2021-1 C++프로그래밍 실습과제 10

(1) 각 문제에 대한 분석과 및 해결 방법

10.1. 10.9절의 Monster World 5 프로그램을 두 사람이 경쟁하면서 게임을 할 수 있도록 다음과 같이 확장하라.

(1) 한 사람(오른쪽)은 화살표 키를 사용하고, 다른 사람(왼쪽)은 상하좌우를 위해 'w', 's', 'a', 'd'키를 사용한다.

(2) 왼쪽 사람과 오른쪽 사람의 아이콘은 각각 "좌"와 "우"로 한다.

(3) Human을 상속한 Tuman 클래스를 만들고, 인간의 움직임만을 위한 move()함수를 추가한다. 이 함수는 Human::move()과 비슷하지만 입력되는 문자에 따라 움직이는 함수이며, 함수 안에서 kbhit()를 사용하지 않아야 한다.

(4) 몬스터 월드에 일반 몬스터를 모두 추가한 후 마지막으로 두 개의 Tuman 객체를 추가한다. 이제, pMon[nMon-2]는 왼쪽 경기자가 되고, pMon[nMon-1]은 오른쪽 경기자가 된다.

[문제분석 및 해결방법] : Human을 확장하고 함수를 적절히 오버라이딩하여 Tuman 클래스를 구현하고 kbhit()과 getche() 함수를 이용하여 키 입력 핸들링을 수행하였다. MonsterWorld.h의 move 호출 로직을 수정하여 배열의 마지막 두 요소가 각각 왼쪽과 오른쪽 경기자가 될 수 있게 코드를 구성하였다.

(2) 자신이 구현한 주요 코드

10.1. 10.9절의 Monster World 5 프로그램을 두 사람이 경쟁하면서 게임을 할 수 있도록 다음과 같이 확장하라.

<Human.h>

```
#pragma once
#include "Monster.h"
#include <conio.h >
#include "DeprecatedController.h"
enum Direction { Left =75, Right =77, Up =72, Down =80 };
class Human : public Monster {
public:
  Human(string n ="미래인류", string i ="♀", int px =0, int py =0)
    : Monster(n, i, px, py) {}
  virtual ~Human() { cout <<" [Human    ]"; }
  virtual int getDirKey() { return kbhit() && getche() ==224 ? getche() : 0; }
  virtual void move(int ** map, int maxx, int maxy) {
    char ch = getDirKey();
    if (ch == Left) x --;
    else if (ch == Right) x ++;
    else if (ch == Up) y --;
    else if (ch == Down) y ++;
    else return;
    clip(maxx, maxy);
    eat(map);
  }
};
class Tuman : public Human {
protected:
  int dirKey =0;
private:
  void setDirKey(int ch) {
    dirKey = ch;
  }
public:
  Tuman(string n ="미래투맨", string i ="♂", int px =0, int py =0): Human(n, i, px,
py){ }
  ~Tuman() { cout <<" [Tuman    ]"; }
  virtual int getDirKey() { return dirKey; }
  virtual void move(int ** map, int maxx, int maxy, char ch) {
    setDirKey(ch);
    Human::move(map, maxx, maxy);
    setDirKey(0);
  }
};
```

<MonsterWorld.h>

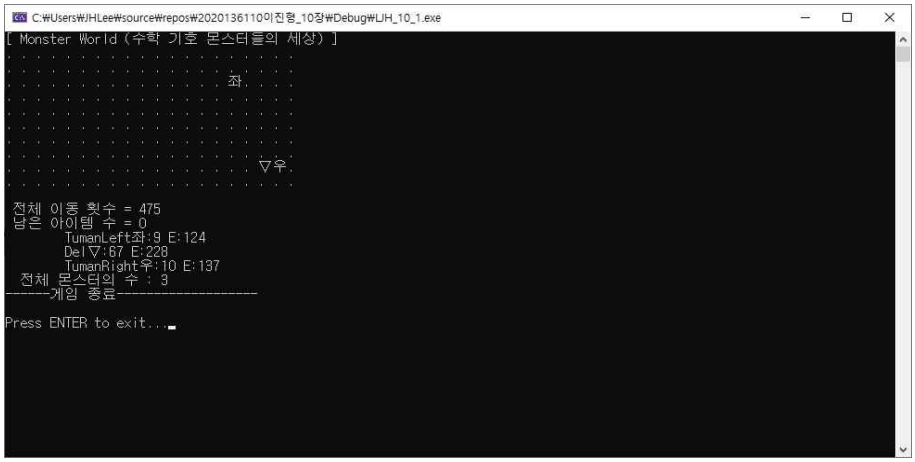
```
#pragma once
#include "Canvas.h"
#include "Monster.h"
#include "Matrix.h"
#include "Human.h"
#include <windows.h >
#include <conio.h >
#include "DeprecatedController.h"
#define DIM  40
#define MAXMONS 7
#define WASD_TO_DIRECTION(ch) (ch =='a' ? Left : (ch =='d' ? Right : (ch =='w' ? Up : (ch =='s' ? Down : ch))))
class MonsterWorld {
~... ~
  void play(int maxwalk, int wait) {
    print();
    cerr <<" 엔터를 누르세요...";
    getchar();
    for (int i =0; i < maxwalk; i ++) {
      Tuman* left =NULL;
      Tuman* right =NULL;
      for (int k =0; k < nMon; k ++) {
        pMon[k]->move(world.Data(), xMax, yMax);
        if (Tuman * tuman =dynamic_cast <Tuman *>(pMon[k])) {
          if (left ==NULL) {
            left = tuman;
          } else {
            right = tuman;
          }
        }
      }

      if (kbhit()) {
        int ch = getche();
        if (ch ==224) {
          ch = getche();
          if (right !=NULL) {
            right->move(world.Data(), xMax, yMax, ch);
          }
        }
        else {
          if (left !=NULL) {
            left->move(world.Data(), xMax, yMax, WASD_TO_DIRECTION(ch));
          }
        }
      }
      nMove++;
      print();
      checkStarvation();
      if (isDone()) break;
      Sleep(wait);
    }
  }
};
```

<LJH_10_1.h>

```
#include "MonsterWorld.h"
#include "VariousMonsters.h"
#include <time.h >
int Monster::nMonster =0;
void Monster::printCount() {
  cout <<"  전체 몬스터의 수 : "<< Monster::nMonster <<endl;
}
int main()
{
  srand((unsigned int)time(NULL));
  int w =20, h =10;
  MonsterWorld game(w, h);
  game.add(new Monster("Plus", "+", rand() % w, rand() % h));
  game.add(new Zombie("Integral", "∫", rand() % w, rand() % h));
  game.add(new Vampire("Delta", "Δ", rand() % w, rand() % h));
  game.add(new KGhost("Del", "▽", rand() % w, rand() % h));
  game.add(new Smombi("Multiply", "×", rand() % w, rand() % h));
  game.add(new Tuman("TumanLeft", "좌", rand() % w, rand() % h));
  game.add(new Tuman("TumanRight", "우", rand() % w, rand() % h));
  game.play(500, 10);
  printf("------게임 종료----------₩n");
  std::cout <<std::endl <<"Press ENTER to exit..."; fflush(stdin); getchar();
  return 0;
}
```

(3) 다양한 입력에 대한 테스트 결과

10.1. 10.9절의 Monster World 5 프로그램을 두 사람이 경쟁하면서 게임을 할 수 있도록 다음과 같이 확장하라.



(4) 코드에 대한 설명 및 해당 문제에 대한 고찰

(1) 한 사람(오른쪽)은 화살표 키를 사용하고, 다른 사람(왼쪽)은 상하좌우를 위해 'w', 's', 'a', 'd'키를 사용한다.

```
<MonsterWorld.h>
~... ~
#define WASD_TO_DIRECTION(ch) (ch =='a' ? Left : (ch =='d' ? Right : (ch =='w' ? Up : (ch =='s' ? Down : ch))))
class MonsterWorld {
~... ~
  void play(int maxwalk, int wait) {
~... ~
      if (kbhit()) {
        int ch = getche();
        if (ch ==224) {
          ch = getche();
          if (right !=NULL) {
            right->move(world.Data(), xMax, yMax, ch);
          }
        }
        else {
          if (left !=NULL) {
            left->move(world.Data(), xMax, yMax, WASD_TO_DIRECTION(ch));
          }
        }
      }
~... ~
    }
  }
};
```

getche()와 kbhit()을 이용하여 wasd 키에 대한 처리를 구현하였다.

(2) 왼쪽 사람과 오른쪽 사람의 아이콘은 각각 "좌"와 "우"로 한다.

```
<LJH_10_1.h>
#include "MonsterWorld.h"
#include "VariousMonsters.h"
#include <time.h >
int Monster::nMonster =0;
void Monster::printCount() {
  cout <<"  전체 몬스터의 수 : "<< Monster::nMonster <<endl;
}
int main()
{
  srand((unsigned int)time(NULL));
  int w =20, h =10;
  MonsterWorld game(w, h);
  game.add(new Monster("Plus", "+", rand() % w, rand() % h));
  game.add(new Zombie("Integral", "∫", rand() % w, rand() % h));
  game.add(new Vampire("Delta", "Δ", rand() % w, rand() % h));
  game.add(new KGhost("Del", "▽", rand() % w, rand() % h));
  game.add(new Smombi("Multiply", "×", rand() % w, rand() % h));
  game.add(new Tuman("TumanLeft", "좌", rand() % w, rand() % h));
  game.add(new Tuman("TumanRight", "우", rand() % w, rand() % h));
  game.play(500, 10);
  printf("------게임 종료----------₩n");
  std::cout <<std::endl <<"Press ENTER to exit..."; fflush(stdin); getchar();
  return 0;
}
```

Tuman 클래스를 구현하고 생성자의 아이콘 매개변수에 "좌"와 "우" 값을 주어 문제의 명세대로 아이콘을 설정하였다.

(3) Human을 상속한 Tuman 클래스를 만들고, 인간의 움직임만을 위한 move()함수를 추가한다. 이 함수는 Human::move()과 비슷하지만 입력되는 문자에 따라 움직이는 함수이며, 함수 안에서 kbhit()를 사용하지 않아야 한다.

<Human.h>

```
#pragma once
#include "Monster.h"
#include <conio.h >
#include "DeprecatedController.h"
enum Direction { Left =75, Right =77, Up =72, Down =80 };
class Human : public Monster {
public:
  Human(string n ="미래인류", string i ="♀", int px =0, int py =0)
    : Monster(n, i, px, py) {}
  virtual ~Human() { cout <<" [Human    ]"; }
  virtual int getDirKey() { return kbhit() && getche() ==224 ? getche() : 0; }
  virtual void move(int ** map, int maxx, int maxy) {
    char ch = getDirKey();
    if (ch == Left) x --;
    else if (ch == Right) x ++;
    else if (ch == Up) y --;
    else if (ch == Down) y ++;
    else return;
    clip(maxx, maxy);
    eat(map);
  }
};
class Tuman : public Human {
protected:
  int dirKey =0;
private:
  void setDirKey(int ch) {
    dirKey = ch;
  }
public:
  Tuman(string n ="미래투맨", string i ="♂", int px =0, int py =0): Human(n, i, px,
py){ }
  ~Tuman() { cout <<" [Tuman    ]"; }
  virtual int getDirKey() { return dirKey; }
  virtual void move(int ** map, int maxx, int maxy, char ch) {
    setDirKey(ch);
    Human::move(map, maxx, maxy);
    setDirKey(0);
  }
};
```

오버로드된 move(int ** map, int maxx, int maxy, char ch) 함수를 만들면서 최대한 기존의 코드를 재활용하기 위해 Human의 getDirKey()와 move를 virtual 함수로 변경하고, kbhit() 조건문을 getDirKey()에 통합시키고 이를 통해 Tuman의 오버로드된 move 함수에서 Human::move(int ** map, int maxx, int maxy)를 재사용 가능하게 하였다.

(4) 몬스터 월드에 일반 몬스터를 모두 추가한 후 마지막으로 두 개의 Tuman 객체를 추가한다. 이제, pMon[nMon-2]는 왼쪽 경기자가 되고, pMon[nMon-1]은 오른쪽 경기자가 된다.

<MonsterWorld.h>

```
~... ~
class MonsterWorld {
~... ~
  void play(int maxwalk, int wait) {
~... ~
    for (int i =0; i < maxwalk; i ++) {
      Tuman* left =NULL;
      Tuman* right =NULL;
      for (int k =0; k < nMon; k ++) {
        pMon[k]->move(world.Data(), xMax, yMax);
        if (Tuman * tuman =dynamic_cast <Tuman *>(pMon[k])) {
          if (left ==NULL) {
            left = tuman;
          } else {
            right = tuman;
          }
        }
      }

      if (kbhit()) {
        int ch = getche();
        if (ch ==224) {
          ch = getche();
          if (right !=NULL) {
            right->move(world.Data(), xMax, yMax, ch);
          }
        }
        else {
          if (left !=NULL) {
            left->move(world.Data(), xMax, yMax, WASD_TO_DIRECTION(ch));
          }
        }
      }
~... ~
    }
  }
};
```

기존의 checkStarvation 함수와 호환성을 유지하기 위해서 dynamic_cast를 이용해 pMon 포인터 배열에 들어있는 Monster들의 상세 형 검사를 실시하여 Tuman인 경우, 배열의 앞쪽에 있는 것이 left Tuman, 뒤쪽에 있는 것이 right Tuman이 되게끔 설정함으로써 몬스터가 굶어죽을 때 메모리 해제 후 자리 스왑을 실시하는 checkStarvation

함수의 로직에 상관없이 Tuman 로직이 작동하도록 구현하였다.

(5) 이번 과제에 대한 느낀점

이번 과제를 해결하기 위해 교재의 코드를 분석하고 조사하면서 모르고 있었던 세부적인 내용을 많이 알 수 있어서 좋았다.