C++프로그래밍

프로젝트

프로젝트 명 : Ncurses를 이용한 Snake Game

제출자 : 소프트웨어학과 20171703 정태원

정보보안암호수학과 20182251 최지원

문서 정보 / 수정 내역

Filename	최종보고서-SnakeGame.hwp
원안작성자	최지원
수정작업자	정태원, 최지원

수정날짜	대표수정자	Revision	추가/수정 항목	내용
2020-06-17	최지원	1.0	최초 작성	보고서 틀, 1. 서론, ‘Ⅰ. 목표, Ⅱ. 설계’ 작성
2020-06-26	정태원	1.1	내용 수정	‘3. 추가한 규칙’, ‘4. 결과’ 세부 내용 작성, ‘Ⅳ. 실행 화면’ 작성
2020-06-26	최지원	1.2	내용 수정	‘2. 단계별 수행 과정’ 세부 내용 작성
2020-06-27	정태원, 최지원	1.3	최종 내용 수정	최종적으로 보고서 정리

목차

(※ 2-1 ~ 2-5, 3-1 ~ 3-5 각각에는 'Ⅰ. 목표, Ⅱ. 설계, Ⅲ. 코드, Ⅳ. 실행 화면' 내용이 포함되어 있음)

1. 서론

1-1. 프로젝트 주제

Ncurses 라이브러리를 활용하여 주어진 규칙을 따르는 Snake Game을 제작하기

1-2. 계획하기

프로젝트 수행을 위한 주차 별 계획 세우기

- 5월 3주차 : 전체 주차 별 계획 세우기 및 협업을 편하게 하기 위한 Ubuntu 설치
- 5월 4주차 : Ncurses 라이브러리 공부 및 rule 1 구현
- 5월 5주차 : rule 2, rule 3 구현
- 6월 1주차 : rule 4 구현
- 6월 2주차 : rule 5, rule 6 구현
- 6월 3주차 : 마무리 및 보고서, ppt 작성, makefile 생성

2. 단계별 수행 과정

2-1. 1단계

Ⅰ. 목표

Ncurses 라이브러리 함수들을 사용해서 2차원 배열로 된 Snake Map을 Game 화면으로 표시하는 프로그램 완성하기

Ⅱ. 설계

(1) 'SnakeGame.h', 'SnakeGame.cpp', 'main.cpp'의 세 파일로 나누기

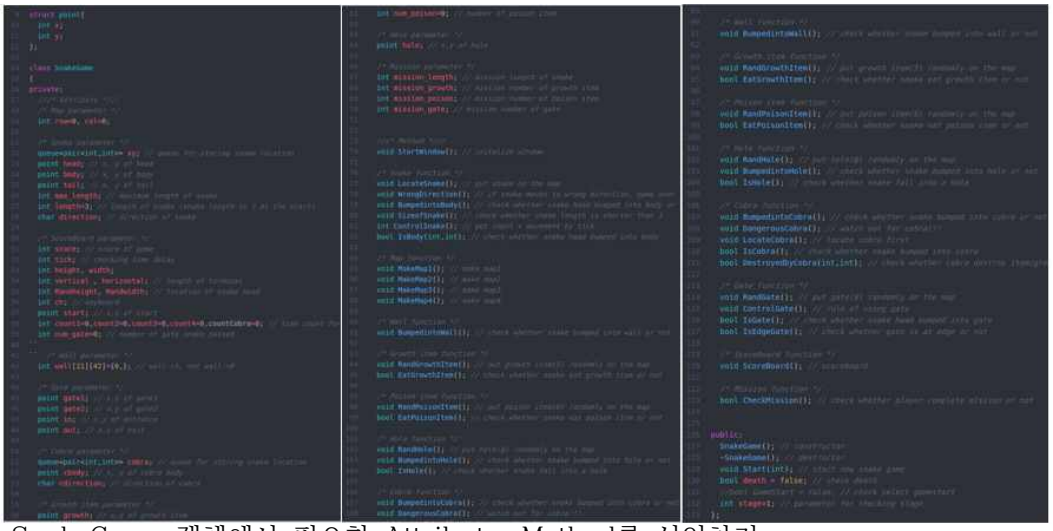
- SnakeGame.h : SnakeGame 클래스를 선언하는 파일
- SnakeGame.cpp : SnakeGame.h에서 선언한 각각의 메소드를 구현하는 파일
- main.cpp : Snake Game을 동작하도록 하는 파일

(2) Ncurses 모드로 창을 여는 StartWindow() 함수 작성하기

(3) Ncurses를 이용하여 Snake Map을 화면에 나타내는 MakeMap1() 함수 작성하기

Ⅲ. 코드

(1) ‘Snake.h’ 파일에 있는 SnakeGame 객체



- SnakeGame 객체에서 필요한 Attribute, Method를 선언하기
  - · Snake
  - · Map
  - · Wall
  - · Growth item
  - · Poison item
  - · ScoreBoard
  - · Mission
  - · Hole (3-3 참고)
  - · Cobra (3-4 참고)

(2) main.cpp

```
int main()
{

    SnakeGame StartGame;
    while(StartGame.stage<=4)
    {
      beep();
      StartGame.Start(StartGame.stage);

      if(StartGame.death == TRUE)
      {
        StartGame.Start(0);
      }
    }
    return 0;
```

(6) SnakeGame을 실행하기 위한 StartGame 객체를 생성합니다.

(7-16) stage마다 StartGame 실행합니다.

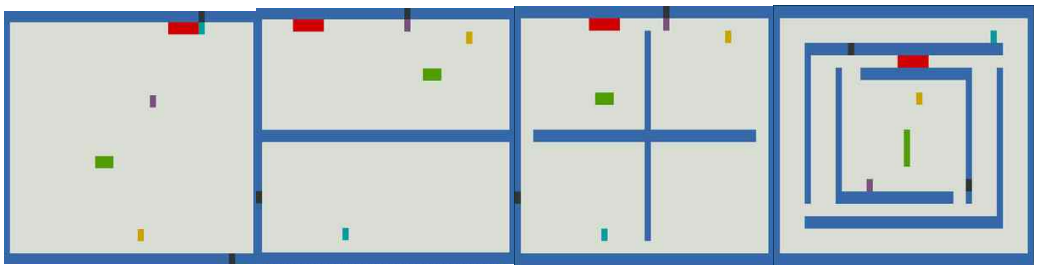
(7) stage가 1,2,3,4일 때에만 실행합니다.
(9) beep() 함수로 효과음을 냅니다. (3-1 참고)
(10) 해당 stage에서 게임 실행합니다.
(12-15) Gameover 되면 StartGame.start(0)으로 돌아갑니다.

(3) MakeMap1() 함수

```
51  void SnakeGame::MakeMap1()
52  {
53      height=21;
54      width=42;
55      start.y=start.x=0;
56
57      for(int i=1;i<width-1;i++)
58      {
59        for(int j=1;j<height-1;j++)
60        {
61          attron(COLOR_PAIR(5));
62          mvwprintw(stdscr,j,i," ");
63          attroff(COLOR_PAIR(5));
64        }
65      }
66      attron(COLOR_PAIR(7));
67      mvwprintw(stdscr,0, 0,"x");
68      mvwprintw(stdscr,height-1,0 ,"x");
69      mvwprintw(stdscr,0, width-1,"x");
70      mvwprintw(stdscr,height-1, width-1,"x");
71
72      for(int j=1;j<height-1;j++)
73      {
74        mvwprintw(stdscr,j,0,".");
75        wall[j][0]= 1;
76        mvwprintw(stdscr ,j,width-1,".");
77        wall[j][width-1]=1;
78      }
79      for(int i=1;i<width-1;i++)
80      {
81        mvwprintw(stdscr ,0,i,".");
82        wall[0][i]=1;
83        mvwprintw(stdscr ,height-1,i,".");
84        wall[height-1][i]=1;
85      }
86      attroff(COLOR_PAIR(7));
87  }
```

(57-65) Wall이 아닌 곳을 COLOR_PAIR(5)로 설정합니다.
(66, 86) Wall과 Immune wall을 COLOR_PAIR(7)로 설정합니다.
(67-70) Immune wall 설정합니다.
(72-85) Wall 설정합니다.
(75, 77, 82, 84) Wall인 좌표는 wall 배열에서 1로 설정합니다. ((i) 참고)

Ⅳ. 실행 화면



2-2. 2단계

Ⅰ. 목표

1단계의 Map 위에 Snake를 표시하고, 화살표를 입력받아 Snake가 움직이도록 프로그램 완성하기

Ⅱ. 설계

(1) Snake를 구성하는 좌표를 queue에 pair로 저장하기

(2) Ncurses의 getch() 함수를 이용해서 키보드 입력 받기

(3) Snake가 진행 방향의 반대 방향으로 이동했을 경우 게임을 종료하는 WrongDirection() 함수 작성하기

(4) Snake가 자신의 body를 만나는지 확인하는 IsBody()함수를 작성하고 만나면 게임을 종료하는 BumpedintoBody() 함수 작성하기

(5) Snake가 Wall과 만나면 게임을 종료하는 BumpedintoWall() 함수 작성하기

· Map의 모든 좌표를 담을 수 있는 이차원 배열(wall)을 만들어 0으로 초기화하고 Wall인 경우 배열 값에 1을 대입하기…(ⅰ)

· Snake의 head 좌표의 wall 배열 값이 1이라면 게임을 종료하는 BumpedintoWall() 함수 작성하기

(6) 키보드로 입력받은 방향키에 따라 Snake가 0.2초마다 움직이도록 코드 작성하기

Ⅲ. 코드

(1) Snake를 구성하는 좌표를 queue에 pair로 저장하기

```
queue<pair<int,int>> xy; // queue for storing snake location
```

- Snake body를 pop하기 편하도록 xy.back이 Snake head가 되고 나머지가 Snake body가 되도록 queue에 좌표값을 pair로 저장합니다.

(2) getch() 함수(ControlSnake 함수의 일부), WrongDirection() 함수

```
369   if(ch =getch())
370   {
371     switch(ch)
372     {
373       case KEY_LEFT:
374         if(direction!='r')
375         {
376           direction='l';
377         }
378         else
379         {
380           WrongDirection();
381         }
382         break;
383
384       case KEY_RIGHT:
385         if(direction!='l')
386         {
387           direction='r';
388         }
389         else
390         {
391           WrongDirection();
392         }
393         break;
394
```

```
394
395       case KEY_UP:
396         if(direction!='d')
397         {
398           direction='u';
399         }
400         else
401         {
402           WrongDirection();
403         }
404         break;
405
406       case KEY_DOWN:
407         if(direction!='u')
408         {
409           direction='d';
410         }
411         else
412         {
413           WrongDirection();
414         }
415         break;
416       }
417     }
```

(369) getch() 함수로 키보드 입력받습니다.

(373, 384, 395, 406) 각각의 키보드 입력 값에 대해서

(374-377, 385-388, 396-399, 407-410) 방향을 잡아줍니다.

(378-381, 389-393, 400-404, 411-415) Snake의 진행 방향과 반대 방향이 입력되었을 경우 WrongDirection()함수를 호출해서 게임을 종료합니다.

(3) IsBody() 함수

```
829   bool SnakeGame::IsBody(int rand1, int rand2)
830   {
831     for(int i=0;i<xy.size();i++)
832     {
833       body.x = xy.front().second;
834       body.y = xy.front().first;
835
836       if(body.x==rand2 && body.y==rand1)
837       {
838         return TRUE;
839       }
840       else
841       {
842         xy.pop();
843       }
844       xy.push(make_pair(body.y,body.x));
845     }
846     return FALSE;
847   }
```

(829) (x,y) 좌표값 (rand2, rand1)을 입력받아서 해당 좌표값이 Body의 좌표값과 일치하는지 확인하여 일치하면 true, 아니면 false를 반환하는 함수입니다.

(831) Snake의 길이만큼 for문을 반복합니다.

(833, 834) Snake의 queue의 front를 body.x, body.y 각각에 대입합니다.

(836-843) body.x, body.y값과 입력값(rand2, rand1)이 일치하면 true반환,
일치하지 않으면 xy를 pop하여 다음 queue값 비교합니다.
(844) pop한 pair값을 다시 push합니다.
(845) xy의 모든 값이 일치하지 않으면 false를 반환합니다.

(4) BumpedintoBody() 함수

```
1065  void SnakeGame::BumpedintoBody()
1066  {
1067    clear();
1068    WINDOW * win = newwin(height, width, start.y,start.x);
1069    refresh();
1070    mvwprintw(win, 10, 5,"Game Over!!!\tBumped into Body!!!");
1071    wrefresh(win);
1072    beep();
1073    usleep(2000000);
1074    delwin(win);
1075  }
```

(1067) clear() 함수로 화면 전체를 지웁니다.
(1068) height*width 크기의 새로운 WINDOW 화면 win을 생성합니다.
커서는 (start.x, start.y)에 위치하게 됩니다.
(1069) refresh() 함수로 터미널에 win이 출력되게 합니다.
(1070) win 화면의 (5,10) 지점에 'Game Over!!! Bumped into Body!!!'를 출력합니다.
(1072) beep() 함수로 효과음을 낸다 (3-1 참고)
(1073) usleep() 함수로 2초동안 유지되게 합니다.
(1074) win 화면을 종료합니다.

(5) BumpedintoWall() 함수

```
660  void SnakeGame::BumpedintoWall()
661  {
662    head.x=xy.back().second;
663    head.y=xy.back().first;
664
665    if(wall[head.y][head.x]==1)
666    {
667      clear();
668      WINDOW * win = newwin(height, width, start.y,start.x);
669      refresh();
670      mvwprintw(win, 10, 5,"Game Over!!!\tBumped into Wall!!");
671      wrefresh(win);
672      beep();
673      usleep(2000000);
674      delwin(win);
675      stage = 0;
676      death = TRUE;
677    }
678  }
```

(662-663) Snake의 queue의 back를 head.x, head.y 각각에 대입합니다.
(665) 해당 좌표값이 wall 배열에서 1일 때, 즉 해당 좌표가 Wall일 때 실행합니다.
(667-674) (4) BumpedintoBody() 함수와 동일한 과정으로 진행됩니다.
(670) 'Game over!!! Bumped into Wall!!'를 출력합니다.

(675) stage를 0으로 초기화합니다.
(676) death를 true로 선언합니다.

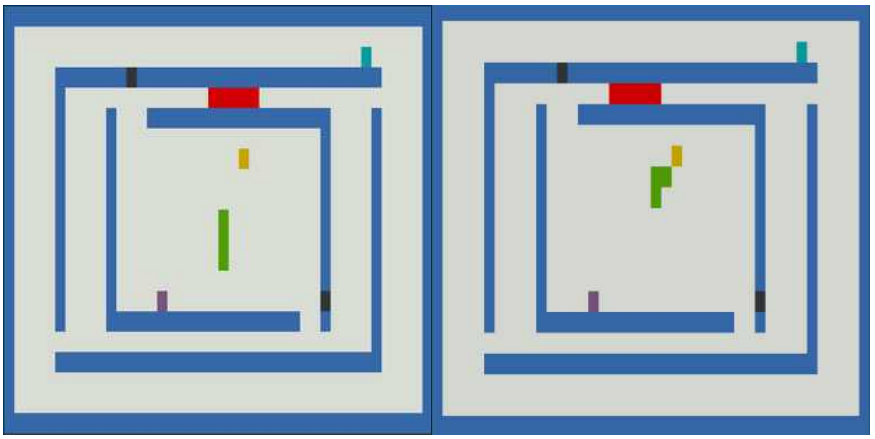
(6) ControlSnake() 함수

```
419       if(direction == 'l')
420       {
421         tail.y = xy.front().first;
422         tail.x = xy.front().second;
423         attron(COLOR_PAIR(5));
424         mvwprintw(stdscr, xy.front().first, xy.front().second," ");
425         attroff(COLOR_PAIR(5));
426         xy.pop();
427         attron(COLOR_PAIR(3));
428         mvwprintw(stdscr, xy.back().first, xy.back().second,"4");
429         attroff(COLOR_PAIR(3));
430         if(IsBody(xy.back().first, xy.back().second-1)==TRUE)
431         {
432           BumpedintoBody();
433           death =TRUE;
434         }
435
436         xy.push(make_pair(xy.back().first, xy.back().second-1));
437         ControlGate();
438         attron(COLOR_PAIR(3));
439         mvwprintw(stdscr, xy.back().first, xy.back().second,"3");
440         attroff(COLOR_PAIR(3));
```

- 키보드로부터 'l', 'r', 'u', 'd'를 입력받지만 같은 내용이므로 'l'에 대해서만 설명하겠습니다.

(419) 키보드로부터 입력받은 방향이 'l'일 때
(421-422) Snake의 queue의 front를 tail.y, tail.x에 각각에 대입합니다.
(423-425) xy.front를 다시 COLOR_PAIR(5)로 설정합니다.
(426) Snake가 왼쪽으로 움직여야 하므로 tail를 pop합니다.
(427-429) xy.back을 COLOR_PAIR(3)으로 설정하여 body로 만듭니다.
(430-434) 왼쪽으로 이동할 좌표를 IsBody() 함수에 대입해서 true이면 BumpedintoBody() 함수를 호출하고 death를 true로 선언합니다. (game over)
(436) 해당 좌표가 body와 만나지 않으므로 이동해도 되는 방향이므로 xy에 push합니다.
(438-440) 해당 좌표는 head이기 때문에 COLOR_PAIR(5)로 설정하여 head로 만듭니다.

Ⅳ. 실행 화면



키보드로부터 'r'을 입력받았습니다.

2-3. 3단계

Ⅰ. 목표

2단계 프로그램에서 Map 위에 Growth item과 Poison item이 출현하도록 프로그램 완성하기

Ⅱ. 설계

(1) Wall과 Snake를 제외한 Map 위에 랜덤으로 Growth item, Poison item이 생성되도록 RandGrowthItem(), RandPoisonItem() 함수 작성하기

· 랜덤으로 돌린 좌표의 wall 배열 값이 1이 아닌 곳에 Growth item, Poison item이 생성되도록 RandGrowthItem(), RandPoisonItem() 함수를 작성한다

· 임의의 좌표를 대입해서 해당 좌표가 Snake의 좌표에 속하면 true를 반환하고 속하지 않으면 false를 반환하는 IsBody() 함수를 만든다.
Snake에 Growth item, Poison item이 생성되지 않도록 랜덤으로 돌린 좌표가 IsBody()에 대입했을 때, false가 되도록 RandGrowthItem(), RandPoisonItem() 함수를 작성한다

(2) Snake의 head가 Growth item과 Poison item과 만나면 true를 반환, 만나지 않았다면 false를 반환하는 EatGrowthItem(), EatPoisonItem() 함수 작성하기

(3) EatGrowthItem()이 true라면 Snake 길이를 1 늘리고, EatPoisonItem()이 true라면 Snake 길이를 1 줄이기
이 때, Snake 길이가 3보다 작으면 게임을 종료하는 SizeofSnake() 함수 작성하기

(4) 특정 시간이 지나면(6초) Growth item과 Poison item의 위치가 바뀌도록 RandGrowthItem(), RandPoisonItem() 함수 수정하기

Ⅲ. 코드

(1) RandGrowthitem() 함수

```
725  void SnakeGame::RandGrowthItem()
726  {
727    if((DestroyedbyCobra(growth.y,growth.x)==TRUE)||(EatGrowthItem()==TRUE)||
728    (count1%30==0 && count1!=0))
729    {
730      attron(COLOR_PAIR(5));
731      mvwprintw(stdscr,growth.y,growth.x," ");
732      attroff(COLOR_PAIR(5));
733      while(1)
734      {
735        growth.x=rand()%(width-2)+1;
736        growth.y=rand()%(height-2)+1;
737        if(IsBody(growth.y,growth.x)==false && wall[growth.y][growth.x]!=1)
738        {
739          break;
740        }
741      }
742      attron(COLOR_PAIR(9));
743      mvwprintw(stdscr,growth.y,growth.x,"5");
744      attroff(COLOR_PAIR(9));
745    }
746    if(count1 == 0)
747    {
748      while(1)
749      {
750        growth.x=rand()%(width-2)+1;
751        growth.y=rand()%(height-2)+1;
752        if(IsBody(growth.y,growth.x)==false && wall[growth.y][growth.x]!=1)
753        {
754          break;
755        }
756      }
757      attron(COLOR_PAIR(9));
758      mvwprintw(stdscr,growth.y,growth.x,"5");
759      attroff(COLOR_PAIR(9));
760    }
761    count1++;
762  }
```

(727-728) 해당 좌표가 DestroyedbyCobra에서 true(3-4 참고)이거나 EatGrowthItem()에서 true(참고)이거나 count가 0이 아닌 30의 배수일 때 실행합니다.
(count1는 Growth item이 특정 시간이 지나면 다른 곳에 생기게 하기 위해 지정한 변수. 6초가 지나면 다른 곳에 생기게 설정하였습니다.)

(730-732) 기존 Growth item이 있던 곳을 COLOR_PAIR(5)로 설정합니다.

(733-741) growth.x, growth.y 변수를 rand() 함수를 이용해서 랜덤으로 설정하고 해당 좌표를 IsBody() 함수에 넣었을 때 false, 즉 Body 위의 좌표가 아니고 해당 좌표를 wall 배열에 넣었을 때 1이 아니면, 즉 Wall이 아니면 해당 좌표에 Growth item이 생겨도 되므로 break문을 써서 무한루프문을 탈출합니다.

(742-744) 해당 좌표에 Growth item이 생겨야 하므로 COLOR_PAIR(9)로 설정합니다.

(746-760) count1이 0이면, 즉 해당 stage를 처음 시작했을 때, (733-744) 과정을 실행합니다.

(761) count1값 1 증가시킵니다.

(2) RandPoisonItem() 함수

```
763  void SnakeGame::RandPoisonItem()
764  {
765    if((DestroyedbyCobra(poison.y,poison.x)==TRUE)||(EatPoisonItem()==TRUE)||
766    (count2%30==0 && count2!=0))
767    {
768      attron(COLOR_PAIR(5));
769      mvwprintw(stdscr,poison.y,poison.x," ");
770      attroff(COLOR_PAIR(5));
771      while(1)
772      {
773        poison.x=rand()%(width-2)+1;
774        poison.y=rand()%(height-2)+1;
775        if(IsBody(poison.y, poison.x)==false && wall[poison.y][poison.x]!=1)
776        {
777          break;
778        }
779      }
780      attron(COLOR_PAIR(8));
781      mvwprintw(stdscr,poison.y,poison.x,"6");
782      attroff(COLOR_PAIR(8));
783    }
784    if(count2 == 0)
785    {
786      while(1)
787      {
788        poison.x=rand()%(width-2)+1;
789        poison.y=rand()%(height-2)+1;
790        if(IsBody(poison.y, poison.x)==false && wall[poison.y][poison.x]!=1)
791        {
792          break;
793        }
794      }
795      attron(COLOR_PAIR(8));
796      mvwprintw(stdscr,poison.y,poison.x,"6");
797      attroff(COLOR_PAIR(8));
798    }
799    count2++;
800  }
```

- (1) RandGrowthItem() 함수와 같은 원리로 동작하는 함수입니다.

(3) EatGrowthItem() 함수

```
802  bool SnakeGame::EatGrowthItem()
803  {
804    if((growth.x==xy.back().second) && (growth.y==xy.back().first))
805    {
806      return TRUE;
807    }
808    else
809    {
810      return FALSE;
811    }
812  }
```

(802) Growth item을 먹으면 true, 먹지 않았으면 false를 반환하는 함수입니다.

(804-811) xy.back (Snake head)이 Growth item의 좌표값과 같다면 Growth item을 먹은 것이므로 true를 반환하고 아니면 false를 반환합니다.

(4) EatPoisonItem() 함수

```
814  bool SnakeGame::EatPoisonItem()
815  {
816    if((poison.x==xy.back().second) && (poison.y==xy.back().first))
817    {
818      if(xy.size()<3)
819      {
820        SizeofSnake();
821      }
822      return TRUE;
823    }
824    else
825    {
826      return FALSE;
827    }
828  }
```

- (3) EatGrowthItem() 함수와 같은 원리로 동작하는 함수입니다.

(818-821) Snake의 크기가 3보다 작아지면 SizeofSnake() 함수를 호출합니다.

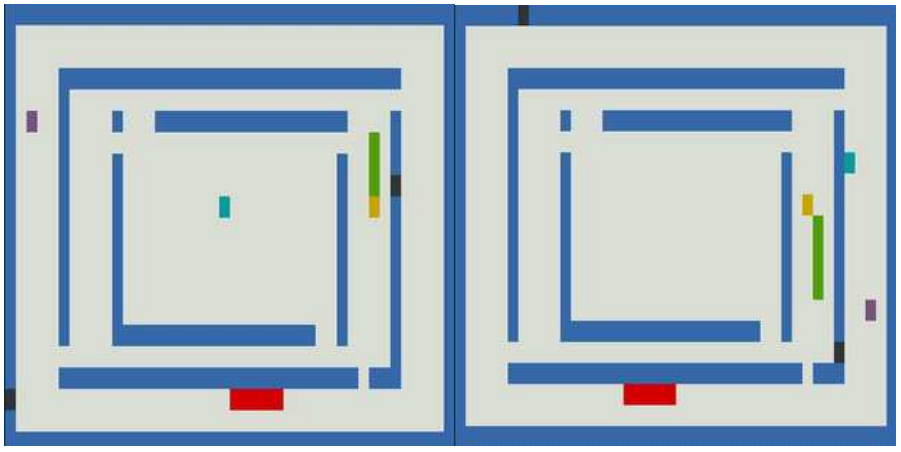
(5) SizeofSnake() 함수

```
850  void SnakeGame::SizeofSnake()
851  {
852    clear();
853    WINDOW * win = newwin(height, width, start.y, start.x);
854    refresh();
855    mvwprintw(win, 10, 5,"Game Over!!!\tToo Short!!!");
856    wrefresh(win);
857    beep();
858    usleep(2000000);
859    delwin(win);
860  }
```

- 2-2.Ⅲ.(4) BumpedintoBody() 함수와 동일한 과정으로 진행됩니다.

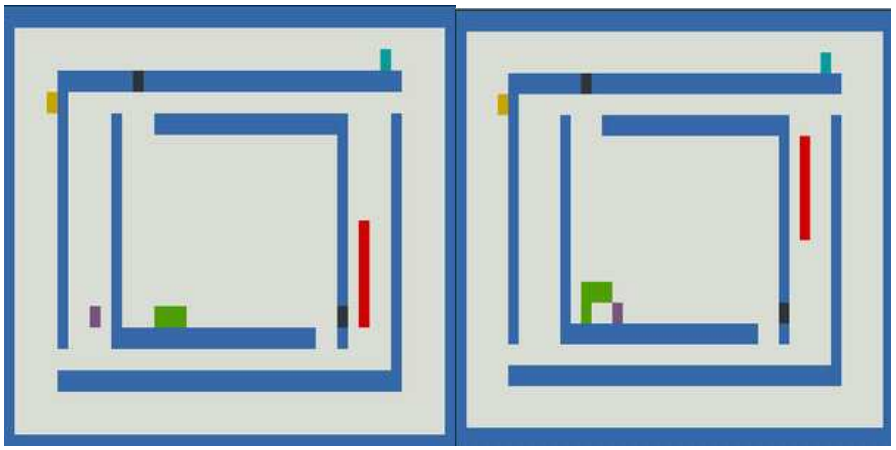
(855) 'Game Over!!! Too Short!!!'를 출력합니다.

Ⅳ. 실행 화면

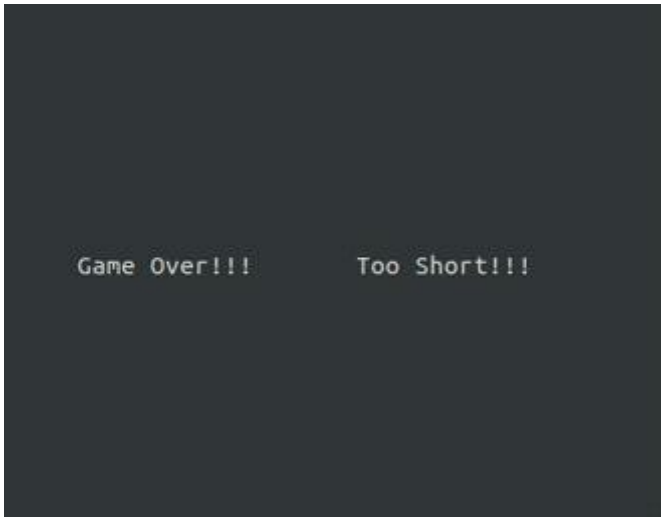


아이템을 먹고 증가(EatGrowthItem()), 아이템의 위치가 최신화(RandGrowthitem())된 모습입니다.

먹지않아도 시간이 지나면 새로 나타납니다.



반대로 줄어드는 모습입니다. (EatPoisonItem(), RandPoisonItem())



SizeofSnake() 함수 실행 화면입니다.

2-4. 4단계

Ⅰ. 목표

3단계 프로그램에서 Map의 Wall의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate에 Snake가 통과할 수 있도록 프로그램 완성하기

Ⅱ. 설계

(1) Wall에 임의의 한 쌍의 Gate가 생기도록 RandGate() 함수 작성하기

· Wall에 Gate가 생성되어야 하므로 랜덤으로 돌린 좌표 값이 wall 배열에서 1이라면 해당 좌표에 Gate 생성

· 각각의 Gate가 같은 위치에 생기면 다른 위치에 생길 때까지 위의 과정 반복하기

(2) Snake의 head가 Gate의 좌표와 만나면 해당 Gate를 in, 나머지 Gate를 out으로 지정한 후 true를 반환하고, 만나지 않으면 false를 반환하는 IsGate() 함수 작성하기

(3) Gate가 가장자리에 생기면 True를 내부에 생기면 False를 반환하는 IsEdgeGate()

함수 작성하기

(4) 임의의 좌표에서 IsGate()의 반환값이 true일 때, IsEdgeGate()의 반환값이 true인지 false인지에 따라 규칙을 달리하여 direction을 부여하고, 해당 direction에 따라 규칙에 만족하도록 Snake가 움직이게 ControlGate() 함수 작성하기

(5) 특정 시간이 지나면(8초) Gate의 위치가 바뀌도록 RandGate() 함수 수정하기

Ⅲ. 코드

(1) RandGate() 함수

```
680  void SnakeGame::RandGate()
681  {
682    if(count3%40==0 && count3!=0)
683    {
684      wall[gate1.y][gate1.x]=1;
685      wall[gate2.y][gate2.x]=1;
686      attron(COLOR_PAIR(7));
687      mvwprintw(stdscr,gate1.y,gate1.x,".");
688      mvwprintw(stdscr,gate2.y,gate2.x,".");
689      attroff(COLOR_PAIR(7));
690      while(1)
691      {
692        gate1.x=rand()%42;
693        gate1.y=rand()%21;
694        gate2.x=rand()%42;
695        gate2.y=rand()%21;
696        if(((gate1.x!=gate2.x)||(gate1.y!=gate2.y)) && wall[gate1.y][gate1.x]==1 && wall[gate2.y][gate2.x]==1)
697        {
698          break;
699        }
700      }
701    }
702    if(count3 == 0)
703    {
704      while(1)
705      {
706        gate1.x=rand()%42;
707        gate1.y=rand()%21;
708        gate2.x=rand()%42;
709        gate2.y=rand()%21;
710        if(((gate1.x!=gate2.x)||(gate1.y!=gate2.y)) && wall[gate1.y][gate1.x]==1 && wall[gate2.y][gate2.x]==1)
711        {
712          break;
713        }
714      }
715    }
716    wall[gate1.y][gate1.x]=0;
717    wall[gate2.y][gate2.x]=0;
718    attron(COLOR_PAIR(0));
719    mvwprintw(stdscr,gate1.y,gate1.x," ");
720    mvwprintw(stdscr,gate2.y,gate2.x," ");
721    attroff(COLOR_PAIR(0));
722    count3++;
723  }
```

(682) count3이 0이 아닌 40의 배수일 때 실행됩니다.
(count3는 Gate가 특정 시간이 지나면 다른 곳에 생기게 하기 위해 지정한 변수. 8초가 지나면 다른 곳에 생기게 설정하였습니다.)

(684-685) 기존 Gate가 있던 좌표는 다시 Wall이 되어야 하므로 wall 배열에서 1로 설정합니다.

(686-689) 해당 좌표를 COLOR_PAIR(7)로 설정합니다.

(690-701) gate1, gate2의 좌표값을 랜덤으로 잡고 gate1과 gate2가 서로 같지 않고 해당 좌표가 wall 배열에서 1일 때 Gate가 생길 수 있기 때문에 break문을 통해 무한루프문을 탈출하게 됩니다.

(702) count3이 0이면, 즉 stage를 처음 시작했을 때

(704-714) (690-700)과 동일한 과정으로 진행됩니다.

(716-717) 해당 좌표가 Gate가 되어야 하므로 wall 배열에서의 값을 0으로 설정합니

다.

(718-721) 해당 좌표를 COLOR_PAIR(0)으로 설정합니다.

(722) count3값 1 증가시킵니다.

(2) IsGate() 함수

```
862  bool SnakeGame::IsGate()
863  {
864    if(xy.back().second==gate1.x && xy.back().first==gate1.y)
865    {
866      in.x=gate1.x;
867      in.y=gate1.y;
868      out.x=gate2.x;
869      out.y=gate2.y;
870      num_gate++;
871      ScoreBoard();
872      return true;
873    }
874    else if(xy.back().second==gate2.x && xy.back().first==gate2.y)
875    {
876      in.x=gate2.x;
877      in.y=gate2.y;
878      out.x=gate1.x;
879      out.y=gate1.y;
880      num_gate++;
881      ScoreBoard();
882      return true;
883    }
884    else
885    {
886      return false;
887    }
888  }
```

(862) 해당 좌표가 Gate면 true, 아니면 false를 반환하는 함수입니다.

(864) xy.back (Snake head) 좌표값이 gate1의 좌표값과 같으면

(866-869) gate1을 in으로 gate2를 out으로 설정합니다.

(870) num_gate를 1 증가시킵니다.

(871) ScoreBoard() 함수를 호출합니다.

(882) true를 반환합니다.

(884-887) 아니면 false를 반환합니다.

(3) IsEdgeGate() 함수

(1038) 가장자리의 Gate이면 true를 내부의 Gate이면 false를 반환하는 함수입니다.

(1040-1044) out.y가 0, 즉 가장자리일 때 direction을 'd'로 주고 true 반환합니다.

(1045-1049) out.y가 20, 즉 가장자리일 때 direction을 'u'로 주고 true 반환합니다.

(1050-1054) out.x가 0, 즉 가장자리일 때 direction을 'r'로 주고 true 반환합니다.

(1055-1059) out.x가 41, 즉 가장자리일 때 direction을 'l'로 주고 true 반환합니다.

(1060-1063) 그 외에는 false 반환합니다.

```
1038  bool SnakeGame::IsEdgeGate()
1039  {
1040    if(out.y==0)
1041    {
1042      direction = 'd';
1043      return true;
1044    }
1045    else if(out.y==20)
1046    {
1047      direction = 'u';
1048      return true;
1049    }
1050    else if(out.x==0)
1051    {
1052      direction = 'r';
1053      return true;
1054    }
1055    else if(out.x==41)
1056    {
1057      direction = 'l';
1058      return true;
1059    }
1060    else
1061    {
1062      return false;
1063    }
1064  }
```

(4) ControlGate() 함수

```
909  void SnakeGame::ControlGate()
910  {
911    if(IsGate()==true)
912    {
913      if(IsEdgeGate()==true)
914      {
915        for(int i=0;i<xy.size()-1;i++)
916        {
917          body.x = xy.front().second;
918          body.y = xy.front().first;
919          xy.pop();
920          xy.push(make_pair(body.y,body.x));
921        }
922        xy.pop();
923        attron(COLOR_PAIR(0));
924        mvwprintw(stdscr, in.y, in.x, " ");
925        attroff(COLOR_PAIR(0));
926
927        switch(direction){
928          case 'r':
929              xy.push(make_pair(out.y,out.x+1));
930              break;
931          case 'l':
932              xy.push(make_pair(out.y,out.x-1));
933              break;
934          case 'u':
935              xy.push(make_pair(out.y-1,out.x));
936              break;
937          case 'd':
938              xy.push(make_pair(out.y+1,out.x));
939              break;
940        }
941      }
```

```
942    else if(IsEdgeGate()!=true)
943    {
944      switch(direction){
945        case 'r':
946            if(wall[out.y][out.x+1] != 1)
947            {
948              xy.push(make_pair(out.y,out.x+1));
949              direction='r';
950            }
951            else if(wall[out.y+1][out.x] != 1)
952            {
953              xy.push(make_pair(out.y+1,out.x));
954              direction='d';
955            }
956            else if(wall[out.y-1][out.x] != 1)
957            {
958              xy.push(make_pair(out.y-1,out.x));
959              direction='u';
960            }
961            else
962            {
963              xy.push(make_pair(out.y,out.x-1));
964              direction='l';
965            }
966            break;
```

(911-913) IsGate와 IsEdgeGate()가 true인 경우, 즉 해당 좌표가 EdgeGate인 경우

(915-922) Snake head부터 반대편 Gate로 넘어가야 하기 때문에 Snake head가 나

올 때까지 pop, push를 반복합니다.

(923-925) Snake가 이동하므로 tail은 COLOR_PAIR(0)으로 설정합니다.

(927-940) IsEdgeGate() 함수에서 설정했던 direction에 따라 push를 해줍니다.

(924) IsEdgeGate()가 false인 경우, 즉 해당 좌표가 내부의 Gate인 경우

(944-945) direction은 'r', 'l', 'u', 'd'가 있지만 같은 원리이므로 'r'만 설명하겠습니다.

(946-950) 같은 'r' 방향의 좌표가 wall 배열에서 1이 아니면, 즉 해당 좌표가 Wall이 아니면 빠져나갈 수 있기 때문에 push해주고 direction을 'r'로 설정합니다.

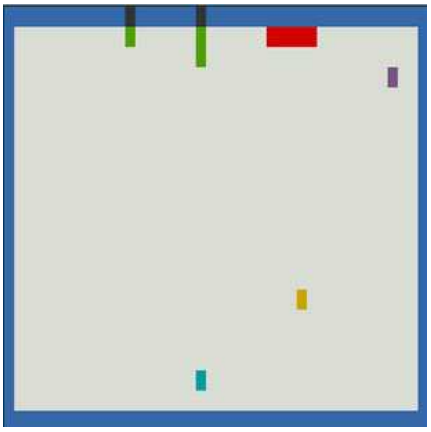
(951-955) 다음 우선순위인 'd' 방향의 좌표가 Wall이 아니면 push해주고 direction을 'd'로 설정합니다.

(956-960) 다음 우선순위인 'u' 방향의 좌표가 Wall이 아니면 push해주고 direction을 'u'로 설정합니다.

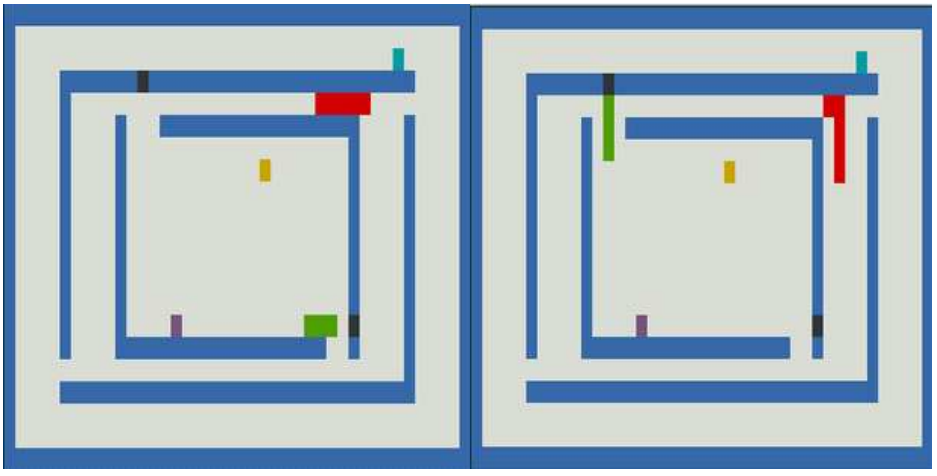
(961-965) 마지막 우선순위인 'l' 방향의 좌표가 Wall이 아니면 push해주고 direction을 'l'로 설정합니다.

(966) switch문을 break문으로 탈출하게 됩니다.

Ⅳ. 실행 화면



가장자리의 Gate를 통해서 Snake가 올바른 방향으로 나오는 모습입니다.



방향에 맞게 잘나오고 있습니다.(우선 순위 1- 오른쪽 2- 아래쪽이지만 오른쪽이 막혀 있어 아래로 나온 모습입니다.)

2-5. 5단계

Ⅰ. 목표

(1) 4단계 프로그램에서 우측에 게임 점수를 표시하는 화면이 뜨도록 프로그램 완성하기

(2) 각 Stage의 Mission을 달성하면 다음 Map으로 진행하도록 프로그램 완성하기

Ⅱ. 설계

(1) Map의 우측에 게임 점수를 표시하는 ScoreBoard() 함수 작성하기

(2) Snake의 길이가 바뀌는 경우, Growth item, Poison item을 먹은 경우, Gate를 지나간 경우 Scoreboard에서 갱신하도록 ScoreBoard() 함수 수정하기

(3) stage를 네 개로 구성해서 해당 stage별로 map을 만드는 함수 MakeMap2(), MakeMap3(), MakeMap4()를 추가적으로 작성하기

(4) 해당 stage의 미션을 모두 성공하면 true를 반환, 다 성공하지 못하면 false를 반환하도록 CheckMission() 함수 작성하기

(5) (4)에서 true를 반환하면 main.cpp에서 다음 stage로 넘어가도록 코드 작성하기 단, stage는 4단계로 구성되어 있으므로 4단계 미션 성공 시 프로그램은 종료된다

Ⅲ. 코드

(1) ScoreBoard() 함수

```
1078  void SnakeGame::ScoreBoard()
1079  {
1080    mvwprintw(stdscr,2,45,"<Score Board>");
1081    mvwprintw(stdscr,3,45,"B :                ");
1082    mvwprintw(stdscr,4,45,"+ :                ");
1083    mvwprintw(stdscr,5,45,"- :                ");
1084    mvwprintw(stdscr,6,45,"G :                ");
1085    attron(COLOR_PAIR(3));
1086    mvwprintw(stdscr,7,45,"-");
1087    attroff(COLOR_PAIR(3));
1088    mvwprintw(stdscr,7,46,": snake");
1089    attron(COLOR_PAIR(4));
1090    mvwprintw(stdscr,8,45,"-");
1091    attroff(COLOR_PAIR(4));
1092    mvwprintw(stdscr,8,46,": gate");
1093    attron(COLOR_PAIR(7));
1094    mvwprintw(stdscr,9,45,"-");
1095    attroff(COLOR_PAIR(7));
1096    mvwprintw(stdscr,9,46,": wall");
1097    attron(COLOR_PAIR(8));
1098    mvwprintw(stdscr,10,45,"-");
1099    attroff(COLOR_PAIR(8));
1100    mvwprintw(stdscr,10,46,": poison");
1101    attron(COLOR_PAIR(9));
1102    mvwprintw(stdscr,11,45,"-");
1103    attroff(COLOR_PAIR(9));
1104    mvwprintw(stdscr,11,46,": growth");
1105    attron(COLOR_PAIR(2));
1106    mvwprintw(stdscr,12,45,"-");
1107    attroff(COLOR_PAIR(2));
1108    mvwprintw(stdscr,12,46,": hole");
1109    attron(COLOR_PAIR(1));
1110    mvwprintw(stdscr,13,45,"-");
1111    attroff(COLOR_PAIR(1));
1112    mvwprintw(stdscr,13,46,": cobra");
```

```
1114    char len1[100];
1115    char len2[100];
1116    char len3[100];
1117    char len4[100];
1118    char len[100];
1119    char growth[100];
1120    char poison[100];
1121    char gate[100];
1122    if(max_length < length)
1123    {
1124      max_length = length;
1125    }
1126    if(max_length>=mission_length)
1127    {
1128      mvwprintw(stdscr,3,49,"Mission Clear");
1129    }
1130    else
1131    {
1132      string itos1= to_string(max_length);
1133      strcpy(len1,itos1.c_str());
1134      mvwprintw(stdscr,3,49,len1);
1135      itos1= to_string(mission_length);
1136      strcpy(len,itos1.c_str());
1137      mvwprintw(stdscr,3,51,"/");
1138      mvwprintw(stdscr,3,52,len);
1139    }
```

(1080-1112) ScoreBoard에 필요한 값들 출력해줍니다.

(1122-1125) max_length에 Snake의 최대 길이 대입해줍니다.

(1126-1129) max_length가 mission_length보다 길거나 같을 경우 'Mission Clear'

출력해줍니다.

(1130-1139) 아닐 경우 max_length(int) 값을 string으로 바꾸고 char배열에 담은 후 ScoreBoard에 길이 출력해줍니다.

- 나머지 ScoreBoard() 함수의 코드는 비슷한 내용이므로 생략하였습니다.

(2) MakeMap2(), MakeMap3(), MakeMap4() 함수

```
void SnakeGame::MakeMap2()
{
    height=21;
    width=42;
    start.y=start.x=0;

    for(int i=1;i<width-1;i++)
    {
      for(int j=1;j<height-1;j++)
      {
        attron(COLOR_PAIR(5));
        mvwprintw(stdscr,j,i," ");
        attroff(COLOR_PAIR(5));
      }
    }
    attron(COLOR_PAIR(7));
    mvwprintw(stdscr,0, 0,"x");
    mvwprintw(stdscr,height-1,0 ,"x");
    mvwprintw(stdscr,0, width-1,"x");
    mvwprintw(stdscr,height-1, width-1,"x");

    for(int j=1;j<height-1;j++)
    {
      mvwprintw(stdscr,j,0,".");
      wall[j][0]= 1;
      mvwprintw(stdscr ,j,width-1,".");
      wall[j][width-1]=1;
    }
    for(int i=1;i<width-1;i++)
    {
      mvwprintw(stdscr,10,i,".");
      wall[10][i]=1;
    }
    for(int i=1;i<width-1;i++)
    {
      mvwprintw(stdscr ,0,i,".");
      wall[0][i]=1;
      mvwprintw(stdscr ,height-1,i,".");
      wall[height-1][i]=1;
    }
    mvwprintw(stdscr,10,0,"x");
    mvwprintw(stdscr,10,width-1,"x");
    attroff(COLOR_PAIR(7));
}
```

```
void SnakeGame::MakeMap3()
{
    height=21;
    width=42;
    start.y=start.x=0;

    for(int i=1;i<width-1;i++)
    {
      for(int j=1;j<height-1;j++)
      {
        attron(COLOR_PAIR(5));
        mvwprintw(stdscr,j,i," ");
        attroff(COLOR_PAIR(5));
      }
    }
    attron(COLOR_PAIR(7));
    mvwprintw(stdscr,0, 0,"x");
    mvwprintw(stdscr,height-1,0 ,"x");
    mvwprintw(stdscr,0, width-1,"x");
    mvwprintw(stdscr,height-1, width-1,"x");

    for(int j=1;j<height-1;j++)
    {
      mvwprintw(stdscr,j,0,".");
      wall[j][0]= 1;
      mvwprintw(stdscr ,j,width-1,".");
      wall[j][width-1]=1;
    }
    for(int i=3;i<width-3;i++)
    {
      mvwprintw(stdscr,10,i,".");
      wall[10][i]=1;
    }
    for(int j=2;j<height-2;j++)
    {
      mvwprintw(stdscr,j,21,".");
      wall[j][21]=1;
    }
    for(int i=1;i<width-1;i++)
    {
      mvwprintw(stdscr ,0,i,".");
      wall[0][i]=1;
      mvwprintw(stdscr ,height-1,i,".");
      wall[height-1][i]=1;
    }
    attroff(COLOR_PAIR(7));
}
```

```
void SnakeGame::MakeMap4()
{
    height=21;
    width=42;
    start.y=start.x=0;

    for(int i=1;i<width-1;i++)
    {
      for(int j=1;j<height-1;j++)
      {
        attron(COLOR_PAIR(5));
        mvwprintw(stdscr,j,i," ");
        attroff(COLOR_PAIR(5));
      }
    }
    attron(COLOR_PAIR(7));
    mvwprintw(stdscr,0, 0,"x");
    mvwprintw(stdscr,height-1,0 ,"x");
    mvwprintw(stdscr,0, width-1,"x");
    mvwprintw(stdscr,height-1, width-1,"x");

    for(int j=1;j<height-1;j++)
    {
      mvwprintw(stdscr,j,0,".");
      wall[j][0]= 1;
      mvwprintw(stdscr ,j,width-1,".");
      wall[j][width-1]=1;
    }

    for(int i=1;i<width-1;i++)
    {
      mvwprintw(stdscr ,0,i,".");
      wall[0][i]=1;
      mvwprintw(stdscr ,height-1,i,".");
      wall[height-1][i]=1;
    }

    for(int j=3;j<height-5;j++)
    {
      mvwprintw(stdscr,j,5,".");
      wall[j][5]= 1;
    }
```

- MakeMap1()과 비슷하게 구현하였습니다.

(3) CheckMission() 함수

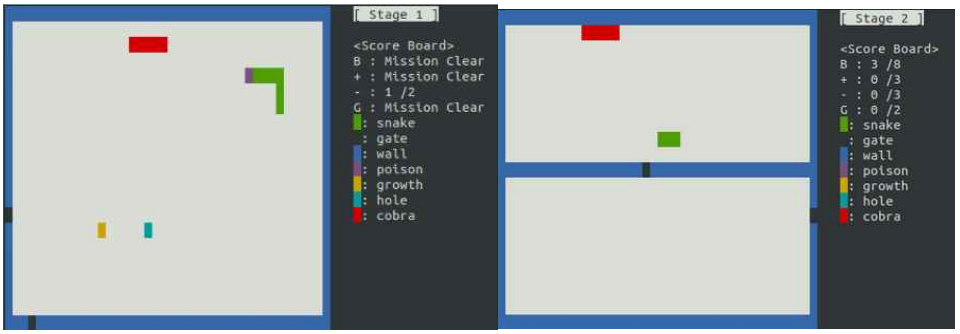
```
bool SnakeGame::CheckMission()
{
  if(stage==1)
  {
    if(max_length>=7 && num_growth>=3 && num_poison>=2 && num_gate>=1)
    {
      stage++;
      countCobra = 0;
      return TRUE;
    }
    else
    {
      return FALSE;
    }
  }
  else if(stage==2)
  {
    if(max_length>=8 && num_growth>=3 && num_poison>=3 && num_gate>=2)
    {
      stage++;
      countCobra = 0;
      return TRUE;
    }
    else
    {
      return FALSE;
    }
  }
```

```
1215    else if(stage==3)
1216    {
1217      if(max_length>=9 && num_growth>=4 && num_poison>=4 && num_gate>=3)
1218      {
1219        stage++;
1220        countCobra = -2;
1221        return TRUE;
1222      }
1223      else
1224      {
1225        return FALSE;
1226      }
1227    }
1228    else if(stage==4)
1229    {
1230      if(max_length>=10 && num_growth>=5 && num_poison>=5 && num_gate>=4)
1231      {
1232        stage++;
1233        countCobra = 0;
1234        return TRUE;
1235      }
1236      else
1237      {
1238        return FALSE;
1239      }
1240    }
1241    else
1242    {
1243      return FALSE;
1244    }
1245  }
```

parameter stage	max_length	num_growth	num_poison	num_gate
1	7	3	2	1
2	8	3	3	2
3	9	4	4	3
4	10	5	5	4

- 각 stage별로 각각의 mission_parameter를 위의 표와 같이 설정해서 성공하면 stage를 1 증가시키고 countCobra를 0으로 초기화한 후, true를 반환, 성공하지 못하면 false를 반환하게 됩니다.

Ⅳ. 실행 화면



마지막 MISSION 클리어하고 맵 전환이 완료되었습니다.(STAGE1 -> STAGE2)

3. 추가한 규칙

3-1. 게임이 종료된 경우나 다음 stage로 넘어가는 경우 효과음 넣기

Ⅰ. 목표

게임이 종료된 경우나 다음 stage로 넘어가는 경우 효과음 넣기

Ⅱ. 설계

Ncurses의 beep() 함수 이용하기

Ⅲ. 코드

beep() 함수 호출

```
660  void SnakeGame::BumpedintoWall()
661  {
662    head.x=xy.back().second;
663    head.y=xy.back().first;
664
665    if(wall[head.y][head.x]==1)
666    {
667      clear();
668      WINDOW * win = newwin(height, width, start.y,start.x);
669      refresh();
670      mvwprintw(win, 10, 5,"Game Over!!!\tBumped into Wall!!");
671      wrefresh(win);
672      beep();
673      usleep(2000000);
674      delwin(win);
675      stage = 0;
```

위와 같이 조건을 걸어놓고 그 안에 beep()함수를 호출하면 시스템 음이 작동하게 됩니다.

Ⅳ. 실행 화면

소리가 발생함으로 화면은 따로 없습니다.

3-2. stage 1이 시작할 때 뱀이 랜덤 위치에 놓이도록

Ⅰ. 목표

stage 1이 시작할 때 Snake가 Wall을 제외한 Map 위에 랜덤하게 놓이도록 하기

Ⅱ. 설계

랜덤하게 Snake의 head의 위치와 방향을 잡아주고 그에 따라 Snake 위치시키기

Ⅲ. 코드

(1) LocateSnake() 호출하기

```
void SnakeGame::LocateSnake()
{
  if(stage == 1)
  {
    srand((unsigned int)time(0));
    Randwidth = rand() % (width-6)+3;
    Randheight = rand() % (height-6)+3;
    int c=rand()%4;

    direction=' ';

    if(c==0)
    {
      mvwprintw(stdscr,Randheight,Randwidth,"3");
      mvwprintw(stdscr,Randheight,Randwidth+1,"4");
      mvwprintw(stdscr,Randheight,Randwidth+2,"4");
      xy.push(make_pair(Randheight,Randwidth+2));
      xy.push(make_pair(Randheight,Randwidth+1));
      xy.push(make_pair(Randheight,Randwidth));
      direction='l'; // left
    }
    if(c==1)
    {
      mvwprintw(stdscr,Randheight,Randwidth,"3");
      mvwprintw(stdscr,Randheight,Randwidth-1,"4");
      mvwprintw(stdscr,Randheight,Randwidth-2,"4");
      xy.push(make_pair(Randheight,Randwidth-2));
      xy.push(make_pair(Randheight,Randwidth-1));
      xy.push(make_pair(Randheight,Randwidth));
      direction='r'; // right
    }
    if(c==2)
    {
      mvwprintw(stdscr,Randheight,Randwidth,"3");
      mvwprintw(stdscr,Randheight+1,Randwidth,"4");
      mvwprintw(stdscr,Randheight+2,Randwidth,"4");
      xy.push(make_pair(Randheight+2,Randwidth));
      xy.push(make_pair(Randheight+1,Randwidth));
      xy.push(make_pair(Randheight,Randwidth));
      direction='u'; // up
    }
    if(c==3)
    {
      mvwprintw(stdscr,Randheight,Randwidth,"3");
      mvwprintw(stdscr,Randheight-1,Randwidth,"4");
      mvwprintw(stdscr,Randheight-2,Randwidth,"4");
      xy.push(make_pair(Randheight-2,Randwidth));
      xy.push(make_pair(Randheight-1,Randwidth));
      xy.push(make_pair(Randheight,Randwidth));
```

stage1인 경우에만 랜덤으로 가로와 세로의 길이를 Randwidth와 Randheight 변수에 저장하고 rand()함수를 통해 0에서 3의 숫자를 c라는 변수에 담아 이를 이용하여 방향에 따른 출력 형태를 잡아줍니다. c의 값에 따라 상하좌우에 따른 출력을 mvwprintw()를 써서 작성하고 xy queue안에 make_pair로 짝을 지어 값을 넣어 줍니다. 이로써 stage1만의 랜덤하게 출력되는 뱀의 초기위치를 잡아 주었습니다.

Ⅳ. 실행 화면

2-1 실행 화면과 동일합니다.

3-3. 랜덤으로 생기는 Hole에 빠지면 게임 종료

Ⅰ. 목표

Hole을 랜덤하게 생성하여 Snake가 Hole과 만나면 게임을 종료하도록 하기

Ⅱ. 설계

(1) Hole이 Wall과 Snake를 제외한 Map의 랜덤한 위치에 놓이도록 RandHole() 함수 작성하기

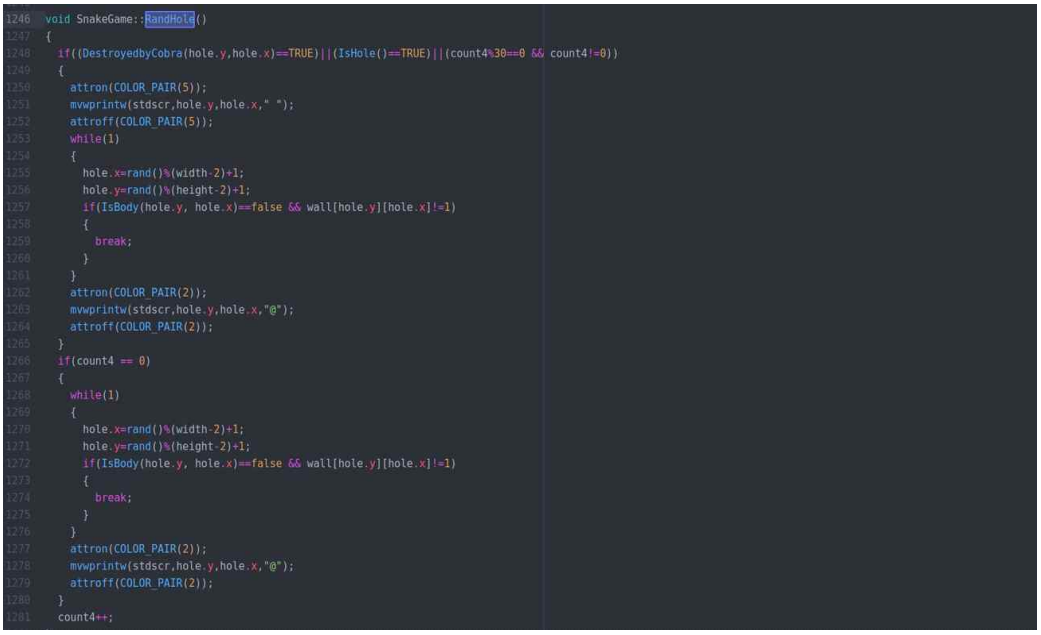
(2) Snake의 head가 Hole과 만나면 true를 반환, 만나지 않았다면 false를 반환하는 IsHole() 함수 작성하기

(3) (2)에서 true를 반환하면 게임을 종료하는 BumpedintoHole() 함수 작성하기

(4) 특정 시간이 지나면 Hole의 위치가 바뀌도록 RandHole() 함수 수정하기

Ⅲ. 코드

(1) RandHole() 함수



```
1246  void SnakeGame::RandHole()
1247  {
1248    if((DestroyedbyCobra(hole.y,hole.x)==TRUE)||(IsHole()==TRUE)||(count4%30==0 && count4!=0))
1249    {
1250      attron(COLOR_PAIR(5));
1251      mvwprintw(stdscr,hole.y,hole.x," ");
1252      attroff(COLOR_PAIR(5));
1253      while(1)
1254      {
1255        hole.x=rand()%(width-2)+1;
1256        hole.y=rand()%(height-2)+1;
1257        if(IsBody(hole.y, hole.x)==false && wall[hole.y][hole.x]!=1)
1258        {
1259          break;
1260        }
1261      }
1262      attron(COLOR_PAIR(2));
1263      mvwprintw(stdscr,hole.y,hole.x,"@");
1264      attroff(COLOR_PAIR(2));
1265    }
1266    if(count4 == 0)
1267    {
1268      while(1)
1269      {
1270        hole.x=rand()%(width-2)+1;
1271        hole.y=rand()%(height-2)+1;
1272        if(IsBody(hole.y, hole.x)==false && wall[hole.y][hole.x]!=1)
1273        {
1274          break;
1275        }
1276      }
1277      attron(COLOR_PAIR(2));
1278      mvwprintw(stdscr,hole.y,hole.x,"@");
1279      attroff(COLOR_PAIR(2));
1280    }
1281    count4++;
```

(1246-1252)

cobra의 좌푯값을 확인하는 함수 DestroyedbyCobra(hole.y, hole.x), snake의 좌푯값을 확인해주는 IsHole()함수, 실행 될 때 마다 1씩 증가하며 count4변수를 30으로 나눈 나머지가 0일 경우 총 3가지 경우를 검사의 조건으로 넣고 아래 코드를 출력하도록 합니다.

(1253-1265)

다른 아이템이나 gate와 거의 유사합니다.

rand()함수를 통해 가로 세로측의 좌푯값을 잡습니다.

IsBody()를 통해 뱀의 좌푯값을 확인하고 wall의 인덱스값으로 집어넣어 벽의 값이

나왔는지도 확인합니다. 확인하고 접촉이 있으면 다시 rand()를 돌리고 접촉이 없으면 break로 빠져나와 가로 세로 좌푯값을 mvwprintw()로 화면에 출력합니다.
(1265-1281)
아래는 위와 코드가 유사하지만 count4가 0일 경우에는 1251줄의 code처럼 “ ”으로 이전 위치의 hole을 덮어줄 필요가 없기에 나눠놓은 경우입니다.

(2) IsHole() 함수

```
1296  bool SnakeGame::IsHole()
1297  {
1298    if((hole.x==xy.back().second) && (hole.y==xy.back().first))
1299    {
1300      return TRUE;
1301    }
1302    else
1303    {
1304      return FALSE;
1305    }
1306  }
```

xy큐는 뱀의 몸과 머리가 담겨있는 큐로 큐내에서 back은 머리를 말합니다.
즉 머리와 hole의 x,y좌푯값이 다르면 false 모두 값으면 true를 리턴하는 함수입니다.

(3) BumpedintoHole() 함수

```
1284  void SnakeGame::BumpedintoHole()
1285  {
1286    clear();
1287    WINDOW * win = newwin(height, width, start.y,start.x);
1288    refresh();
1289    mvwprintw(win, 10, 5,"Game Over!!!\tFall into a Hole!!");
1290    wrefresh(win);
1291    beep();
1292    usleep(2000000);
1293    delwin(win);
1294  }
```

다른 Bumpedinto 함수들과 동일합니다. 호출되면 newwin()을 통해 새로운 WINDOW를 작성합니다. refresh()를 통해 화면 최신화 mvwprintw로 중앙에 gameover와 그 이유를 알립니다. usleep(2000000)를 통해 잠시 유지 delwin을 통해 화면 종료입니다.

Ⅳ. 실행 화면

Hole에 닿아 BumpedintoHole()이 실행된 장면입니다.

3-4. Map을 따라 도는 Cobra와 만나면 게임 종료

Ⅰ. 목표

Snake가 Map을 따라 일정하게 도는 Cobra와 만나면 게임을 종료하도록 하기

Ⅱ. 설계

(1) Cobra를 구성하는 좌표를 queue에 pair로 저장하기

(2) 각 stage가 시작할 때마다 Cobra의 초기 위치를 잡아주는 LocateCobra() 함수 작성하기

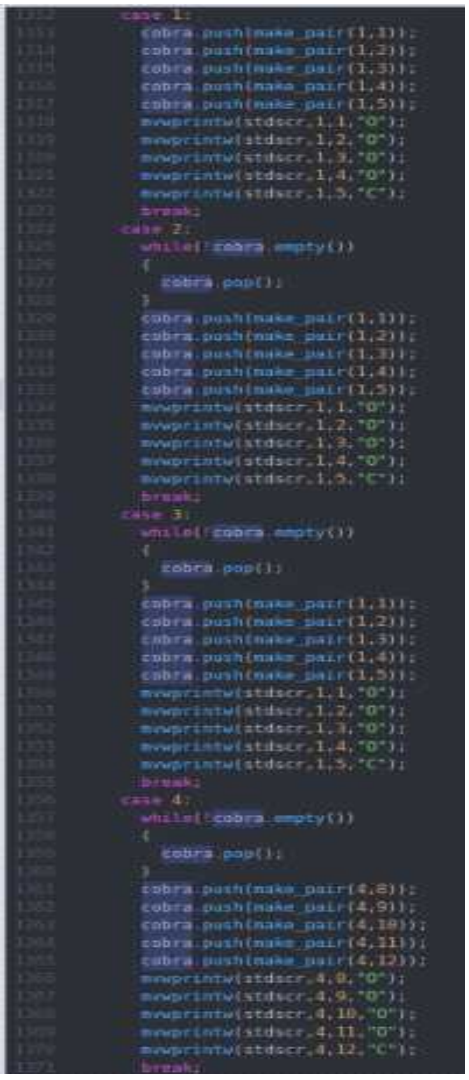
(3) 각 stage마다 Cobra가 Map을 따라 돌도록 DangerousCobra() 함수 작성하기

(4) Snake가 Cobra와 만나면 true를 반환, 만나지 않았다면 false를 반환하는 IsCobra() 함수 작성하기

(5) (4)에서 true를 반환하면 게임을 종료하는 BumpedintoCobra() 함수 작성하기

Ⅲ. 코드

(1) LocateCobra() 함수



```
case 1:
  cobra.push(make_pair(1,1));
  cobra.push(make_pair(1,2));
  cobra.push(make_pair(1,3));
  cobra.push(make_pair(1,4));
  cobra.push(make_pair(1,5));
  mvwprintw(stdscr,1,1,"O");
  mvwprintw(stdscr,1,2,"O");
  mvwprintw(stdscr,1,3,"O");
  mvwprintw(stdscr,1,4,"O");
  mvwprintw(stdscr,1,5,"C");
  break;
case 2:
  while(!cobra.empty())
  {
    cobra.pop();
  }
  cobra.push(make_pair(1,1));
  cobra.push(make_pair(1,2));
  cobra.push(make_pair(1,3));
  cobra.push(make_pair(1,4));
  cobra.push(make_pair(1,5));
  mvwprintw(stdscr,1,1,"O");
  mvwprintw(stdscr,1,2,"O");
  mvwprintw(stdscr,1,3,"O");
  mvwprintw(stdscr,1,4,"O");
  mvwprintw(stdscr,1,5,"C");
  break;
case 3:
  while(!cobra.empty())
  {
    cobra.pop();
  }
  cobra.push(make_pair(1,1));
  cobra.push(make_pair(1,2));
  cobra.push(make_pair(1,3));
  cobra.push(make_pair(1,4));
  cobra.push(make_pair(1,5));
  mvwprintw(stdscr,1,1,"O");
  mvwprintw(stdscr,1,2,"O");
  mvwprintw(stdscr,1,3,"O");
  mvwprintw(stdscr,1,4,"O");
  mvwprintw(stdscr,1,5,"C");
  break;
case 4:
  while(!cobra.empty())
  {
    cobra.pop();
  }
  cobra.push(make_pair(4,8));
  cobra.push(make_pair(4,9));
  cobra.push(make_pair(4,10));
  cobra.push(make_pair(4,11));
  cobra.push(make_pair(4,12));
  mvwprintw(stdscr,4,8,"O");
  mvwprintw(stdscr,4,9,"O");
  mvwprintw(stdscr,4,10,"O");
  mvwprintw(stdscr,4,11,"O");
  mvwprintw(stdscr,4,12,"C");
  break;
```

cobra의 좌표를 stage에 따라 좌푯값을 직접 화면에 출력해주고 queue에 push를 통해 집어넣습니다.

(2) DangerousCobra() 함수

```
void SnakeGame::DangerousCobra()
{
  switch(stage){
    case 1:
      row=27;
      col=15;
      break;
    case 2:
      row=27;
      col=6;
      break;
    case 3:
      row=34;
      col=18;
      break;
    case 4:
      row=20;
      col=12;
      break;
  }

  if(countCobra<=row)
  {
    cdirection = 'r';
  }
  else if((row+col>=countCobra)&&(countCobra > row))
  {
    cdirection = 'd';
  }
  else if((  4+row+row+col>=countCobra) && (countCobra > row+col))
  {
    cdirection = 'l';
  }
  else if(( 4+row+row+col+col>countCobra) && (countCobra > 4+row+row+col))
  {
    cdirection = 'u';
  }
  else if(countCobra % (4+row+row+col+col) ==0){
    countCobra = -4;
  }

```

```
  switch (cdirection){
    case 'r':
      attron(COLOR_PAIR(5));
      mvwprintw(stdscr,cobra.front().first,cobra.front().second," ");
      attroff(COLOR_PAIR(5));
      cobra.pop();
        attron(COLOR_PAIR(1));
      mvwprintw(stdscr,cobra.back().first,cobra.back().second,"O");
      cobra.push(make_pair(cobra.back().first, cobra.back().second+1));
      mvwprintw(stdscr,cobra.back().first,cobra.back().second,"C");
        attroff(COLOR_PAIR(1));
        break;
    case 'l':
      attron(COLOR_PAIR(5));
      mvwprintw(stdscr,cobra.front().first,cobra.front().second," ");
      attroff(COLOR_PAIR(5));
      cobra.pop();
        attron(COLOR_PAIR(1));
      mvwprintw(stdscr,cobra.back().first,cobra.back().second,"O");
      cobra.push(make_pair(cobra.back().first, cobra.back().second-1));
      mvwprintw(stdscr,cobra.back().first,cobra.back().second,"C");
        attroff(COLOR_PAIR(1));
        break;
    case 'u':
      attron(COLOR_PAIR(5));
      mvwprintw(stdscr,cobra.front().first,cobra.front().second," ");
      attroff(COLOR_PAIR(5));
      cobra.pop();
        attron(COLOR_PAIR(1));
      mvwprintw(stdscr,cobra.back().first,cobra.back().second,"O");
      cobra.push(make_pair(cobra.back().first-1, cobra.back().second));
      mvwprintw(stdscr,cobra.back().first,cobra.back().second,"C");
        attroff(COLOR_PAIR(1));
        break;
    case 'd':
      attron(COLOR_PAIR(5));
      mvwprintw(stdscr,cobra.front().first,cobra.front().second," ");
      attroff(COLOR_PAIR(5));
      cobra.pop();
        attron(COLOR_PAIR(1));
      mvwprintw(stdscr,cobra.back().first,cobra.back().second,"O");
      cobra.push(make_pair(cobra.back().first+1, cobra.back().second));
      mvwprintw(stdscr,cobra.back().first,cobra.back().second,"C");
        attroff(COLOR_PAIR(1));
        break;
      }
  countCobra++;
```

(1378-1461) stage의 값을 switch문의 조건으로 넣고 stage에 따른 가로 세로의 길이로 넣었습니다. 각 스테이지의 row와 col의 값에 따라 아래 조건의 식의 countCobra가 증가할 때마다 적용되는 범위를 제한하고 cobra의 direction이 자동으로 지정되게 해줍니다. 결국 DangerousCobra()의 호출에 따라 countCobra가 증가하고 이에 따라 cdirection이 지정되고 결정된 cdirection에 의해 controlsnake함수에 있던 방향에 따라 뱀의 움직임을 queue에 저장하고 화면에 출력해주는 코드를 재활용합니다.
(1463) countCobra를 함수의 마지막에 증가시킵니다.

(3) IsCobra() 함수

```
1479  bool SnakeGame::IsCobra()
1480  {
1481    for(int j = 0; j < cobra.size();j++)
1482    {
1483       cbody.x = cobra.front().second;
1484       cbody.y = cobra.front().first;
1485       cobra.pop();
1486
1487      for(int i=0;i<xy.size();i++)
1488      {
1489        body.x = xy.front().second;
1490        body.y = xy.front().first;
1491
1492        if(body.x==cbody.x && body.y==cbody.y)
1493        {
1494        return TRUE;
1495        }
1496        else
1497        {
1498        xy.pop();
1499        }
1500        xy.push(make_pair(body.y,body.x));
1501      }
1502      cobra.push(make_pair(cbody.y, cbody.x));
1503    }
1504    return FALSE;
1505  }
```

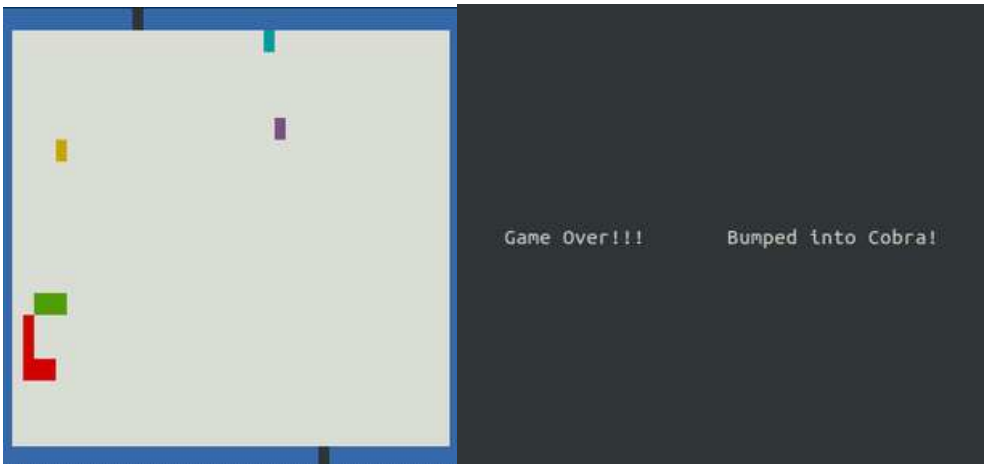
(1479-1504)
이중 for문을 통해 xy queue내의 snake 좌푯값과 cbody내의 cobra 좌푯값을 전부 비교하는 코드입니다. queue로 구현했기에 무조건적으로 pop한 값을 다시 push해야 하는 까다로움이 있습니다. 어느 한 값이라도 겹치는 좌표가 있다면 True를 리턴합니다.

(4) BumpedintoCobra() 함수

```
void SnakeGame::BumpedintoCobra()
{
  clear();
  WINDOW * win = newwin(height, width, start.y,start.x);
  refresh();
  mvwprintw(win, 10, 5,"Game Over!!!\tBumped into Cobra!");
  wrefresh(win);
  beep();
  usleep(2000000);
  delwin(win);
  stage =0;
}
```

다른 Bumpedinto 함수들과 모든 것이 동일합니다. 'Game over!!! Bumped into Cobra!'를 출력합니다.

Ⅳ. 실행 화면



Cobra에 부딪혀 gameover된 화면입니다. BumpedintoCobra()가 실행되었습니다.

3-5. Snake Game 화면에 색 입히기

Ⅰ. 목표

게임의 가시성을 높이기 위해서 색을 입혀보기

Ⅱ. 설계

attron, attroff , COLOR_PAIR 함수를 통해 필요한 위치에 색을 입히기

Ⅲ. 코드

(1) start_color() , init_pair() 함수

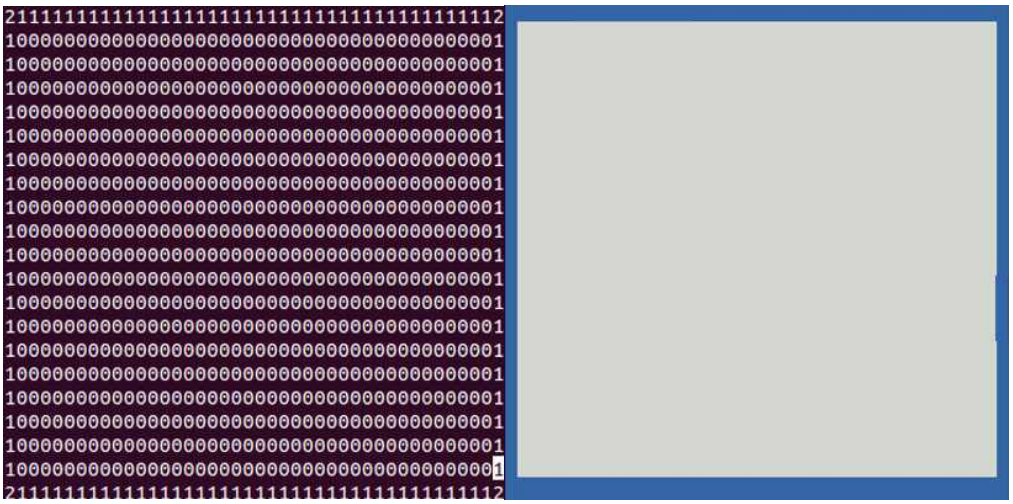
```
void SnakeGame::StartWindow()
{
  initscr(); // start ncurses mode
  start_color();

  init_pair(0, COLOR_BLACK, COLOR_BLACK);
  init_pair(1, COLOR_RED, COLOR_RED);
  init_pair(2, COLOR_CYAN, COLOR_CYAN);
  init_pair(3, COLOR_GREEN, COLOR_GREEN);
  init_pair(4, COLOR_WHITE, COLOR_WHITE);
  init_pair(5, COLOR_BLACK, COLOR_WHITE);
  // init_pair(6, COLOR_BLUE, COLOR_BLACK);
  init_pair(6, COLOR_BLUE, COLOR_WHITE);
  init_pair(7, COLOR_BLUE, COLOR_BLUE);
  init_pair(8, COLOR_MAGENTA, COLOR_MAGENTA);
  init_pair(9, COLOR_YELLOW, COLOR_YELLOW);
```

게임을 제작하기 위해 초기에 설정해주는 StartWindow()함수에서 start_color()를 통해 color 속성을 켜주고 init_pair로 원하는 팔레트를 제작하였습니다.

```
void SnakeGame::ScoreBoard()
{
  mvwprintw(stdscr,2,45,"<Score Board>");
  mvwprintw(stdscr,3,45,"B :                    ");
  mvwprintw(stdscr,4,45,"+ :                    ");
  mvwprintw(stdscr,5,45,"- :                    ");
  mvwprintw(stdscr,6,45,"G :                    ");
  attron(COLOR_PAIR(3));
  mvwprintw(stdscr,7,45,"-");
  attroff(COLOR_PAIR(3));
  mvwprintw(stdscr,7,46,": snake");
  attron(COLOR_PAIR(4));
  mvwprintw(stdscr,8,45,"-");
  attroff(COLOR_PAIR(4));
  mvwprintw(stdscr,8,46,": gate");
  attron(COLOR_PAIR(7));
  mvwprintw(stdscr,9,45,"-");
  attroff(COLOR_PAIR(7));
  mvwprintw(stdscr,9,46,": wall");
  attron(COLOR_PAIR(8));
  mvwprintw(stdscr,10,45,"-");
  attroff(COLOR_PAIR(8));
  mvwprintw(stdscr,10,46,": poison");
  attron(COLOR_PAIR(9));
  mvwprintw(stdscr,11,45,"-");
  attroff(COLOR_PAIR(9));
  mvwprintw(stdscr,11,46,": growth");
  attron(COLOR_PAIR(2));
  mvwprintw(stdscr,12,45,"-");
  attroff(COLOR_PAIR(2));
  mvwprintw(stdscr,12,46,": hole");
  attron(COLOR_PAIR(1));
  mvwprintw(stdscr,13,45,"-");
  attroff(COLOR_PAIR(1));
  mvwprintw(stdscr,13,46,": cobra");
```

위는 scoreboard()함수입니다. attron함수를 통해 컬러속성을 키고 attoff를 통해 끔니다. 두 속성이 켜지고 꺼지는 사이에 설정한 COLOR_PAIR에 의해 mvwprintw에 의한 모든 것이 색칠되게 됩니다.

Ⅳ. 실행 화면



색 넣기 전과 넣고 후의 사진입니다.

4. 결과

4-1. 실행 방법

4-2. 개선할 점

Ⅰ. tick에 의한 에러

함수의 호출에 변수를 증감시켜서 구현했더니 계산량에 의해 코드계산에 글리치가 발생하였습니다. 예를 들면 Cobra가 직사각형 안에서만 맴돌아야하는데 랜덤 아이템을 먹는 일이 반복되면 본래 가야할 길이 아닌 옆으로 이동한 사각형을 유지하게 되는 부분이 있습니다. newwin으로 구현하지 못하였습니다.
더 다채롭게 윈도우를 활용할 수도 있었겠으나 아쉽게도 이미 창을 한 개로 구현해 버린 까닭에 수정할 수가 없었습니다.
(길이 : 세로가 길고 가로가 짧습니다.)

Ⅱ. Game over된 이후의 에러

Game의 규칙에 의해 Game over가 되는 경우, 가끔 온전히 게임이 중지되지 않고 Game over 출력되는 화면에 게임이 실행되는 모습이 보이는 경우가 발생합니다.
newwin을 통해 만들어진 창이 제대로 닫히지 않아서 발생하는 것이라고 예상합니다.