전자공학프로그래밍 C 프로젝트 보고서

> 전자공학과
> 201820908 오병준
> 개발 환경 : VS 2019

1. 개발 요구 사항

산성비 게임 개발하기

1. 기본 구현 사양

- 파일 입출력을 통한 단어 목록 입력
- 일정 주기로 떨어지는 단어 생성
- 사용자 입력과 일치하는 단어 삭제
- 단어 하강 속도 및 생성 주기에 따른 점수 계산
- 점수에 따른 게임 속도 계산

2. 구현 요구 사양

- `정지` 입력 시 3초간 생성 및 하강 중지
- `폭탄` 입력 시 화면의 모든 단어 제거
- 단어가 화면 하단에 닿으면 체력 감소
- 체력 소진 시 게임 종료

3. 추가 구현 사양

- 잔여 체력을 하단에 인디케이터로 표시하여 직관적으로 파악
- 입력하면 체력을 회복하는 보너스 단어
- 보너스 단어의 색상 변경

4. 프로그램 구조

1. 구현 기능 관련 매크로 변수

1. `PAUSE` : `정지` 기능 이용 가능 횟수 초기값
2. `PAUSE_TIME` : `정지` 기능 사용 시 정지 시간
3. `BOMB` : `폭탄` 기능 이용 가능 횟수 초기값
4. `LIFE` : 체력 초기값(3에서 10으로 수정)

2. 구현 기능 관련 구조체

1. RAINDROP
    i. char* `word` : 단어 텍스트 저장
    ii. int `x`, `y` : 단어 위치 좌표
    iii. int `length` : 단어 길이
    iv. int `period` : 단어 갱신주기
    v. clock_t `lastUpdatedTime` : 최신 갱신 시간
    vi. char `isBonus` : 체력 회복 여부 → 추가 기능에 이용
2. RAINDROP_LIST
    1. RAINDROP `raindrops` : 화면에 존재하는 단어 목록
    1. int `cntRaindrop` : 화면에 존재하는 단어 개수

3. 구현 기능 관련 함수

1. `printWord()`
   문자열을 지정한 좌표의 함수에 출력한다.
    - 반환형 : **void**

2. `removeRaindropFromConsol(RAINDROP_LIST * raindropList, int idx)`
   특정 인덱스의 단어를 화면에서 제거한다.
    - 반환형 : **void**

3. `removeRaindropFromList(RAINDROP_LIST * raindropList, int idx)`
   특정 인덱스의 단어를 저장된 목록에서 제거하고 뒤의 단어 인덱스를 한 칸씩 당긴다.
    - 반환형 : **void**

4. `addRaindrop(RAINDROP_LIST * raindropList, WORD_LIST wordList, clock_t time)`
   새 낙하 단어를 생성한다.
    - 반환형 : **void**

4. 구현 기능 관련 변수

- 기존 변수
    i. int `speed` : 백분율 단어 낙하 속도
    ii. INPUT_WORD `input` : 키보드 입력
    iii. RAINDROP_LIST `raindropList` : 존재하는 낙하 단어 목록
    iv. int `score` : 현재 점수
    v. char `run` : 게임 진행 여부
    vi. int `remainPause` : 남은 단어 낙하 정지 횟수
    vii. int `remainBomb` : 남은 단어 전체 제거 횟수
    viii. int `remainLife` : 남은 체력

- 구현 기능 변수
    i. char `isPause` : 단어 낙하 정지 활성화 여부
    ii. clock_t `pauseStart` : 단어 낙하 정지 활성화 시작 시간

2. 코드 분석

1. `main()`

게임 진행 여부 결정 단계

```
while (run) { ... }
```

`run` 플래그가 참인 동안 낙하하는 단어 관리 반복

단어 입력 후 특수 명령어 검출 단계

```
if (strcmp(input.word, "종료") == 0) {
  pthread_cancel(thread);
  break;
}
if (!strcmp(input.word, "폭탄") && remainBomb) {
  for (int j = 0; j < 3; j++) {
    for (int i = 0; i < raindropList.cntRaindrop; i++) {
      removeRaindropFromConsol(&raindropList, i);
      removeRaindropFromList(&raindropList, i);
    }
  }
  sprintf(tempstring, "SCORE: %8d", score);
  printWord(SCORE_X, DESCRIPTION_Y, tempstring, FALSE);
  sprintf(tempstring, "SPEED: %5d%%", speed);
  printWord(SPEED_X, DESCRIPTION_Y, tempstring, FALSE);
  remainBomb--;
}
if (!strcmp(input.word, "정지") && remainPause) {
  isPause = !isPause;
  pauseStart = getClock();
  remainPause--;
}
```

입력한 단어와 특수 명령어가 일치하는지 검출하여 동작 수행

- 입력한 단어가 `종료`이면 게임을 종료한다.
- 입력한 단어가 `폭탄`이면 화면에 존재하는 모든 단어와 그 목록을 제거한다.
- 입력한 단어가 `정지`이면
    i. 낙하 정지 활성화 플래그를 참으로 설정한다.
    ii. 정지 시작 시간을 기록한다.
    iii. 낙하 정지 사용 가능 횟수를 1 감소시킨다.

입력한 단어의 화면 존재 여부 검출 단계

```
do {
  idx = findRaindropIdx(raindropList, input.word);                                    //빗방울중
  if (idx != -1) {
    score += calculateScore(raindropList.raindrops[idx], speed);              //점수 계산
    sprintf(tempstring, "SCORE: %8d", score);
    printWord(SCORE_X, DESCRIPTION_Y, tempstring, FALSE);
    speed = 100 + score / 100;
    sprintf(tempstring, "SPEED: %5d%%", speed);
    printWord(SPEED_X, DESCRIPTION_Y, tempstring, FALSE);
    if (raindropList.raindrops[idx].isBonus && remainLife < LIFE) remainLife++;
    removeRaindropFromConsol(&raindropList, idx);                                     //일치한 단
    removeRaindropFromList(&raindropList, idx);                                       //일치한 단
  }
} while (idx != -1);
```

1. 입력한 단어를 화면에 존재하는 단어 중에서 찾는다.

- 인덱스가 -1이 아니라면, 즉 일치하는 단어가 존재한다면,

i. 해당 단어의 점수를 계산에 총점에 합산한다.
ii. 속도 증분을 계산해 게임에 반영한다.
iii. 점수 및 속도 디스플레이를 업데이트한다.
iv. **추가 기능** → 해당 단어가 체력 회복 단어라면, 체력을 1 회복시킨다.
체력이 최대치라면 회복하지 않는다.
v. 입력한 단어를 화면 및 단어 목록에서 제거한다.

단어(빗방울 제어) 단계

```
if (!isPause) { ... }
```

단어 낙하 정지가 활성화 상태가 아닐 경우만 진행한다.

```
for (itr = 0; itr < raindropList.cntRaindrop; itr++) {
  if (raindropList.raindrops[itr].lastUpdatedTime + (double)raindropList.raindrops[itr].period / speed * 1(
    raindropList.raindrops[itr].lastUpdatedTime = currentTime;
    removeRaindropFromConsol(&raindropList, itr);
    raindropList.raindrops[itr].y++;
    if (raindropList.raindrops[itr].y >= MAX_Y) {
      pthread_cancel(thread);
      remainLife--;
      removeRaindropFromList(&raindropList, itr);
    }
    else printWord(raindropList.raindrops[itr].x, raindropList.raindrops[itr].y, raindropList.raindrops[itr
  }
}
```

1. 목록의 모든 단어에 대해, 단어 위치를 아래로 한 줄씩 내린다.
2. 만약 단어가 y방향 최대 위치에 도달하면, 체력을 1 줄이고 단어를 목록에서 제거한다.
3. 단어가 화면 하단에 닿지 않았다면, 단어를 화면에 출력한다.

```
if (lastRaindropTime + (double)GENERATE_PERIOD / speed * 100 <= currentTime && raindropList.cntRaindrop < M
  lastRaindropTime = currentTime;
  addRaindrop(&raindropList, wordList, currentTime);
}
```

단어 낙하 개시 주기가 돌아오고 최대 낙하 단어 개수를 초과하지 않는다면, 낙하하는 단어를 1개 추가한다.

```
else if (currentTime - pauseStart > PAUSE_TIME) {
  isPause = !isPause;
}
```

단어 낙하 정지가 활성화 상태인 경우, 시작 시간으로부터 3000ms가 경과하면 낙하 정지 활성화 상태를 반전한다.

게임 진행 여부 결정 단계

```
if (!remainLife) run = FALSE;
```

남은 체력이 없다면, `run` 플래그를 FALSE로 전환하여 게임을 종료한다.

추가 기능 → 화면 하단 체력 인디케이터 출력 단계

```
for (itr = 0; itr < remainLife * 8; itr++)
  printWord(itr++, MAX_Y, "■", FALSE);
for (itr = remainLife * 8; itr < 80; itr++)
  printWord(itr, MAX_Y, " ", FALSE);
sprintf(tempstring, "남은 체력: %2d 남은 정지 횟수: %2d 남은 폭탄 횟수: %2d", remainLife, remainPause, remainI
```

1. 체력 1 당 ■ 네 개로 화면 하단에 체력 바를 표시한다.
2. 남은 체력, 정지, 폭탄 개수 디스플레이를 갱신한다.

2. `printWord()`

추가 기능 - 체력 회복 단어의 색상 표시 기능

```
gotoxy(x, y);
if (isColor) SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 10);
printf("%s", word);
SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 7);
gotoxy(MAX_X, DESCRIPTION_Y);
wait(2);
```

`printWord()` 의 파라미터로 `isColor` 변수를 추가로 받는다.
`isColor` 변수가 `TRUE` 라면, 해당 출력의 색상을 녹색으로 변경한다.
`isColor` 변수는 해당 빗방울 단어의 `isBonus` 변수가 참인지 여부로 결정된다.

3. `removeRaindropFromList()`

```
void removeRaindropFromList(RAINDROP_LIST * raindropList, int idx) {
        int itr;
        free(raindropList->raindrops[idx].word);
        for (itr = idx; itr < raindropList->cntRaindrop - 1; itr++)
        {
                raindropList->raindrops[itr].lastUpdatedTime = raindropList->raindrops[itr + 1].lastUpdatec
                raindropList->raindrops[itr].length = raindropList->raindrops[itr + 1].length;
                raindropList->raindrops[itr].period = raindropList->raindrops[itr + 1].period;
                raindropList->raindrops[itr].x = raindropList->raindrops[itr + 1].x;
                raindropList->raindrops[itr].y = raindropList->raindrops[itr + 1].y;
                raindropList->raindrops[itr].word = raindropList->raindrops[itr + 1].word;
                raindropList->raindrops[itr].isBonus = raindropList->raindrops[itr + 1].isBonus;
        }
        raindropList->cntRaindrop--;
}
```

1. 전달받은 인덱스의 포인터 메모리를 할당 해제한다.
2. 전달받은 인덱스부터 (저장된 최대 인덱스 - 1)까지의 모든 요소에 대하여, 한 칸씩 인덱스를 앞으로 당긴다.
3. 저장된 최대 인덱스를 1 감소시킨다.

추가 기능용 코드 수정

```
raindropList->raindrops[itr].isBonus = raindropList->raindrops[itr + 1].isBonus;
```

체력 회복 여부를 결정하는 `isBonus` 변수 또한 다른 요소와 마찬가지로 인덱스를 변경한다.

4. `addRaindrop()`

```
void addRaindrop(RAINDROP_LIST * raindropList, WORD_LIST wordList, clock_t time) {
        int idx, tempInt;
```

```
        idx = rand() % wordList.cntWord;
        raindropList->raindrops[raindropList->cntRaindrop].lastUpdatedTime = time;
        raindropList->raindrops[raindropList->cntRaindrop].period = rand() % (DROP_PERIOD_MAX - DROP_PERIOD
        raindropList->raindrops[raindropList->cntRaindrop].x = rand() % MAX_X + 1;
        raindropList->raindrops[raindropList->cntRaindrop].y = MIN_Y;
        raindropList->raindrops[raindropList->cntRaindrop].isBonus = ((rand() % 5) == 3);


        tempInt = strlen(wordList.words[idx]);
        raindropList->raindrops[raindropList->cntRaindrop].word = (char*)malloc(sizeof(char) * tempInt + 1)
        strcpy(raindropList->raindrops[raindropList->cntRaindrop].word, wordList.words[idx]);
        raindropList->raindrops[raindropList->cntRaindrop].length = tempInt;
        raindropList->cntRaindrop++;
}

int calculateScore(RAINDROP raindrop, int speed) {
        return (speed * (DROP_PERIOD_MAX - raindrop.period + 100)) / 100;
}
```

1. 저장된 단어 목록 중 한 개의 인덱스를 무작위로 선택한다.
2. 단어 목록의 마지막 인덱스의 요소에 값을 저장한다.

- lastUpdatedTime 에 현재 시간을 저장한다.
- period 에 업데이트 주기를 범위 내에서 무작위로 결정하여 저장한다.
- x 에 단어의 낙하 위치를 무작위로 저장한다.
- y 에 단어의 낙하 시작 위치를 저장한다.
- word 에 단어 길이만큼 변수를 할당하여 저장한다.

1. 저장된 최대 인덱스를 1 증가시킨다.

추가 기능용 코드 수정

```
raindropList->raindrops[raindropList->cntRaindrop].isBonus = ((rand() % 5) == 3);
```

단어 빗방울 생성 시, 체력 회복 기능을 제어하는 `isBonus` 파라미터가 20%의 확률로 참이 되도록 설정한다.

---

5. `setWordList()`

```
void setWordList(WORD_LIST * wordList) {
        FILE* fp;
        int itr;
        int tempInt;
        char tempString[MAX_LENGTH_OF_STRING + 1];

        fp = fopen(FILE_NAME, "r");
        fscanf(fp, "%d", &wordList->cntWord);
        wordList->words = (char**)malloc(sizeof(char*) * wordList->cntWord);
        for (itr = 0; itr < wordList->cntWord; itr++)
        {
                fscanf(fp, "%s", tempString);
                tempInt = strlen(tempString);
                wordList->words[itr] = (char*)malloc(sizeof(char) * tempInt + 1);
                strcpy(wordList->words[itr], tempString);
        }
        fclose(fp);
        return;
}
```

---

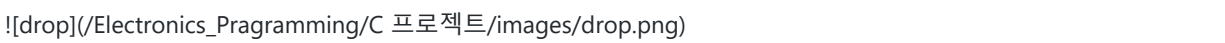
1. 단어 목록이 저장된 파일을 읽기 모드로 연다.

2. 맨 첫 줄에 있는 단어의 개수를 읽어와 `wordList`의 `cntWord`에 저장한다.
3. `cntWord` 변수의 값만큼 단어를 한 줄씩 읽어와 `wordList`의 단어 목록 `words[]` 배열에 하나씩 저장한다.
4. 파일 포인터를 닫는다.

---

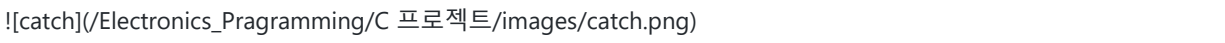
3. 실행 결과 분석

I. 단어 빗방울 강하



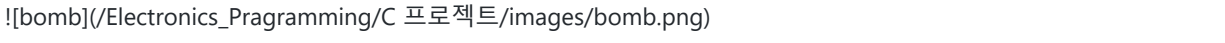
게임을 시작하면 단어가 떨어지기 시작한다.

II. 단어 제거



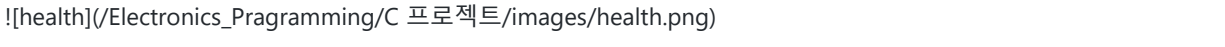
화면에 낙하하는 단어와 일치하는 단어를 입력하면 해당 단어를 제거한다.

III. 폭탄 기능



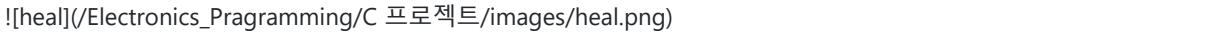
`폭탄` 명령어를 입력하면 화면의 모든 단어를 제거하나, 점수는 증가하지 않는다.

IV. 체력 감소



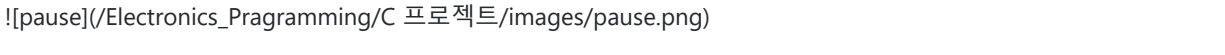
단어가 화면 하단에 닿으면 체력이 1 줄고, 체력 인디케이터가 감소한다.

V. 체력 회복



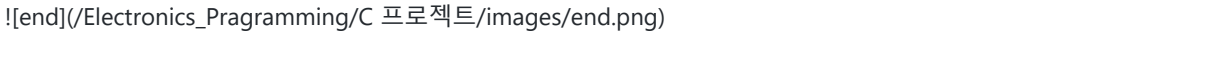
초록색으로 표시되는 체력 회복 단어를 입력하면, 체력을 1 회복하고 체력 인디케이터가 증가한다.

VI. 낙하 중지



`정지` 명령어를 입력하면 화면의 모든 단어가 낙하를 중지하고, 새 낙하 단어를 생성하지 않는다.

VII. 게임 종료



체력이 0이 되면 게임을 종료한다.

4. 전체 코드