**오픈프레임웍스로 팩맨(Pacman) 만들기**

김수연

2021년 8월 2일

**차례**

1. 프로젝트 소개
2. 프로젝트 구현
3. 맵 그리기
4. 팩맨 구현
5. 고스트 구현
6. 게임의 진행
7. 결과
8. 마무리

**프로젝트 소개**

지난 학기, 실습 위주로 진행된 수업에서 오픈프레임웍스를 처음 접하게 되었다. 이를 활용하여 작성한 과제물은 기말고사 기간에 작성한 점, 짧은 마감기한 등의 이유로 상당히 단순하며, 함수의 활용 또한 제한적이었다. 방학을 맞이해 넉넉한 시간을 활용하여 오픈프레임웍스에 대한 이해도를 높이고, 함수들과 클래스를 조금 더 면밀히 공부하기 위해 해당 프로젝트를 진행한다.

1. 프로젝트의 목적

오픈프레임웍스를 이용하여 게임 ‘팩맨’을 만들어 본다.

1. 계획

이 게임의 구성 요소를 분석하여 크게 맵, 팩맨(플레이어), 고스트, 게임 플로우로 나누어 구현한다.

**프로젝트 구현**

1. 클래스

기본적으로 제공되는 클래스 ofApp과 새롭게 정의한 클래스 ofPacman, ofGhost 총 3개의 클래스를 사용했다. 맵을 그리는 과정에서는 ofApp, 팩맨을 구현할 때에는 ofApp과 ofPacman, 고스트를 구현할 때에는 ofGhost를 사용했다. 자세한 설명은 해당 부분을 설명할 때 기술한다.

1. 맵 그리기

맵 정보는 텍스트 파일로 저장된다. 함수 getMapInfo는 텍스트 파일의 내용을 한 줄씩 읽고 2차원 배열에 저장한다. 2차원 배열은 통로와 벽 여부를 표시하는 myMap, 도트 유무를 표시하는 food 총 2개이다.

그 후 myMap과 food의 정보를 이용하여 draw함수에서 맵을 그린다. myMap을 앞에서부터 탐색하면서 그리는데, 값이 1인요소, 즉 벽을 나타내는 칸은 고려하지 않는다. 통로를 나타내는 칸에서 상하좌우에 벽이 있는지 여부를 확인하고 벽을 그린다. 도트는 food 배열을 탐색하면서 값이 1인 부분에 그린다.

각 칸의 가로, 세로는 24이다. myMap에서 사용하는 인덱스를 (x, y)라고 할 때, 각 칸의 정 가운데의 좌표는 (24\*x+36, 24\*y+36)이다. 배경 색은 검정색, 벽은 파란색, 도트는 노란색으로 그린다.

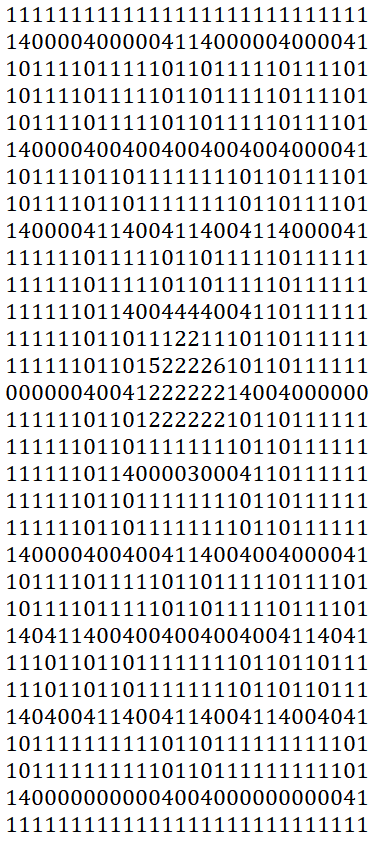
1. 클래스와 변수

ofApp 클래스에 다음과 같은 변수와 함수를 추가하였다. myMap은 전역변수로 정의하여 사용한다.

|  |  |
| --- | --- |
| ofTrueTypeFont myFont; | 폰트 |
| ofBuffer input\_buf; | 맵 정보를 읽기 위한 버퍼 |
| int row, col; | row는 맵의 세로 길이, col은 맵의 가로 길이를 저장 |
| int score\_posX, score\_posY; | 화면에 점수를 표기하기 위한 좌표 |
| int food[MAX\_ROW][MAX\_COL]; | 도트의 정보를 저장하기 위한 2차원 배열\* |
| int num\_of\_food; | 맵에 그려지는 총 도트의 개수 |
| void getMapInfo(); | 맵을 읽는 함수 |

\*MAX\_ROW와 MAX\_COL은 전역변수로, 각각 50으로 정의했다.

1. 텍스트 파일

<map01.txt>

왼쪽의 이미지에서 각 숫자가 의미하는 것은 다음과 같다:

0: 통로

1: 벽

2: 고스트 하우스 내부

3: 팩맨의 시작지점

4: 통로 중, 방향전환이 가능한 부분

고스트 하우스 입구 앞 통로는 고스트가 나올 때 방향전환이 필요하기 때문에 4로 설정하였다.

5: 빨간색 고스트의 시작지점

6: 주황색 고스트의 시작지점

1. getMapInfo() 맵 정보 읽기

|  |  |
| --- | --- |
|  | void ofApp::getMapInfo(){ |
|  | line 2 ~ 7  변수:   * line input\_buffer에 저장된 내용을 한 줄씩 받는 버퍼   맵 정보를 읽기 위한 버퍼에 메모리를 할당하고, map01.txt를 읽는다. 그 다음, line에 버퍼에 내용을 한 줄 받아 row와 col의 값을 계산한다. |
| 1  2  3  4  5  6  7 | input\_buf.allocate(930);  input\_buf = ofBufferFromFile("map/map01.txt", false);  string line = new char[30];  line = input\_buf.getFirstLine();  col = line.length();  row = input\_buf.size() / (col + 1); |
|  | line 9 ~ 14:  line에 저장되어 있는 첫째 줄의 내용을 읽고 myMap에 정보를 저장한다. 첫째줄은 모두 ‘1’ 아니면 ‘0’ 이므로 두 가지 경우에 대해서만 처리한다. map01.txt의 경우 ‘1’만 있으나, 다른 맵을 사용할 경우를 고려해서 ‘0’도 처리한다. |
| 8  9  10  11  12  13  14 | for (int i = 0; i < col; i++) {  if (line[i] == '1')  myMap[0][i] = 1;  else if (line[i] == '0')  myMap[0][i] = 0;  } |
|  | line 16 ~ 54:  첫째줄 이후의 내용을 한줄 씩 읽고 myMap과 food에 정보를 저장한다. myMap에는 line과 동일한 숫자를 저장한다. line이 ‘3’일 경우, 해당 좌표의 인덱스를 팩맨의 처음위치로 저장한다. 마찬가지로 ‘5’일 경우 빨간 고스트의 초기 좌표, ‘6’인 경우 주황 고스트의 초기 좌표로 저장한다.  food는 통로일 경우에만 1로, 그렇지 않은 경우 2를 저장한다. 1로 저장한 경우, num\_of\_food를 1 증가시킨다. |
| 15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | for (int i = 1; i < row; i++) {  line = input\_buf.getNextLine();  for (int j = 0; j < col; j++) {  if (line[j] == '1') {  myMap[i][j] = 1;  food[i][j] = 2;  }  else if (line[j] == '0') {  myMap[i][j] = 0;  food[i][j] = 1;  num\_of\_food++;  }  else if (line[j] == '2') {  myMap[i][j] = 2;  food[i][j] = 2;  }  else if (line[j] == '3') {  myMap[i][j] = 3;  pacman.posX\_init = j; pacman.posY\_init = i;  food[i][j] = 2;  }  else if (line[j] == '4') {  myMap[i][j] = 4;  food[i][j] = 1;  num\_of\_food++;  }  else if (line[j] == '5') {  myMap[i][j] = 5;  red.posX\_init = j; red.posY\_init = i;  food[i][j] = 2;  }  else if (line[j] == '6') {  myMap[i][j] = 6;  orange.posX\_init = j; orange.posY\_init = i;  food[i][j] = 2;  }  }  } |
|  | line 56 ~ 57:  점수를 표기하기 위한 좌표를 설정한다. |
| 55  56  57 | score\_posX = 36;  score\_posY = 24 \* row + 64; |
|  | } |

1. draw() 맵 그리기

|  |  |
| --- | --- |
|  | line 2 ~ 4:  myMap를 탐색하는데, 1인 칸은 고려하지 않는다. |
| 1  2  3  4  5 | // draw map  for (int i = 0; i < row; i++) {  for (int j = 0; j < col; j++) {  if (myMap[i][j] != 1) { |
|  | line 6 ~ 9:  food를 탐색하면서 해당하는 칸의 정 가운데에 도트를 그린다. 색은 노란색이다. |
| 6  7  8  9 | if (food[i][j] == 1) {  ofSetColor(255, 255, 0);  ofDrawCircle(12 + 24 \* (j + 1), 12 + 24 \* (i + 1), 2);  } |
|  | line 10 ~ 24:  각 칸의 상하좌우의 값을 확인하고 벽을 그린다. |
| 10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | ofSetColor(0, 0, 255);  ofSetLineWidth(5);  if (myMap[i - 1][j] == 1) { // wall at up  ofDrawLine(24 \* (j + 1), 24 \* (i + 1), 24 \* (j + 2), 24 \* (i + 1));  }  if (myMap[i][j - 1] == 1) { // wall at left  ofDrawLine(24 \* (j + 1), 24 \* (i + 1), 24 \* (j + 1), 24 \* (i + 2));  }  if (myMap[i][j + 1] == 1) { // wall at right  ofDrawLine(24 \* (j + 2), 24 \* (i + 1), 24 \* (j + 2), 24 \* (i + 2));  }  if (myMap[i + 1][j] == 1) { // wall at down  ofDrawLine(24 \* (j + 1), 24 \* (i + 2), 24 \* (j + 2), 24 \* (i + 2));  }  }  }  } |

1. 팩맨 구현하기

클래스 pacman을 선언하였다. 사용자의 입력에 따라 방향을 설정하기 위해 두 가지 변수 currDir, nextDir을 사용한다. currDir은 현재 이동하는 방향을 나타낸다. currDir이 0이라면 정지상태이다. 함수 keypressed()에서 사용자로부터 방향을 입력받고, 함수 movePacman()에서 currDir의 값에 따라 팩맨의 좌표를 바꾼다.

팩맨이 위치한 곳의 map값이 4이고, 팩맨의 좌표가 해당 칸의 정 가운데가 되면 방향을 바꾼다. 먼저 진행 중인 방향으로 계속 이동이 가능한지 확인한다. 앞이 막혀있다면 currDir를 0으로 설정한다. nextDir이 0이 아닐 때, nextDir의 방향으로 이동이 가능한 지 확인한 후 currDir에 nextDir의 값을 저장하고 nextDir을 0으로 설정한다.

eatFood() 함수에서는 팩맨의 위치의 좌표 정보로 food를 탐색하고, 해당 위치의 food값이 1이면 0으로 바꾼 후 점수를 1 증가시킨다.

draw() 함수에서 팩맨의 방향과 이미지의 인덱스를 고려하여 그린다.

상하좌우 4 방향은 각각 1부터 4까지의 숫자로 정의한다 (위 = 1, 오른쪽 = 2, 아래 = 3, 왼쪽 = 4).

1. 클래스와 변수

ofPacman 클래스를 사용한다. ofPacman의 변수 및 함수는 다음과 같다.

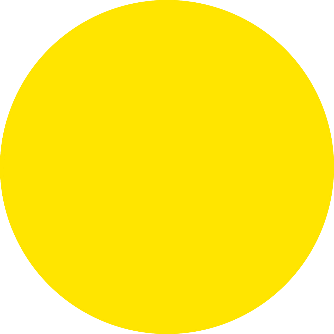
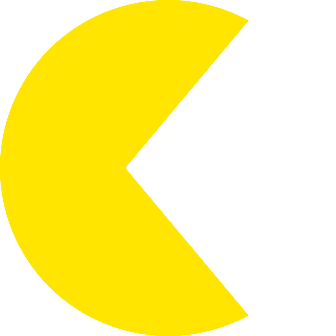
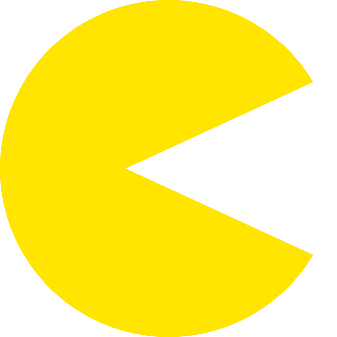
|  |  |
| --- | --- |
| ofPacman(); | 생성자 |
| void init(); | 팩맨의 변수와 이미지를 초기화하는 함수 |
| int posX\_init, posY\_init; | 팩맨의 초기 위치의 인덱스를 저장 |
| int posX, posY; | 현재 팩맨의 위치의 좌표를 저장 |
| int posX\_idx, posY\_idx; | 현재 팩맨의 위치의 인덱스를 저장 |
| int score; | 점수 |
| float pacmanPic\_idx; | 팩맨 이미지의 인덱스 |
| ofImage pacmanPic[4]; | 팩맨 이미지를 저장하지 위한 변수 |

ofApp에 추가한 변수와 함수이다.

|  |  |
| --- | --- |
| ofPacman pacman; | 팩맨 |
| int currDir, nextDir; | 팩맨의 방향을 나타내는 변수 |
| void movePacman(); | 팩맨의 위치를 바꾸는 함수 |
| void eatFood(); | 도트를 먹는 기능을 수행하는 함수 |

1. 팩맨 이미지

가운데 그림은 입을 열고 닫는 두가지 과정 모두에서 사용된다. 인덱스 0부터 3까지 사용하여 총 4개의 이미지를 저장한다.



1. init() 팩맨 정보 초기화

|  |  |
| --- | --- |
|  | void ofPacman::init() { |
|  | line 2 ~ 5:  팩맨의 현재 위치를 맵에 표시된 초기 위치로 설정한다.  line 7 ~ 8:  점수와 팩맨의 이미지 인덱스를 0으로 설정한다.  line 10 ~ 20:  이미지를 각 인덱스에 맞게 변수에 저장한다. |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | posX = 24 \* posX\_init + 36;  posY = 24 \* posY\_init + 36;  posX\_idx = posX\_init;  posY\_idx = posY\_init;  score = 0;  pacmanPic\_idx = 0;  pacmanPic[0].loadImage("images/pacman\_1.png");  pacmanPic[0].setAnchorPercent(0.5, 0.5);  pacmanPic[1].loadImage("images/pacman\_2.png");  pacmanPic[1].setAnchorPercent(0.5, 0.5);  pacmanPic[2].loadImage("images/pacman\_3.png");  pacmanPic[2].setAnchorPercent(0.5, 0.5);  pacmanPic[3].loadImage("images/pacman\_4.png");  pacmanPic[3].setAnchorPercent(0.5, 0.5); |
|  | } |

1. keyPressed() 사용자로부터 입력 받기

|  |  |
| --- | --- |
|  | void ofApp::keyPressed(int key){ |
|  | line 2 ~ 4:  사용자가 esc를 누르면 게임을 종료한다.  line 6 ~ 28:  팩맨이 움직이는 도중에 사용자가 방향키를 누르면, 먼저 현재 움직이는 방향을 확인한다. 현재 방향(currDir)과 새로 입력된 방향이 반대방향이라면 즉시 현재 방향을 입력된 방향으로 바꾼다. 만약 반대방향이 아니라면 입력된 방향을 nextDir에 저장한다. |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | if (key == OF\_KEY\_ESC) {  ofExit();  }  if (currDir != 0) { // pacman is moving  if (key == OF\_KEY\_UP) {  if (currDir == 3)  currDir = 1;  else nextDir = UP;  }  else if (key == OF\_KEY\_RIGHT) {  if (currDir == 4)  currDir = 2;  else nextDir = RIGHT;  }  else if (key == OF\_KEY\_DOWN) {  if (currDir == 1)  currDir = 3;  else nextDir = DOWN;  }  else if (key == OF\_KEY\_LEFT) {  if (currDir == 2)  currDir = 4;  else nextDir = LEFT;  }  } |
|  | line 29 ~ 46:  팩맨이 현재 움직이고 있지 않을 경우, 사용자가 입력한 방향으로 이동이 가능한지 확인한다. 가능하다면 currDir에 입력된 방향을 저장한다. |
| 29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | else if (currDir == 0) { // pacman is not moving  if (key == OF\_KEY\_UP) {  if(myMap[pacman.posY\_idx - 1][pacman.posX\_idx] == 0 || myMap[pacman.posY\_idx - 1][pacman.posX\_idx] == 4)  currDir = UP;  }  else if (key == OF\_KEY\_RIGHT) {  if (myMap[pacman.posY\_idx][pacman.posX\_idx + 1] == 0 || myMap[pacman.posY\_idx][pacman.posX\_idx + 1] == 4)  currDir = RIGHT;  }  else if (key == OF\_KEY\_DOWN) {  if (myMap[pacman.posY\_idx + 1][pacman.posX\_idx] == 0 || myMap[pacman.posY\_idx + 1][pacman.posX\_idx] == 4)  currDir = DOWN;  }  else if (key == OF\_KEY\_LEFT) {  if (myMap[pacman.posY\_idx][pacman.posX\_idx - 1] == 0 || myMap[pacman.posY\_idx][pacman.posX\_idx - 1] == 4)  currDir = LEFT;  }  } |
|  | } |

1. movePacman() 팩맨 이동시키기

|  |  |
| --- | --- |
|  | void ofApp::movePacman() { |
|  | line 1:  팩맨은 currDir이 0이 아닐 때에만 이동한다.  line 3 ~ 7:  팩맨의 이미지를 순서대로 출력하도록 한다. 각 프레임마다 인덱스를 0.2씩 증가시키고, 범위를 벗어나면 다시 0으로 설정한다. |
| 1  2  3  4  5  6  7 | if (currDir != 0) {  pacman.pacmanPic\_idx += 0.2;  if (pacman.pacmanPic\_idx > 3) {  pacman.pacmanPic\_idx = 0;  } |
|  | line 8 ~ 23:  currDir에 따라 팩맨의 좌표를 1씩 증감하고, 바뀐 좌표에 따라 위치 인덱스를 변화시킨다. |
| 8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | if (currDir == 1) { // UP  pacman.posY--;  pacman.posY\_idx = pacman.posY / 24 - 1;  }  else if (currDir == 2) { // RIGHT  pacman.posX++;  pacman.posX\_idx = pacman.posX / 24 - 1;  }  else if (currDir == 3) { // DOWN  pacman.posY++;  pacman.posY\_idx = pacman.posY / 24 - 1;  }  else if (currDir == 4) { // LEFT  pacman.posX--;  pacman.posX\_idx = pacman.posX / 24 - 1;  } |
|  | line 25 ~ 32:  팩맨이 맵의 끝에 다다르면 반대쪽 끝으로 이동시킨다. |
| 24  25  26  27  28  29  30  31  32 | if (pacman.posX\_idx == col) {  pacman.posX\_idx = 0;  pacman.posX = 24 \* pacman.posX\_idx + 36;  }  if (pacman.posX\_idx == -1) {  pacman.posX\_idx = col - 1;  pacman.posX = 24 \* pacman.posX\_idx + 36;  } |
|  | line 34 ~ 51:  팩맨이 맵 값이 4인 칸의 정 가운데에 도착하면 방향을 바꿀 수 있다. 해당 위치에 도달하면 먼저 진행중인 방향으로 계속 갈 수 있는지 먼저 확인한다. 갈 수 없다면 currDir를 0으로 설정한다. |
| 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | if (myMap[pacman.posY\_idx][pacman.posX\_idx] == 4 && (pacman.posX == 24 \* pacman.posX\_idx + 36 && pacman.posY == 24 \* pacman.posY\_idx + 36)) {  if (currDir == 1) { // UP  if (myMap[pacman.posY\_idx - 1][pacman.posX\_idx] == 1 || myMap[pacman.posY\_idx - 1][pacman.posX\_idx] == 2)  currDir = 0;  }  else if (currDir == 2) { // RIGHT  if (myMap[pacman.posY\_idx][pacman.posX\_idx + 1] == 1 || myMap[pacman.posY\_idx][pacman.posX\_idx + 1] == 2)  currDir = 0;  }  else if (currDir == 3) { // DOWN  if (myMap[pacman.posY\_idx + 1][pacman.posX\_idx] == 1 || myMap[pacman.posY\_idx + 1][pacman.posX\_idx] == 2)  currDir = 0;  }  else if (currDir == 4) { // LEFT  if (myMap[pacman.posY\_idx][pacman.posX\_idx - 1] == 1 || myMap[pacman.posY\_idx][pacman.posX\_idx - 1] == 2)  currDir = 0;  } |
|  | line 53 ~ 75:  nextDir이 0이면 팩맨은 정지하거나 진행 중인 방향으로 직진한다. nextDir이 0이 아니면 해당 방향으로 진행 할 수 있는지 확인한다. nextDir의 방향으로 진행이 가능하면 currDir에 nextDir의 값을 저장하고 nextDir에 0을 저장한다. |
| 52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81 | if (nextDir != 0) {  if (nextDir == 1) { // UP  if (myMap[pacman.posY\_idx - 1][pacman.posX\_idx] == 0 || myMap[pacman.posY\_idx - 1][pacman.posX\_idx] == 4) {  currDir = 1;  nextDir = 0;  }  }  else if (nextDir == 2) { // RIGHT  if (myMap[pacman.posY\_idx][pacman.posX\_idx + 1] == 0 || myMap[pacman.posY\_idx][pacman.posX\_idx + 1] == 4) {  currDir = 2;  nextDir = 0;  }  }  else if (nextDir == 3) { // DOWN  if (myMap[pacman.posY\_idx + 1][pacman.posX\_idx] == 0 || myMap[pacman.posY\_idx + 1][pacman.posX\_idx] == 4) {  currDir = 3;  nextDir = 0;  }  }  else if (nextDir == 4) { // LEFT  if (myMap[pacman.posY\_idx][pacman.posX\_idx - 1] == 0 || myMap[pacman.posY\_idx][pacman.posX\_idx - 1] == 4) {  currDir = 4;  nextDir = 0;  }  }  }  }  } |
|  | } |

1. eatFood() 도트 먹기

|  |  |
| --- | --- |
|  | void ofApp::eatFood() { |
|  | 팩맨이 위치한 칸의 food값이 1이면 0으로 바꾸고, 점수에 1을 더한다. |
| 1  2  3  4 | if (food[pacman.posY\_idx][pacman.posX\_idx] == 1) {  food[pacman.posY\_idx][pacman.posX\_idx] = 0;  pacman.score++;  } |
|  | } |

1. draw() 팩맨 그리기

|  |  |
| --- | --- |
|  | 팩맨의 위치를 기준으로 진행 중인 방향에 맞게 회전시킨 후 팩맨을 그린다. |
| 1  2  3  4  5  6 | // pacman  ofPushMatrix();  ofTranslate(pacman.posX, pacman.posY);  ofRotate(currDir \* 90 - 180);  pacman.pacmanPic[(int)pacman.pacmanPic\_idx].draw(0, 0, 15, 15);  ofPopMatrix(); |

1. 고스트 구현하기

고스트의 이동은 크게 시작할 때 고스트 하우스 내부에서의 이동과 외부에서의 이동으로 나누었다. 처음 시작할 때 고스트 하우스 안에서 밖으로 나가는 과정은 함수 goingOut()에서 담당하고, 그 외의 경우는 함수 move()가 담당한다. 고스트가 이동하는 방향은 변수 currDir에 저장한다.

고스트 하우스 내부에서 고스트는 오른쪽 시계방향으로 벽을 따라 이동하다가 출구를 찾으면 밖으로 나간다.

고스트 하우스 외부에서는 팩맨의 위치에 따라 방향을 전환하며 이동한다. 방향을 바꿀 수 있는 위치, 즉 맵 값이 4인 칸의 가운데에 도달하면 팩맨의 위치를 변수 targetX와 targetY에 저장한다. 현재 진행 중인 방향을 기준으로, 타깃이 오른쪽에 있으면 오른쪽으로, 왼쪽에 있으면 왼쪽으로 방향을 바꾸고, 앞에 있으면 직진한다. 만일 해당 방향에 길이 없다면, 본래 진행하던 방향으로 가도록 하고, 직진도 할 수 없다면 위, 오른쪽, 아래, 왼쪽 순서대로 탐색하며 진행 가능한 방향으로 전환한다. 고스트는 왔던 길로 되돌아갈 수 없다. 방향을 바꾼 후에는 currDir의 값을 수정한다.

고스트를 그릴 때는 방향을 고려하여 ghostPic의 인덱스를 바꾸어서 그린다.

1. 클래스와 변수

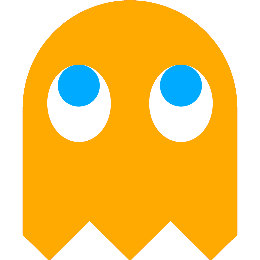
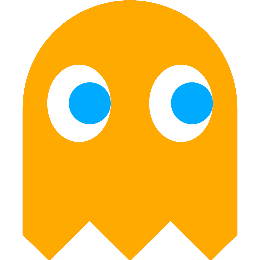
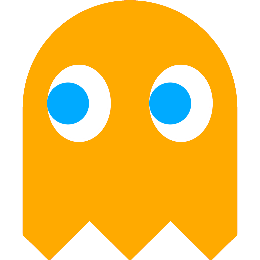
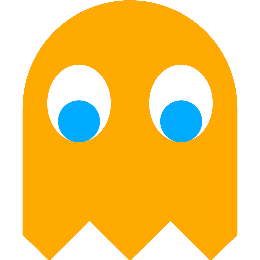
클래스 ofGhost를 선언하였다.

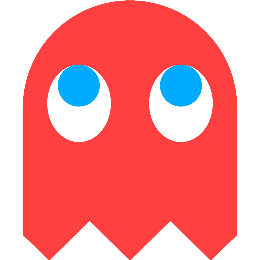
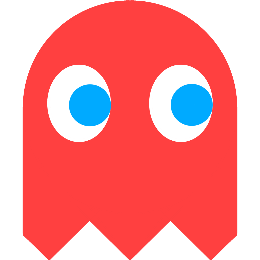
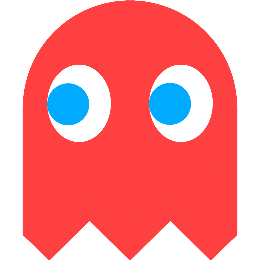
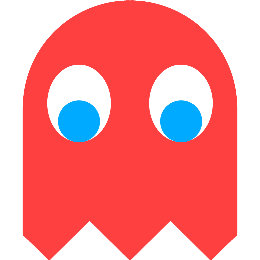
|  |  |
| --- | --- |
| ofGhost(); | 생성자 |
| void init(); | 변수들을 초기화하고 이미지를 불러오는 함수 |
| void move(int myMap[][50],int,int); | 고스트를 이동시키는 함수 |
| void goingOut(int myMap[][50]); | 고스트 하우스 안에서 밖으로 나가기 위한 함수 |
| string name; | 고스트의 이름 |
| int findOut; | goingOut()에서 나가는 길을 찾았는지 여부 |
| int targetX, targetY; | 방향전환 시 팩맨의 위치 |
| int posX\_init, posY\_init; | 맵에 표시된 초기위치 |
| int posX, posY; | 현재 위치의 좌표 |
| int posX\_idx, posY\_idx; | 현재 위치의 인덱스 |
| int currDir; | 고스트가 진행 중인 방향 |
| ofImage ghostPic[4]; | 고스트의 이미지 |

ofApp에 다음의 변수를 추가하였다.

|  |  |
| --- | --- |
| ofGhost red; | 빨간색 고스트 |
| ofGhost orange; | 주황색 고스트 |

1. 고스트 이미지





1. goingOut() 고스트 하우스 밖으로 나가기

|  |  |
| --- | --- |
|  | void ofGhost::goingOut(int myMap[][50]) { |
|  | line 2:  나가는 길을 찾지 못했음을 확인한다.  line 3 ~ 67:  고스트가 오른쪽으로 진행 중일 때에는 위로 나가는 길이 있는지 확인하면서 진행한다. 고스트가 위로 이동할 수 있고, 현재 칸의 정 가운데에 있을 때 currDir를 1(위)로 바꾸고 findOut을 1로 설정한다. 그 외의 경우, 오른쪽으로 계속 진행할 수 없다면 방향을 아래로 3(아래)로 바꾸고, 진행할 수 있다면 x좌표 값을 1 증가시킨다.  위 과정을 모든 방향에 적용한다. 나가는 길을 찾는 방향은 오른쪽 진행시 위, 아래쪽 진행시 오른쪽, 왼쪽으로 진행시 아래, 위로 진행시 왼쪽이다. 즉, 고스트 하우스의 바깥쪽으로 탐색한다. |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67 | if (!findOut) {  if (currDir == 2) { // RIGHT  if (myMap[posY\_idx - 1][posX\_idx] != 1 && posX == 24 \* posX\_idx + 36) {  currDir = 1;  findOut = 1;  posX--;  }  else {  if (myMap[posY\_idx][posX\_idx + 1] == 1 && posX == 24 \* posX\_idx + 36) {  currDir = 3;  }  else {  posX++;  posX\_idx = posX / 24 - 1;  }  }  }  else if (currDir == 3) { // DOWN  if (myMap[posY\_idx][posX\_idx + 1] != 1 && posY == 24 \* posY\_idx + 36) {  currDir = 2;  findOut = 1;  posY--;  }  else {  if (myMap[posY\_idx + 1][posX\_idx] == 1 && posY == 24 \* posY\_idx + 36) {  currDir = 4;  }  else {  posY++;  posY\_idx = posY / 24 - 1;  }  }  }  else if (currDir == 4) { // LEFT  if (myMap[posY\_idx + 1][posX\_idx] != 1 && posX == 24 \* posX\_idx + 36) {  currDir = 3;  findOut = 1;  posX++;  }  else {  if (myMap[posY\_idx][posX\_idx - 1] == 1 && posX == 24 \* posX\_idx + 36) {  currDir = 1;  }  else {  posX--;  posX\_idx = posX / 24 - 1;  }  }  }  else if (currDir == 1) { // UP  if (myMap[posY\_idx][posX\_idx - 1] != 1 && posY == 24 \* posY\_idx + 36) {  currDir = 4;  findOut = 1;  posY++;  }  else {  if (myMap[posY\_idx - 1][posX\_idx] == 1 && posY == 24 \* posY\_idx + 36) {  currDir = 2;  }  else {  posY--;  posY\_idx = posY / 24 - 1;  }  }  }  } |
|  | line 68 ~ 87:  findOut이 1이면 나가는 길로 진행한다. 이 부분은 고스트가 위치한 맵의 값이 0이 아닐 때, 즉 통로로 진입하기 전까지 실행된다. currDir에 따라 고스트의 좌표를 바꾼다. |
| 68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87 | else{  if (myMap[posY\_idx][posX\_idx] != 0) {  if (currDir = 1) { // UP  posY--;  posY\_idx = posY / 24 - 1;  }  else if (currDir = 2) { // RIGHT  posX++;  posX\_idx = posX / 24 - 1;  }  else if (currDir = 3) { // DOWN  posY++;  posY\_idx = posY / 24 - 1;  }  else if (currDir = 4) { // LEFT  posX--;  posX\_idx = posX / 24 - 1;  }  }  } |
|  | } |

1. move() 고스트 이동하기

|  |  |
| --- | --- |
|  | void ofGhost::move(int myMap[][50], int posX\_of\_pacman,  int posY\_of\_pacman) { |
|  | line 2 ~ 10:  배열 direction을 초기화하고, 고스트의 이름에 따라 target을 설정한다. 빨간색 고스트는 팩맨의 좌표를 그대로 target으로 정하고, 주황색 고스트는 팩맨의 x좌표와 y좌표에 5씩 더한 값을 target으로 정한다. direction은 고스트가 해당 칸에서 각 방향으로 진행 가능한지 여부를 저장한다. |
| 1  2  3  4  5  6  7  8  9  10 | int direction[4] = { 0 };  if (name == "red") {  targetX = posX\_of\_pacman;  targetY = posY\_of\_pacman;  }  else if (name == "orange") {  targetX = posX\_of\_pacman + 5;  targetY = posY\_of\_pacman + 5;  } |
|  | line 12 ~ 30:  고스트의 위치가 통로일 때(4가 아닐 때)와 통로가 아니지만 칸 가운데에 위치할 때에는 currDir에 따라 이동한다. |
| 11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | if (myMap[posY\_idx][posX\_idx] != 4 || (posX == 24 \* posX\_idx + 36 && posY == 24 \* posY\_idx + 36)) {  if (currDir == 1) { // UP  posY--;  posY\_idx = posY / 24 - 1;  }  else if (currDir == 2) { // RIGHT  posX++;  posX\_idx = posX / 24 - 1;  }  else if (currDir == 3) { // DOWN  posY++;  posY\_idx = posY / 24 - 1;  }  else if (currDir == 4) { // LEFT  posX--;  posX\_idx = posX / 24 - 1;  }  } |
|  | line 31 ~ 204:  고스트가 4인 칸 내에 있지만 가운데에 위치하지는 않을 때, 이동중인 방향에 따라 좌표를 1씩 증감한다. 이때 바뀐 좌표가 칸의 가운데라면 방향을 바꾼다. target이 고스트의 오른쪽에 있으면 우회전, 왼쪽에 있으면 좌회전, 앞에 있으면 직진한다. 이 때 해당 방향으로 이동이 가능한지 먼저 판단하고, 그 여부를 direction에 저장한다. target이 해당 방향에 있다면 바로 currDir를 바꾸고 레이블 changed로 이동한다. 만일 target이 있는 방향으로 이동할 수 없다면 직진하고, 직진도 할 수 없다면 direction을 탐색하면서 갈 수 있는 방향으로 currDir를 바꾼다. currDir를 바꾼 후에는 changed로 이동한다. |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204 | else if (myMap[posY\_idx][posX\_idx] == 4) {  if (currDir == 1) { // UP  posY--;  posY\_idx = posY / 24 - 1;  if(posY == 24 \* posY\_idx + 36) {  // change direction  if (myMap[posY\_idx][posX\_idx + 1] == 0 || myMap[posY\_idx][posX\_idx + 1] == 4) { // target is on right side  direction[1] = 1;  if (posX\_idx < targetX) {  currDir = 2; // turn right  goto changed;  }  }  if (myMap[posY\_idx - 1][posX\_idx] == 0 || myMap[posY\_idx - 1][posX\_idx] == 4) { // target is at front  direction[0] = 1;  if (posX\_idx == targetX) {  currDir = 1; // go straight  goto changed;  }  }  if (myMap[posY\_idx][posX\_idx - 1] == 0 || myMap[posY\_idx][posX\_idx - 1] == 4) { // target is on left side  direction[3] = 1;  if (posX\_idx >targetX) {  currDir = 4; // turn left  goto changed;  }  }  if (direction[0] == 1) {  currDir = 1;  goto changed;  }    for (int i = 0; i < 4; i++) {  if (direction[i] == 1) {  currDir = i + 1;  break;  }  }  goto changed;  }  }  else if (currDir == 2) { // RIGHT  posX++;  posX\_idx = posX / 24 - 1;  if (posX == 24 \* posX\_idx + 36) {  // change direction  if (myMap[posY\_idx - 1][posX\_idx] == 0 || myMap[posY\_idx - 1][posX\_idx] == 4) { // target is higher  direction[0] = 1;  if (posY\_idx > targetY) {  currDir = 1; // go up  goto changed;  }  }  if (myMap[posY\_idx][posX\_idx + 1] == 0 || myMap[posY\_idx][posX\_idx + 1] == 4) { // target is at front  direction[1] = 1;  if (posY\_idx == targetY) {  currDir = 2; // go straight  goto changed;  }  }  if (myMap[posY\_idx + 1][posX\_idx] == 0 || myMap[posY\_idx + 1][posX\_idx] == 4) { // target is lower  direction[2] = 1;  if (posY\_idx <targetY) {  currDir = 3; // go down  goto changed;  }  }  if (direction[1] == 1) {  currDir = 2;  goto changed;  }  for (int i = 0; i < 4; i++) {  if (direction[i] == 1) {  currDir = i + 1;  break;  }  }  goto changed;  }  }  else if (currDir == 3) { // DOWN  posY++;  posY\_idx = posY / 24 - 1;  if (posY == 24 \* posY\_idx + 36) {  // change direction  if (myMap[posY\_idx][posX\_idx + 1] == 0 || myMap[posY\_idx][posX\_idx + 1] == 4) { // target is on right side  direction[1] = 1;  if (posX\_idx <targetX) {  currDir = 2; // turn right  goto changed;  }  }  if (myMap[posY\_idx + 1][posX\_idx] == 0 || myMap[posY\_idx + 1][posX\_idx] == 4) { // target is at front  direction[2] = 1;  if (posX\_idx == targetX) {  currDir = 3; // go straight  goto changed;  }  }  if (myMap[posY\_idx][posX\_idx - 1] == 0 || myMap[posY\_idx][posX\_idx - 1] == 4) { // target is on left side  direction[3] = 1;  if (posX\_idx >targetX) {  currDir = 4; // turn left  goto changed;  }  }  if (direction[2] == 1) {  currDir = 3;  goto changed;  }  for (int i = 0; i < 4; i++) {  if (direction[i] == 1) {  currDir = i + 1;  break;  }  }  goto changed;  }  }  else if (currDir == 4) { // LEFT  posX--;  posX\_idx = posX / 24 - 1;  if (posX == 24 \* posX\_idx + 36) {  // change direction  if (myMap[posY\_idx - 1][posX\_idx] == 0 || myMap[posY\_idx - 1][posX\_idx] == 4) { // target is higher  direction[0] = 1;  if (posY\_idx >targetY) {  currDir = 1; // go up  goto changed;  }  }  if (myMap[posY\_idx][posX\_idx - 1] == 0 || myMap[posY\_idx][posX\_idx - 1] == 4) { // target is at front  direction[3] = 1;  if (posY\_idx == targetY) {  currDir = 4; // go straight  goto changed;  }  }  if (myMap[posY\_idx + 1][posX\_idx] == 0 || myMap[posY\_idx + 1][posX\_idx] == 4) { // target is lower  direction[2] = 1;  if (posY\_idx < targetY) {  currDir = 3; // go down  goto changed;  }  }  if (direction[3] == 1) {  currDir = 4;  goto changed;  }  for (int i = 0; i < 4; i++) {  if (direction[i] == 1) {  currDir = i + 1;  break;  }  }  goto changed;  }  }  } |
|  | line 205:  레이블 |
| 205 | changed:; |
|  | } |

1. draw() 고스트 그리기

|  |  |
| --- | --- |
|  | 고스트가 진행하는 방향에 따라 이미지 인덱스를 정하고 그린다. |
| 1  2  3 | // ghost  red.ghostPic[red.currDir - 1].draw(red.posX, red.posY, 15, 15);  orange.ghostPic[orange.currDir - 1].draw(orange.posX, orange.posY, 15, 15); |

1. 게임 플로우

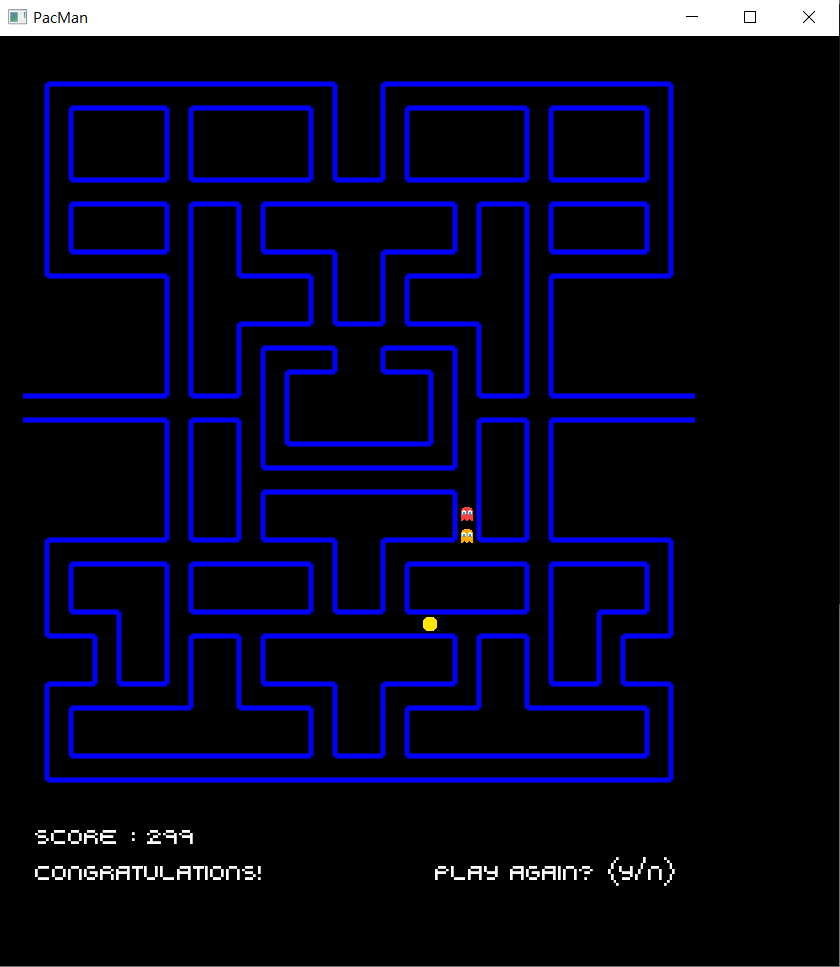
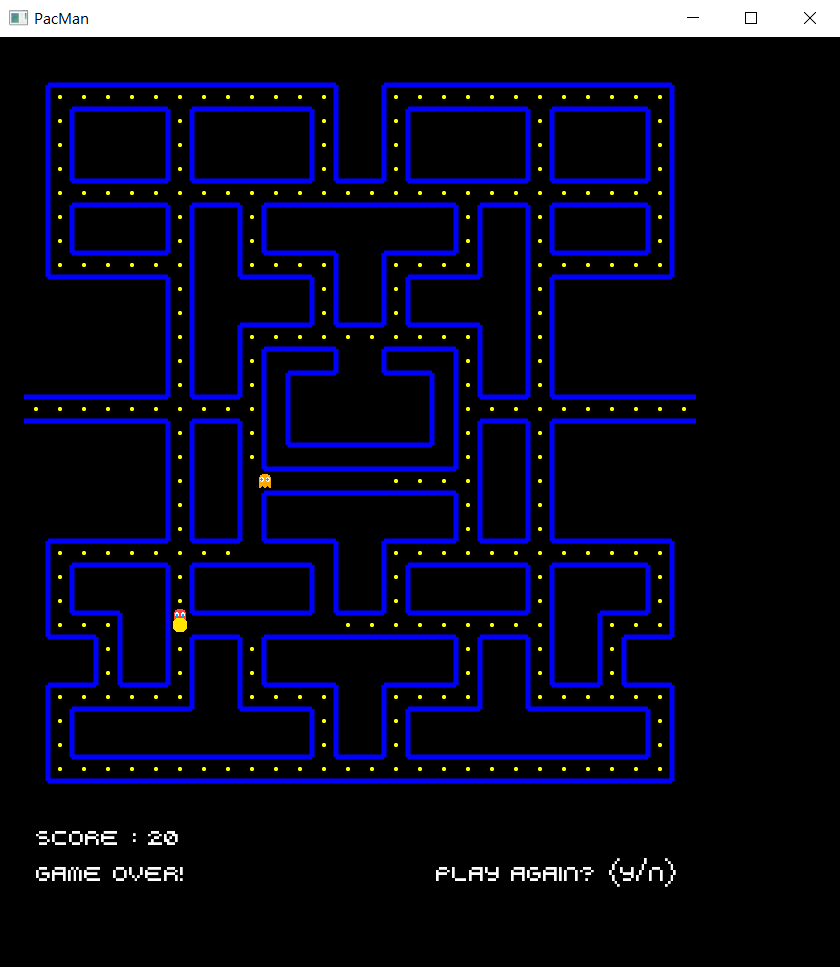
처음 게임이 시작되면 고스트는 고스트 하우스 내부에서 바로 출발한다. 팩맨은 시작 지점에 정지해 있다가 사용자가 방향을 입력하면 움직인다.

모든 도트를 먹으면 ‘congratulations!’이라는 문구가 출력되고, 도트를 먹기 전에 고스트에게 잡히면 ‘game over!’라는 문구가 좌측 하단에 출력된다. 둘 모두의 경우에 ‘again?(y/n)’이라는 문구가 우측 하단에 출력된다. 사용자가 y 또는 Y를 누르면 게임이 다시 시작되고, n 또는 N을 누르면 게임이 종료된다. 플레이 중간에 esc를 눌러도 게임은 종료된다.

게임 종료 여부는 gameover, gamewin을 플래그로 사용하여 표시한다.

**프로젝트 결과**

1. 결과물 스크린샷



위에서부터, 플레이 중, 게임 오버, 게임 승리 화면