객체지향프로그래밍 중간고사 보고서

B877005 김완일

■Screen 구조체

1)변수 : 추가한 변수 없음.

2)함수 : Screen의 size값을 return하는 int getSize()함수만 추가함.

■Player 구조체

1)변수

- char initFace[20] : player의 face가 원래대로 돌아올 때를 위해 초기 face를 저장
- char smileFace[20],deadFace[20]: 각각 enemy를 hit했을 때, player가 죽었을 때 face를 바꾸기 위해 각각 문자열 (^_^), (X_X)를 저장함.
- char hpPrinter[8] : int형 HP값 을 screen에 draw하기 위한 문자열로, snprintf(hpPrinter, "%d", hp);를 사용하여 int형 HP값을 문자열로 변환하여 저장함
- int nHitRemaining,nHpRemaining : 각각 player의 face가 바뀌는 시간과 HP값이 screen에 출력되는 시간을 저장하기 위해 사용됨.
- Int nFlickerInterval : player의 HP가 10이하가 됐을 때 깜빡이는 간격을 정하기 위해 사용됨

2)함수

- void update(bool* is Looping) :
  - player가 죽은 경우 : face를 deadFace로 바꾸고 main함수의 isLooping을 false로 바꾸어 main함수의 while문에서 빠져나오게 되고 게임종료가 되게함.
  - Player가 살아있는 경우 :
    1) checkFlicker()함수를 통해 HP와 nFlickerInterval을 확인하여 player가 깜빡여야 하는 지 확인함
    2) checkHit()함수를 통해 nHitRemaining을 확인하여 face를 바꿔야 하는지

확인함

3) nHpRemaining을 확인하여 HP값을 표시해야 하는지 확인함.

- void draw(Screen* screen)
  - face를 screen에 draw함.
  - nHpRemaining>0인 경우 HP값을 screen에 표시함
  - hp<=10, nFlickerInterval==0인 경우에만 face의 크기만큼 ' '을 screen에 draw해서 깜빡임을 나타냄.
- void fire(Bullets* bullets, Enemy* enemy)
  - bullets의 find_unused_bullet()을 이용해 발사할 bullet을 찾음.
  - 발사할 bullet을 찾지 못하면 return함.
  - 발사할 bullet을 찾으면 그 bullets의 setFire()함수를 실행하여 bullet의 isReady, isFire, direction값을 결정함.
- bool isBeHit(Enemy * enemy)
  - enemy의 위치와 크기를 받아와서 enemy가 1칸이내에 있는 지 확인하기 위한 함수
- void beHit()
  - player가 enemy한테 hit당했을 때 1번의 게임루프가 0.1초(100ms)이므로, 1초에 10을 달게 하기 위해 hp를 1깎음. nHpRemaining=5로 설정해서 HP가 screen에 표시되게 함.
- void checkHit()
  - enemy를 hit했을 때 nHitRemaining변수를 통해 smileFace를 initFace로 바꿈.
- Bool isHpLack()
  - Hp<=10일 때 true를 return함. (hp가 10 이하일 때 깜빡이는 기능을 구현하기 위해)
- Void checkFlicker()
  - Hp<=10일 때, nFlickerInterval이 0이면 nFlickerInterval=hp+1로 설정하여 hp

가 낮아질수록 깜빡이는 속도를 빨라지게함. nFlickerInterval이 0이 아닌 경우 --nFlickerInterval을 함.

- Bool isDie()
  - Hp<=0인 경우 true를 return함

■Enemy 구조체

1)변수

- char initFace[20] : enemy의 face가 원래대로 돌아올 때를 위해 초기 face를 저장
- char cryFace[20],deadFace[20]: 각각 enemy가 hit당했을 때, enemy가 죽었을 때 face를 바꾸기 위해 각각 문자열 (T_#), (X_#)를 저장함.
- char hpPrinter[8] : int형 HP값 을 screen에 draw하기 위한 문자열로, snprintf(hpPrinter, "%d", hp);를 사용하여 int형 HP값을 문자열로 변환하여 저장함
- int initPos,maxHp : enemy 부활 기능 구현을 위해 enemy의 초기 값을 저장함.
- int nBeHitRemaining,nHpRemaining,nDieRemaining : 각각 enemy가 hit당했을 때 face가 바뀌는 시간, HP값이 screen에 표시되는 시간, enemy가 죽었을 때 screen에 남아있는 시간을 저장하기 위해 사용됨.
- Int nMovementInterval, nReSpawnInterval : 각각 enemy가 player쪽으로 이동하는 간격, enemy가 부활하는 간격을 정하기 위해 사용됨

2)함수

- void update(Player* player) :
  - enemy가 죽은 경우
    1) --nReSpawnInterval을 해서 부활 간격을 줄이고 nReSpawnInterval==0이 된 경우 부활하게함.
    2) nDieRemaining변수를 통해 죽은 후 0.5초 동안 screen에 남아있게함.
  - enemy가 살아있는 경우 :
    1) chase(player)함수를 통해 1초당 한칸씩 player를 향해 이동함.

2) nHitRemaining을 확인하여 face를 바꿔야 하는지 확인함.

3) nHpRemaining을 확인하여 HP값을 표시해야 하는지 확인함.

- Void draw(Screen* screen)
  - Enemy가 죽고 nDieRemaining>0인 경우 또는 enemy가 살아있는 경우 face를 screen에 draw함.
  - nHpRemaining>0인 경우 HP값을 screen에 표시함
- void chase(Player* player)
  - nMovementInterval >0인 경우 : --nMovementInterval을 함
  - nMovementInterval ==0인 경우
    1. nMovementInterval을 10으로 바꿈.
    2. Player위치 + player크기<enemy 위치일 때 : 왼쪽으로 1칸 이동함.
    3. enemy위치 + enemy크기<player 위치일 때 : 오른쪽으로 1칸 이동함.
- bool isBeHit(Bullet * bullet)
  - bullet의 위치와 방향을 받아와서 enemy가 bullet에 맞았는지 확인하기 위한 함수
  - bullet위치 == enemy위치로 돼있을 때 간헐적으로 bullet이 enemy를 관통하는 버그가 있어서 등호를 부등호로 고친 뒤 bullet.isFire==true일 때 한번만 데미지를 입게 하고, bullet이 enemy에 맞았을 때 bullet.isFire을 false로 바꾸어 버그를 수정함.
- void beHit()
  - bullet.isFire가 false이거나 enemy가 죽어있는 경우 return(죽어있을 땐 총알에 맞지 않음)
  - hp를 5깎고 face,nHpRemaining,nBeHitRemanining변수들을 수정함.
  - 이 때 hp가 0이 된 경우 die()함수 실행
- void checkBeHit()
  - bullet에 hit당하고 시간이 지났을 때 nBeHitRemaining변수를 통해 cryFace를

initFace로 바꿈.

- Bool isDie()
  - Hp<=0인 경우 true를 return함
- void die()
  - nReSpawnInterval= (rand()%6+5)*10 -> 50~100의 값을 저장, 5~10초 사이에 랜덤하게 부활하게 함.
  - nDieRemaining=5 ->죽은 후에도 0.5초간 화면에 남아있게 함
  - hp가 -가 될 수도 있으니 hp를 0으로 수정, face를 deadFace로 바꿈.
- void reSpawn()
  - Enemy의 face, pos, hp를 초기 값으로 초기화시킴.
- void checkReSpawn()
  - isDie()==true인 경우에만 –nReSpawnInterval을 해서 nReSpawnInterval이 0이 되면 reSpawn()함수를 실행해서 enemy가 부활하게 함.

■Bullet 구조체

1)변수

- bool isFire : bullet의 발사상태를 저장함. screen에서 발사 중인 경우 true, screen 밖에 나가거나 enemy에 적중하면 false가 됨.

2)함수

- void update(Player* player, Enemy* enemy, Screen* screen)
  - isFire==true일 때만 실행함. False인 경우 return함.
  - 발사할 때 Enemy가 있던 방향으로 1칸 이동함.
  - enemy에게 안맞고 screen밖으로 나간 경우 : isFire을 false로 바꾸고 return함.
  - enemy에게 안맞고 screen밖으로 안나간 경우 : 아무것도 안하고 return함.

- enemy에게 맞은 경우
  1) enemy가 bullet에 맞는 함수 실행 (enemy의 beHit함수)
  2) player가 enemy에게 bullet을 맞힌 함수 실행 (player의 onEnemyHit함수)
  3) isFire을 false로 바꿈.

- Void draw(Screen* screen)
  - isFire==true인 경우에만 screen에 draw함.
- void setFire(Player* player, Enemy* enemy)
  - isReady를 false로, isFire을 true로 바꾸고 기본 direction을 오른쪽으로 설정함
  - enemy가 죽은 경우 : return함
  - enemy가 죽지 않은 경우
    1) player위치>enemy위치 인 경우 direction을 왼쪽으로 설정함.
    2) 발사 위치를 player의 위치로 설정함.
    3) Direction이 오른쪽인 경우 그 위치에서 player크기만큼 뺀 위치로 설정해서 bullet과 player가 겹치지 않게 함.
- Bool isFiring()
  - isFire값을 return함. Enemy가 bullet과 충돌을 판정할 때, bullets에서 다 안쓴 탄창을 찾을 때 사용됨.

■Bullets 구조체(탄창)

1)변수

- int nReloadRemaining: 새로운 탄창으로 교체됐을 때 "탄창 교체"메시지를 표시하는 시간을 저장하기 위해 사용됨.
- int nReloadInterval: 총알을 다 쏘고 새로운 탄창이 준비되는 간격을 저장하기 위해 사용됨.
- Bullet* allBullets[3] : allbulets[0]이 기본 탄창이고, allBulets[1]~allBulets[2]가 여분

탄창임. 장전 시간이 10초보다 짧아질 경우 여분 탄창이 여러 개 필요할 수 있기 때문에 여분 탄창을 배열로 설정함.

- Bullet* myBullets : 현재 player가 사용할 탄창을 저장함. allBullets[0]이 default값임.

2)함수

- void update(Player* player, Enemy* enemy, Screen* screen)
  - checkReload()함수를 실행해서 탄창을 교체해야 하는 지 확인하고 교체해야 하면 교체함.ff
  - 발사중인 bullet들을 update함.
  - nReloadRemaining을 확인하여 "탄창 교체"메시지를 표시해야 하는지 확인함.
- Void draw(Screen* screen)
  - 발사 중인 bullet들을 draw함.
  - nReloadRemaining>0인 경우 탄창 교체"메시지를 screen에 표시함.
- void reload()
  - nReloadRemaining=10으로 하여 탄창 교체 메시지가 1초동안 표시되게 함
  - nReloadInterval=100으로 하여 10초 후에 새로운 탄창이 준비되게 함.
  - spareBullets에 find_unused_bullets()함수로 교체할 새로운 탄창을 임시로 저장함.
  - 현재 쓰고 있던 다 쓴 탄창의 bullet들의 isReady값을 true로 바꿈
  - myBullets에 spareBullets에 저장해놨던 새로운 탄창을 저장함.
- void checkReload()
  - 사용중인 탄창의 첫번째 bullet의 isReady==true이면 reload를 확인할 필요가 없으므로 return함.
  - nReloadInterval>0인 경우 : --nReloadInterval을 하고 return함.
  - nReloadInterval==0인 경우

1) 사용중인 탄창 myBullets의 두번째 bullet~마지막 bullet들을 확인해서 isReady가 true인 경우, 즉 아직 쓰지 않은 bullet이 남아 있으면 return 함.

2) 함수 실행 중 return되지 않고 끝까지 오면 탄창이 준비되는 시간 10초가 지나고, 사용중인 탄창의 모든 bullet을 사용한 경우이므로 reload()함수를 실행해서 탄창을 교체함.

- Bullet* find_unused_bullet()
  - myBullets==nullptr이면 nullptr을 return함.
  - myBullets의 bullet중 isReady가 true이고 isFire가 false인 bullet을 찾아 그 주소를 return함.
  - return할 bullet을 찾지 못하면 nullptr을 return함.
- Bullet* find_unused_bullets()
  - 탄창을 교체할 때 비어있는 탄창을 우선 사용할 수 있도록 빈 탄창을 return하는 함수임.
  - 탄창의 첫번째 총알의 isReady가 true이고 isFire가 false이면 그 탄창은 사용할 준비가 된 탄창임.
  - 모든 탄창 중에서 탄창의 첫번째 총알의 isReady가 true이고 isFire가 false인 탄창을 찾아서 그 탄창의 주소(탄창의 첫번째 총알의 주소)를 return함.
  - return할 탄창을 찾지 못하면 nullptr을 return함.