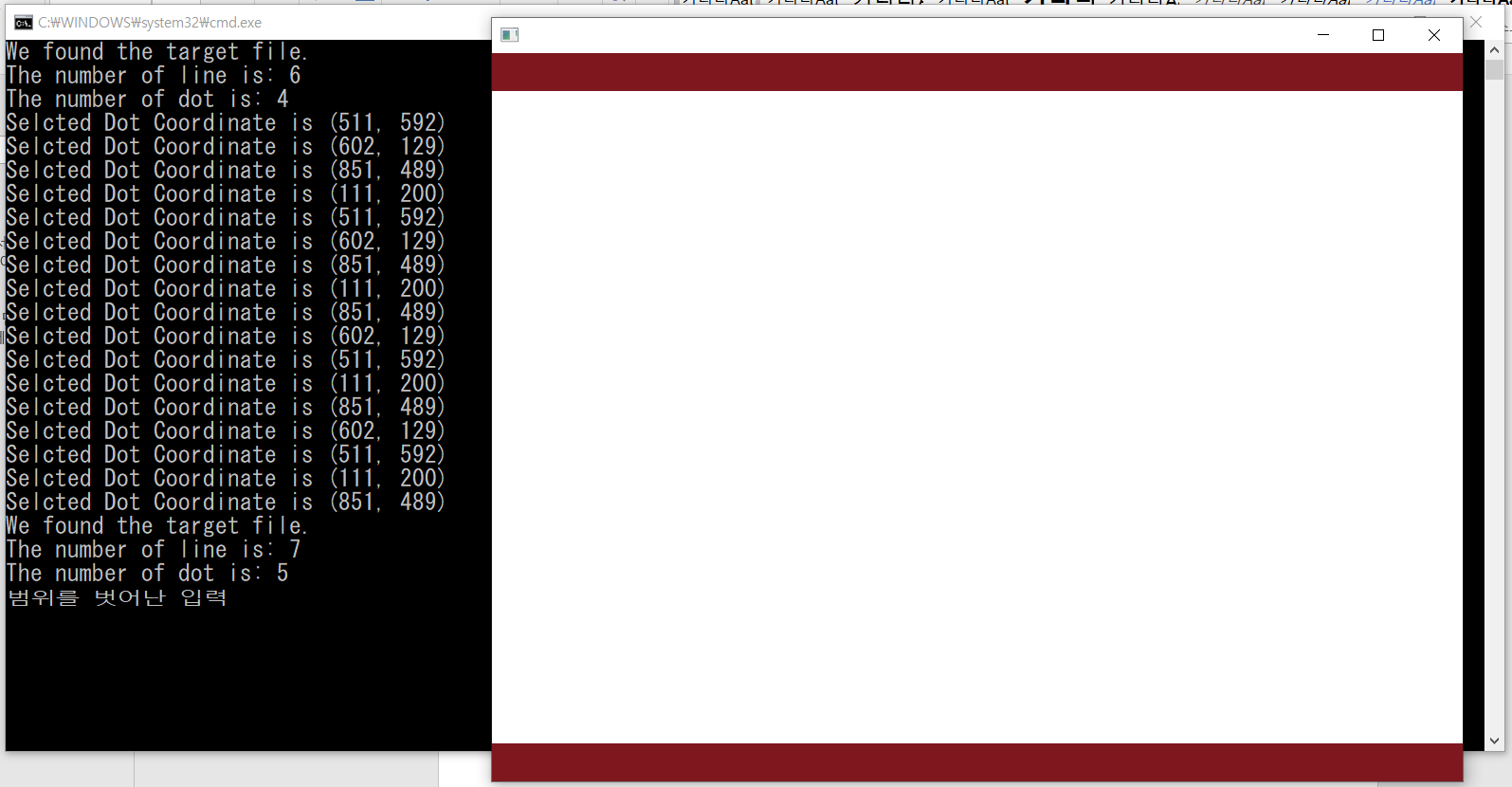
반대로 왼쪽 점에서 왼쪽 키를 누르면 가장 오른쪽 점으로 오는 것도 확인 가능하다.



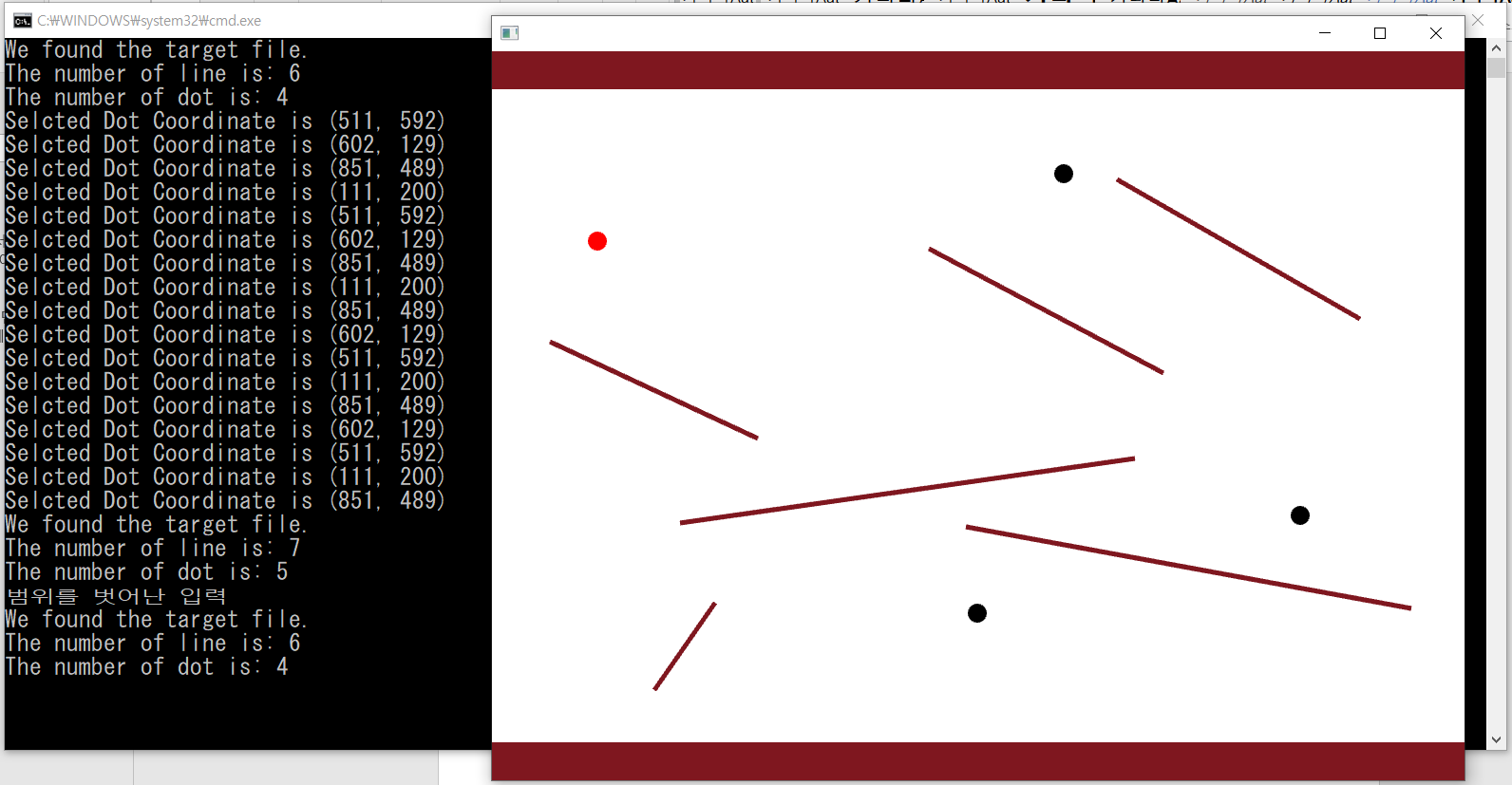
‘d’ 키를 다시 누르면 가장 왼쪽 점이 선택되는 것을 알 수 있다.



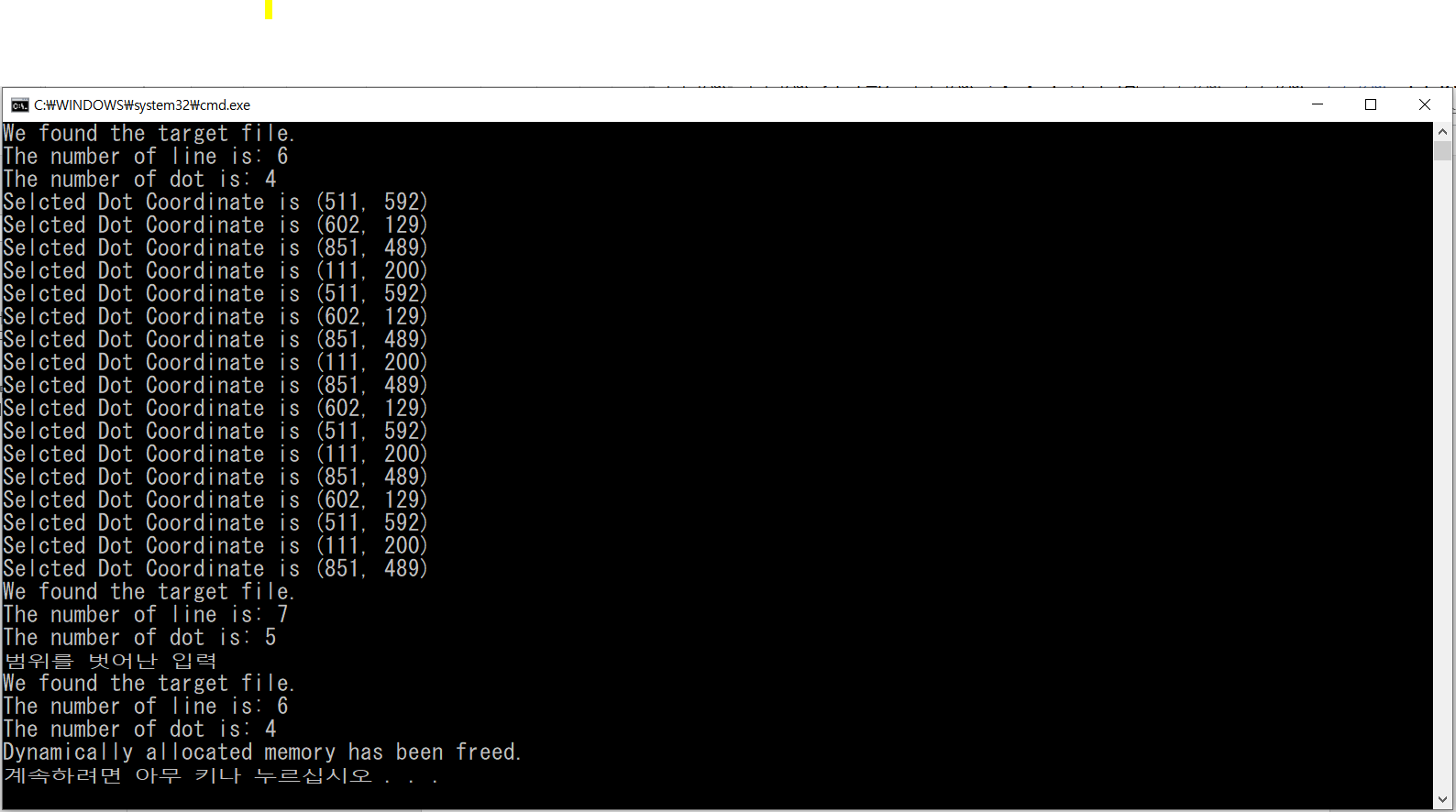
‘l’키를 누르면 파일을 선택하는 창이 다시 뜬다. 범위를 벗어난 파일 input.txt를 선택한다.



범위를 벗어났다는 메시지가 출력된다. ‘d’ 키를 눌러도, 좌우 방향키를 눌러도 어떤 동작도 하지 않는다.



다시 ‘l’ 키를 눌러 input.txt 파일을 열어 ‘d’를 누르면 정상적으로 선분과 점이 출력된다. 범위를 벗어나는 파일, 정상적인 파일의 순서에 관계없이 잘 작동한다.



‘q’를 입력하면 동적 할당된 메모리가 모두 해제되면서 정상적으로 종료된다.

기타

Waterfall 클래스의 shelf와 waterhole을 제외한 모든 멤버 변수들은 접근 제어자가 private이므로 값을 변경하거나 가져오기 위해서는 멤버 함수를 사용해야 한다. 값을 설정하는 멤버함수를 setter 함수, 값을 가져오는 멤버 함수를 getter 함수라고 한다. 위에서 보인 의사 코드 및 순서도에서는 이러한 setter, getter를 생략했다.