

타이틀 : 아무 키나 눌러서 메인메뉴로 진입 가능

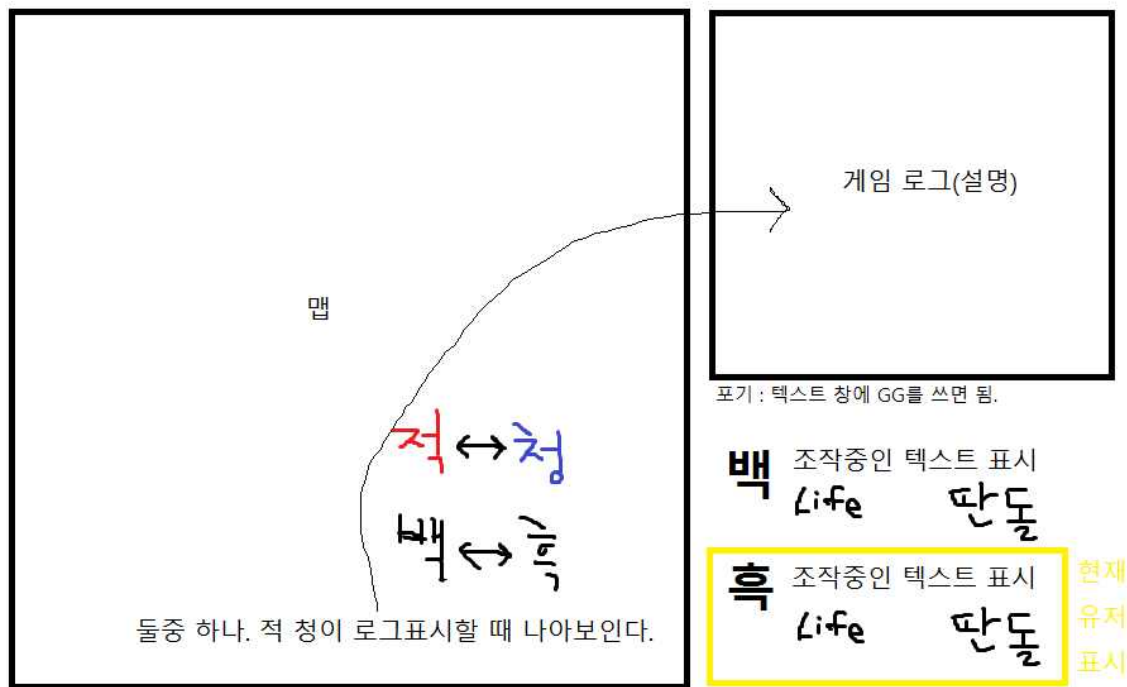
메인메뉴 : 1. 게임시작(게임설명) 2.게임 제작자 정보, 블로그 3.게임 옵션 설정

- 1)번호에 맞는 숫자 키를 누르면 바로 화면을 바꾼다.
- 2)방향키를 누르면 최소 숫자 항목의 배경색을 바꾼다.
- 그 상태에서 엔터를 누르면 원하는 화면으로 간다.

## 2.게임 제작자에 대한 정보를 볼 수 있다.

제작자의 블로그에 접속 가능하다.

## 3.게임 옵션을 설정할 수 있다(아직 미지원)



## 1.게임 설명을 볼 수 있다

게임 설명은 실제 게임의 미니게임 맵을 이용하며 게임의 규칙을 순서대로 설명한다.  
//요약하자면 미니게임 모드 그대로 갖다 붙이는 수준이다.

### 맵에 대한 설명

말이 놓일 수 있는 위치(교차점)를 알려준다.

플레이어의 기지base와 이것을 지키는 것이 게임의 목표임을 간단히 알려준다.(살짝만 말하고 넘어간다.)

### 말 설명

미니게임의 맵에 초기상태로 말을 배치한다.

말을 선택하는 법 알려준다.

선택법:

- 1.번호선택:말 위에 뜨는 번호에 맞는 키를 누르는 방법(엔터를 누르지 않아도 자동 선택됨)
  - 2.방향키이용:방향키를 눌러 선택 가능한 놈을 고르고 엔터키를 누르는 방법
- \*2의 경우 배경색을 바꿔서 키보드를 이용한 선택중인 것을 알려주고, 선택이 완료되면  
\*(1의 경우 키보드가 눌러지면, 2의 경우 엔터키가 눌러지면)또 배경색을 바꾼다.

선택이 되면 말이 이동할 수 있는 교차점을 표시해준다(대충 문자를 깜빡깜빡 거리게 해준다거나...)

말이 선택된 후에 : 말을 움직이는 법을 설명한다.

이동법:

#### 1.정방향 번호 선택법

1 2 3

4 @ 6

7 8 9

이 때 갈 수 있는 곳(교차점)에만 번호를 표시하고, 갈 수 있는 곳을 누르면 이동한다.

이동하는 것을 보여준다.

#비정방향 번호 선택법 : 커스텀 게임 등에서 정방향 개념이 없는 맵(곡선 등...)에서 사용됨.

교차점에 왼쪽, 위쪽 우선하여 작은 숫자를 할당시키고 그걸 누르면 이동하는 방식.

#### 2.방향키 선택법

방향키로 현재 갈 수 있는 교차점을 선택 한 다음 엔터키를 누르면 이동한다.

말이 이동하면 턴이 바뀐다. 다른 유저의 말을 이용하여 선택, 이동이 가능하다.

이 루틴을 반복하다가

말이 적 말을 잡는 순간 (ex 청 말이 홍 말을 잡았다)

잡힌 쪽의 라이프-말 수-는 빠지고 ( 홍 의 라이프 표시 칸의 말 하나 빠기)

잡은 쪽의 따먹은 말 수 칸에는 말 하나가 채워진다.

//이제 이 다음은 그냥 보여주기 식이다. 위쪽까지의 핵심 게임 룰은 체험하기였음.

//보여주기는 스페이스 바를 누르면 넘어가기로 하자.

이제 승리 방법을 보여준다.

1.설명(로그)칸에 “한쪽 말이 모두 죽는 경우Wn 남은쪽이 승리” 라고 출력하면서

맵에 예시를 들어 보여준다.

2.한쪽이 포기할 경우 설명: “당신은 청입니다. GG를 입력하고 엔터를 누르세요” //gg치는법은 체험

설명: “청이 게임을 포기하였습니다! 적의 승리입니다!”

3.한쪽의 기지에 적이 침입 설명: “기지를 점령당하는 경우Wn 점령한 쪽이 승리”

역시 맵에 예시를 들어 보여준다.

마지막으로 제한 조건 설명

자신의 기지에는 들어갈 수 없음....!(사실 이동 체험할 때 바로 알겠지만...)

“skip” OR "skip" 을 입력하고 엔터를 치면 게임을 시작합니다.

## 게임 설정을 할 수 있다.

맵선택 단계에서 1.미니게임, 2.풀게임을 고를 수 있다. 0.뒤로

VS 선택 단계에서 1. man VS man을 고를 수 있다. 0.뒤로

## 게임을 할 수 있다.

UI를 그리고 각 플레이어의 탭을 초기상태로 그려준다.

그래프 데이터에 따라 맵을 그리고 정해진 곳에 말들을 그린다.

로그 : “게임이 준비되었습니다. ”

//모든 렌더링은 커서를 사용해야 한다. 사용자의 조작이 허용될 때 커서는 반드시 현재 플레이어의 입력 텍스트

//표시부에 있어야 한다. 만일 입력텍스트를 표시하는 부분에 커서가 없다면 사용자의 조작은 허용하지 않는다.

로그는 가장 마지막에 쓰여진 글은 흰색, 그 외는 회색으로 표시한다.  
새로운 글은 글의 가장 아랫부분에 출력시킨다.  
일정 길이 이상이 되면 위쪽의 로그는 삭제하고 새 로그를 출력한다.(내부적으로는 모두 저장한다.)

### 적색 차례

#### 턴 진입 시 시각효과

로그: “적색 차례입니다.”

적 탭을 노란 상자로 표시한다.

#### 말 선택

1.번호선택법:선택 가능한 말 위에 숫자를 표시한다.

말 위에 표시된 숫자키를 누르면 바로 말이 ‘선택’ 된다.

선택된 말의 배경색을 달리한다.

2.방향키이용:방향키를 누르기 전엔 변화없음. 방향키를 누르면 최소 번호의 말의 배경색 달라짐.

그 상태에서 엔터키를 누르면 말이 ‘선택’ 됨

선택된 말의 배경색을 달리한다.

#### 말 이동

##### 1.정방향 번호 선택법

1 2 3

4 @ 6

7 8 9

갈 수 있는 곳(교차점)에만 번호를 표시한다.

표시된 번호의 숫자 키를 누르면 이동한다.

이동하는 애니메이션을 보여준다.

#비정방향 번호 선택법 : 커스텀 게임 등에서 정방향 개념이 없는 맵(곡선 등...)에서 사용됨.  
교차점에 왼쪽, 위쪽 우선하여 작은 숫자를 할당시키고 그걸 누르면 이동하는 방식.

##### 2.방향키 선택법

방향키를 누르면 이동 가능한 교차점 중 최소 숫자의 배경색이 바뀐다.

이 상태에서 엔터키를 누르면 이동한다.

이동하는 애니메이션을 보여준다.

로그: “적색 n번 말이 x1에서 x2로 이동했습니다.”

#### \*청(적팀)의 말을 뺐다.

청의 라이프에서 ● 하나를 빼고 공백 ‘ ’ 칸으로 바꾼다. 가장 오른쪽부터 바꾼다.

적색의 판 돌에서 공백 ‘ ’ 하나를 ●로 바꾼다. 가장 오른쪽부터 바꾼다.

로그: “적색 n번 말이 청색 m번 말을 뺏습니다!”

#### \*승리조건—청의 모든 말을 뺐다.

화면 중간에 적색의 승리를 축하하는 거대한 문구를 출력한다. 소리도 넣어줄까.

이 상태에서 아무키나 누르면 <게임오버 메뉴>로 간다.

#### \*승리조건—청의 기지에 말이 도달했다.

화면 중간에 적색의 승리를 축하하는 거대한 문구를 출력한다. 소리도 넣어줄까.

이 상태에서 아무키나 누르면 <게임오버 메뉴>로 간다.

청색의 차례로 넘어간다.

\*특정 문구 입력 시에 일어날 수 있는 사항들.

"GG" 혹은 "gg" 입력

\*승리조건:화면 중간에 **청색**의 승리를 축하하는 거대한 문구를 출력한다. 소리도 넣어줄까.  
이 상태에서 아무키나 누르면 <메뉴>로 간다.

"MENU" 혹은 "menu" 입력

<메뉴>로 간다.

## 청 차례

적색의 차례와 다를 게 없다. 다른 건 오브젝트만 다르다.

<메뉴>: 항목별 숫자키를 누르면 바로 진행되고, 방향키를 누르면 먼저 **선택**하고 엔터키를 누른다.

- 1.재시작
- 2.게임모드(맵)선택
- 3.VS선택 모드
- 4.<게임 종료>
- 0.뒤로-이 때는 여기로 왔던 때로 돌아간다.

<게임종료>

아무키나 누르면 종료된다. 뒤로 갈 수 없다.

껍데기부터 만들어볼까? ‘게임’ 부분 말고 그 외 메뉴 선택이나 그런 거. 디자인적인 부분.

순서?

## 0.<TITLE>

콘솔창 크기를 조절한다.(전처리기를 이용하면 상수 텍스트도 바꿀 수 있다)

적당한 위치에

////////////////////////////////////

KUR's GONU

press any key

////////////////////////////////////

출력

사용자로부터 “키보드의 입력”을 기다린다. (kbhit함수를 쓰면 될 것이다)

\*커서는 오른쪽 하단에 위치시킨다(커서위치이동-대기함수:입력칸이나 오른쪽 아래 등.)

만일 입력이 된다면 1.로 넘어간다.

## 1. <MAIN MENU>


적당한 좌표로


////////////////////////////////////

- 1.게임하기 Play
- 2.만든 놈? Credit?
- 3.옵션 Option

원하는 항목의 숫자를 입력하고 엔터키를 누르시오  
혹은 방향키를 눌러 항목을 선택하고 엔터키로 결정하시오  
////////////////////////////////////  
출력

\*커서는 방향키... 텍스트의 아래쪽에 위치시킨다(커서위치이동-대기함수)

사용자로부터 “키보드의 입력”을 기다린다. 입력 함수(예외처리):원하는 자료형과 범위 등 설정가능.  
입력이

1이면 화면 전부 지우고system("cls") 3.<게임설명>으로 화면전환함수:함수 포인터 배열, 상수키워드 이용  
2이면 화면 전부 지우고system("cls") 2.<CREDIT>  
3이면 화면 전부 지우고system("cls") 8.<OPTION>

## 2. <CREDIT>

////////////////////////////////////  
(니 쓰고 싶은거 쓰세요)

(하단에)

0.뒤로

////////////////////////////////////  
출력

\*0.뒤로 아래쪽에 커서 위치시키기(대기함수)

사용자로부터 “키보드의 입력”을 기다린다. (커서위치이동-입력함수)  
0이면 1.<MAIN MENU>로 간다.( 화면전환함수(MAIN\_MENU) )

## 8.<OPTION>

물라 이거 아직 정할 단계 아녀..

## 3.<게임설명>

이건 게임 실제 플레이와 매우 관련이 깊어서 지금 정할 수가 없다

## 4.<MAP\_SELECT>

그냥 지금 화면 전환 함수, 예외처리된 입력함수, 대기함수 등을 구현해보라. 그리고 내용이 없는 껍데기만을 먼저 만들어보라. 시제품의 프로토타입을 만드는 것이다.

그냥 지금 화면 전환 함수, 예외처리된 입력함수, 대기함수 등을 구현해보라.

**inputReady(커서 위치)** : 사용자 입력 전에 커서위치를 옮기고 받은 문자열로 어떤 함수를 쓸지 결정한다.

**strInput(체크할 문자열)** : 문자열을 받아 “체크할 문자열” 과 같은지 확인하는 함수

**selectInput(min, max)** : min ~ max 까지의 정수를 입력받는 함수이며 그 외는 예외처리한다

**printException(예외에 따라 할당된 번호)** : 예외를 출력하는 함수. 입력 장소에 색이 눈에 띄게 출력해야 한다.

실제로는 흰 배경에 검은 글씨나 빨간 글씨가 낫겠다.

<본 게임>과 <게임 설명> 에서는 로그 창에도 예외를... 띄우지 말까? 그게 낫겠다.

**switchScene(<장면 상수>)** : 지금 장면을 모두 삭제하고 다른 장면으로 바꾸는 함수. <장면 상수>는 <장면>을 그리는 함수를 바꾼다. 함수 포인터의 배열 인덱스들임.

**optInt(min, max)** : 장면을 전환switchScene()한 다음 메뉴를 결정해야할 때 쓴다.(selectInput을 사용)

**Boolean** 혹은 **Bool** : 부울 대수 타입 만들기.

#로그 데이터를 파일에 출력시키는 방법으로 저장하는 게 제일 좋을텐데. 알아볼까. 특히 미로게임 만들 때 로그데이터 저장은 아주 중요함.

버퍼를 쓰는 입력 함수와 그렇지 않은 입력 함수 두가지가 존재한다... 버퍼 안 쓰는 입력함수는 방향키와 엔터만 받으면 된다.

**clearLine\_readBuffer**

입력함수의 스펙?

	리턴값	예외	제대로
(버퍼 0 )정해진 일정 범위의 정수를 입력 -> <b>selectInput(min, max)</b>		0	포인터 값?
(버퍼 0 )정해진 특정 단어 문자열 입력 -> <b>strInput(문자열의 배열)</b>		0	포인터 값?
(버퍼 X ) ‘엔터’ , ‘방향키’ 만 입력. -> <b>noBufferInput()</b>		0	포인터 값?

잠깐... 3개는 다른 함수지만 서로 교환되어야 하는데? **Input()** 함수에서 현재 입력하는 키 값을 인식해서 3개 중 무엇을 호출시킬지 알아야 될 듯? 물론 인식한 키 값을 그대로 3가지 함수로 전달시켜야함.

**inputReady()** -> **selectInput(min, max)**      0 ~ n 고르기  
                   -> **strInput(문자열의 배열)**      “gg” “SKIP” 등  
                   -> **noBufferInput()**

일단 문자열로 받음      @

기능키인가?(버퍼 X)  
 그게 숫자면 ...  
 그게 문자열이면...

본게임 명세

게임시작 -> 초기화 -> 1p -> 2p -> 1p -> 2p ->..... -> 게임 끝 메뉴

## 1단계:대단위 루틴

### 0. 초기화

1. 턴 진입 시각효과
2. 말 선택

### 3. 말 이동

#### 3.5 상태 업데이트

### 4. 승리조건 확인

### 5. 턴 넘기기 -> 1.턴 진입 시각효과

## 2단계:더 자세한 명세

### 0. 인게임 데이터 초기화

필요한 변수들을 모두 초기화시킨다.

### 1. UI를 그린다.

로그를 쓰는 창을 그린다.

각 플레이어의 창을 그린다.

그래프 데이터에 따라 맵을 그린다.

정해진 곳에 각 플레이어들의 말들을 그린다.

로그 : “게임이 준비되었습니다. ”

//모든 렌더링은 커서를 사용해야 한다. 사용자의 조작이 허용될 때 커서는 반드시 현재 플레이어의 입력 텍스트

//표시부에 있어야 한다. 만일 입력텍스트를 표시하는 부분에 커서가 없다면 사용자의 조작은 허용하지 않는다.

로그는 가장 마지막에 쓰여진 글은 흰색, 그 외는 회색으로 표시한다.

새로운 글은 글의 가장 아랫부분에 출력시킨다.

일정 길이 이상이 되면 위쪽의 로그는 삭제하고 새 로그를 출력한다.(내부적으로는 모두 저장한다.)

### 2. 적색(1p) 차례

턴 진입 시 시각효과를 보여준다.

말을 선택할 수 있게 한다.

유저가 말을 선택할 때까지 입력을 대기한다.

    항복명령어를 입력했다

        1p가 졌음을 알린다.

        게임 끝 메뉴로 화면 전환한다.

말이 선택되면 명령을 입력할 수 있게 한다.

    항복 명령어를 입력했다.

        1p가 졌음을 알린다.

        게임 끝 메뉴로 화면 전환한다.

이동 명령을 입력했다.

    명령에 맞게 말이 이동하는 애니메이션을 보여준다.

    로그 : “적색 n번 말이 x1에서 x2로 이동했습니다.”

        적의 말을 뺐다.

        2p의 life를 줄이고 2p 플레이어의 창을 업데이트한다.

        1p의 판 돌 공간을 업데이트한다.

        로그 : “적색 n번 말이 청색 m번 말을 뺐습니다!”

승리조건-2p의 라이프가 0이다

        1p가 승리했음을 알린다.

        게임 끝 메뉴로 화면 전환한다.

승리조건—청의 기지에 말이 도달했다.

        화면 중간에 1p적색의 승리를 축하하는 거대한 문구를 출력한다. 소리도 넣어줄까.



이 상태에서 아무키나 누르면 <게임오버 메뉴>로 간다.  
청색의 차례로 넘어간다.

## 3단계 : 더 자세한 명세

현재 게임 화면으로 전환이 되었다.

### 0. 인게임 데이터 초기화

필요한 변수들을 모두 초기화시킨다. 아직 변수가 무엇인지 정할 수 없음

### 1. UI를 그린다.

오른쪽 상단에 로그 창을 사각형으로 그린다.

로그 창 하단에 1p의 창을 그린다.

1p창 하단에 2p의 창을 그린다.

그래프 데이터에 따라 맵을 그린다.

각 정점의 연결리스트에 접근한다.

연결리스트를 순회하며 간선을 그린다.

간선을 다 그리면 다시 한 번 정점을 순회하며 데이터에 따라 정점을 그린다.

이 부분에서 사용자의 입력에 따라 말의 배치가 달라지는 확장 게임 모드가 가능해야 한다.

그것을 위해선 각각의 단계가 서로 종속되지 않고 모듈화되어 있어 조립할 수 있는 방식이 되어야 한다.

정해진 곳에 각 플레이어들의 말들을 그린다.

말의 정보를 저장한 리스트에 접근하여 말들을 그린다.

로그 : “게임이 준비되었습니다. ”

//모든 렌더링은 커서를 사용해야 한다. 사용자의 조작이 허용될 때 커서는 반드시 현재 플레이어의 입력 텍스트

//표시부에 있어야 한다. 만일 입력텍스트를 표시하는 부분에 커서가 없다면 사용자의 조작은 허용하지 않는다.

로그는 가장 마지막에 쓰여진 글은 흰색, 그 외는 회색으로 표시한다.

새로운 글은 글의 가장 아랫부분에 출력시킨다.

일정 길이 이상이 되면 위쪽의 로그는 삭제하고 새 로그를 출력한다.(내부적으로는 모두 저장한다.)

### 2. 적색(1p) 차례

턴 진입 시 시각효과를 보여준다.

2p의 창은 작고 눈에 덜 띄는 문자(예 : ‘.’)로 그리고

1p의 창은 눈에 잘 띄는 문자(예 : ‘#’)로 그린다. 색깔을 달리할 수도 있다.

말을 선택할 수 있게 한다.

유저가 말을 선택할 때까지 입력을 대기한다.

선택할 수 있는 말 위에 숫자를 띄운다. 약간 복잡하므로 그림을 그린다.

화이트보드, 사진 : 입력 루틴에 관한 문서(그림)를 참조한다. 의사 코드도 참조한다.

### 3. 말 이동 :

명령에 맞게 말이 이동하는 애니메이션을 보여준다.

이동하는 말의 정점 좌표 데이터에서 이동하는 곳의 정점 데이터를 이동함수에 대입

로그 : “적색 n번 말이 x1에서 x2로 이동했습니다.”

적의 말을 뺏다면

2p의 life를 줄이고 2p 플레이어의 창을 업데이트한다.

1p의 판 돌 공간을 업데이트한다.

로그 : “적색 n번 말이 청색 m번 말을 뺏습니다!”

#### 4. 승리조건 확인 :

승리조건-2p의 라이프가 0이다

화면 중간에 1p적색의 승리를 축하하는 거대한 문구를 출력한다.

게임 끝 메뉴로 화면 전환한다.

승리조건—청의 기지에 말이 도달했다.

화면 중간에 1p적색의 승리를 축하하는 거대한 문구를 출력한다. 소리도 넣어줄까.

이 상태에서 아무키나 누르면 <게임오버 메뉴>로 간다.

#### 5. 2p청색의 차례로 넘어간다.

하고 싶은 것

인공지능 구현을 위한 인터페이스가 제공되어야 한다.

하나의 c파일로 인공지능을 구현할 수 있어야 한다.

말배치 기능이 포함될 수 있어야 한다.

말의 종류가 달라질 수 있다.

다양한 룰을 적용할 수 있는 확장성이 있어야 한다.

그래픽 요소는 달라질 가능성이 높다. 여기에 대한 확장성도 있어야 한다.

mini game이 말배치 없는 게임? / full game은 말 배치 있고 해결사들이 있는 게임?

mini game 설명에서는 기초 규칙만 설명하고 확장성을 위해 다른 모드(예 : full game)에선 게임 시작 전에 설명?

## 입력루틴 의사코드

### 1.turn(P1 or P2) 턴 함수의 구조.

```
while( 1 )
    selectInput = inputForSelect()      //말 선택을 위한 입력!

    if selectInput == 선택된 말 정보
        while( 1 )
            moveInput = inputForMove(selectInput, ...)
            //이동할 곳을 정하는 입력!
            //당연히 선택된 말에 대한 정보도 있어야 한다.

            if moveInput == 말 이동 정보나 gg or GG
                말 이동하는 애니메이션 보여주는 함수
                return 턴이 넘어간다.

            else if
                resetBackColor( 바꿀 좌표, 다시 쓸 문자)
                break                //즉 다시 입력을 받기 위해 바깥쪽 루프를 돌린다.

            else if moveInput == gg or GG
                return 지금 조작중인 플레이어의 패배

        end while

    else if selectInput의 반환값 == gg or GG
        return 지금 조작중인 플레이어의 패배

    else
        resetBackColor( 바꿀 좌표, 다시 쓸 문자)
        eraseFlagDigits

    end else

end while
```

1-0.달라진 배경색을 다시 바꿀 함수가 필요하다. 아니면 바뀌어진 색깔들이 계속 남아 있을 것이다.  
턴이 바뀔 때, 말 재선택, 이동위치 재선택 시 필요한 함수다.

resetBackColor( 바꿀 좌표, 다시 쓸 문자) //색을 바꾸는 건 결국 문자를 새로 쓰는 것이다.  
eraseFlagDigits //이전에 정점이나 말에 표시되었던 숫자들을 지우는 함수

솔직히 애들은 직접 만들어보고 필요할 때 넣어야겠다는 생각이 든다. 그래도 충분하고.  
이런 부분까지 지금 설계할 필요가 없다.

#### 1-1.inputForSelect 함수의 구조

선택할 수 있는 말들에 숫자(선택 가능한 숫자)를 부여하고 말들 위에 숫자를 출력한다.

```
input = stdInput
```

```
if input == 방향키
```

```
    최소 숫자의 말 배경색을 주황색으로.
```

```
while(1)
```

```
    _input = stdInput           //말 선택을 위한 입력!
```

```
    if _input == 방향
```

```
        방향키에 맞는 말 배경색을 주황색으로(기존의 주황색은 다시 검게)
```

```
    else if _input == 엔터
```

```
        break
```

```
    else if _input == 숫자
```

```
        if 입력한 수 < 선택 가능한 숫자
```

```
            break
```

```
    else if _input == gg OR GG
```

```
        return 지금 조작중인 플레이어의 패배
```

```
end while
```

```
선택된 말의 배경색을 노란색으로.
```

```
return 선택된 말의 정보
```

```
else if input == 숫자
```

```
    if 입력한 수 < 선택 가능한 숫자
```

```
        선택된 말의 배경색을 노란색으로.
```

```
        return 선택된 말의 정보
```

```
    else
```

```
        return 0           //null값. 기본적인 예외값의 입력을 의미하므로 루프를 다시 돌게 한다.
```

```
else if input == gg OR GG
```

```
    return 현재의 플레이어가 진다
```

```
else
```

```
    return 0           //null값. 기본적인 예외값의 입력을 의미하므로 루프를 다시 돌게 한다.
```

## 1-2.inputForMove(선택된 말 정보, ...) 함수의 구조

선택된 말의 위치에서 갈 수 있는 정점들에 숫자(선택 가능한 숫자들)를 부여하고 숫자를 정점에 출력한다.

```
input = stdInput
```

```
if input == 방향키
```

```
    최소 숫자의 정점 배경색을 주황색으로.
```

```

while(1)
    _input = stdInput      //이동 위치 선택을 위한 입력!

    if _input == 방향
        방향키에 맞는 정점 배경색을 주황색으로(기존의 주황색은 다시 검게)
    else if _input == 엔터
        break
    else if _input == 숫자
        if 입력한 수 < 선택 가능한 숫자
            break
    else if _input == gg OR GG
        return 지금 조작중인 플레이어의 패배

```

end while

```

선택된 정점의 배경색을 노란색으로.
return 선택된 정점의 정보

```

```

else if input == 숫자
    if 입력한 수 < 선택 가능한 숫자
        선택된 정점의 배경색을 노란색으로.
        return 선택된 정점의 정보
    else
        return 0      //null값. 기본적인 예외값의 입력을 의미하므로 루프를 다시 돌게 한다.

```

```

else if input == gg OR GG
    return 현재의 플레이어가 진다

```

```

else
    return 0      //null값. 기본적인 예외값의 입력을 의미하므로 루프를 다시 돌게 한다.

```

보면 지금 `inputForSelect`와 `inputForMove`가 거의 같다. 숫자를 부여하는 위치(말 / 정점)만 다를 뿐 하는 짓은 똑같아. 하나의 함수로 쓴다음 숫자를 부여하는 위치의 함수를 바꾸면 될 거 같다...

입력 루틴 그 자체는 변하지 않고, 숫자를 부여하는 부분을 변하게 함으로써 색다른 기능을 가진 말들을 구현할 수 있다.

예  
2칸씩 움직이는 말

말을 쌓아서 움직일 수 있게 하려면 어떤 게 있어야 할까??

그럼 지금까지의 설계에서 만들어야하는 **최소단위의 함수**는?

현재 게임 화면으로 전환이 되었다.

```

drawMap(Graph)      //그래프 데이터에 따라 맵을 그린다.
    각 정점의 연결리스트에 접근한다.
    연결리스트를 순회하며 간선을 그린다.

```

간선을 다 그리면 다시 한 번 정점을 순회하며 데이터에 따라 정점을 그린다.

이 부분에서 사용자의 입력에 따라 말의 배치가 달라지는 확장 게임 모드가 가능해야 한다.

그것을 위해선 **각각의 단계가 서로 종속되지 않고 모듈화되어 있어 조립할 수 있는 방식**이 되어야 한다.

`drawPieceInit()`

`logWriter`

//모든 렌더링은 커서를 사용해야 한다. 사용자의 조작이 허용될 때 커서는 반드시 현재 플레이어의 입력 텍스트

//표시부에 있어야 한다. 만일 입력텍스트를 표시하는 부분에 커서가 없다면 사용자의 조작은 허용하지 않는다.

로그는 가장 마지막에 쓰여진 글은 흰색, 그 외는 회색으로 표시한다.

새로운 글은 글의 가장 아랫부분에 출력시킨다.

일정 길이 이상이 되면 위쪽의 로그는 삭제하고 새 로그를 출력한다.(내부적으로는 모두 저장한다.)

`turnChangeDisplay()`

`winBoxDisplay()`

`pieceLineMove`(말 구조체, 출발점, 도착점, 이동 시간)

`resetBackColor`( 바꿀 좌표, 다시 쓸 문자) //색을 바꾸는 건 결국 문자를 새로 쓰는 것이다.

`eraseFlagDigits` //이전에 정점이나 말에 표시되었던 숫자들을 지우는 함수

`flagDigitsAtPieces`

//선택할 수 있는 말들에 숫자(선택 가능한 숫자)를 부여하고 말들 위에 숫자를 출력

현재 플레이어의 말 데이터에 접근해야 한다.

`flagDigitsAtCrossing` //선택된 말의 위치에서 갈 수 있는 정점들에 숫자(선택 가능한 숫자들)를 부여하고

숫자를 정점에 출력한다. 현재 선택된 말의 Vertex에서 linkedList에 접근해야한다.

초기화 -> 기본상태 -> 입력 -> 상태 변경 ->탈출조건? -> 입력 -> 상태변경