플레이봇 퍼즐게임 보고서

**- 깃발을 찾아서 -**

**< 코드 설명 >  
< 인코딩/디코딩 편 >**

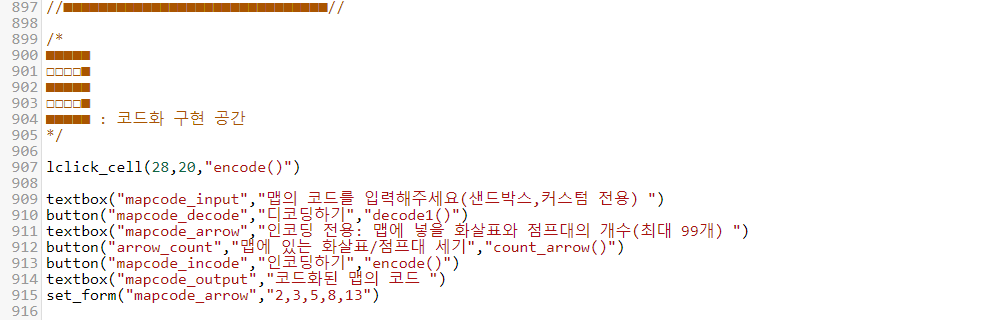
**- 만든 이: 원인식 (MrWon)   
- 만든 날짜: 2020.1.20**

인코딩 원리 설명  
- 기본적으로 W/B 압축 방식을 쓰며 16진법을 씀 (00001110 ->W4B3W1)  
- 그러나 W,B라는 증거를 직접 표시하지 않고 서로 다른 16진법을 써서 해결함  
- W(0)은 1~f의 16진법을 쓰며 B(1)은 g~v에 0~15를 대입하는 16진법을 씀  
(예: 00000000000111000 -> bi3)  
- 그 외에도 게임 특성상 많이 나오는 “10”은 A~P에 0~15를 대입하는 16진법을 쓰며 벽을 제외한 오브젝트는 S~Z에 그냥 대입해서 씀 (예: 1010102310 -> CSTA)

- 그리고 맵코드 마지막에는 0102030405-0607-0809이라는 아이템 코드를 집어넣음 (“-“는 편의상 집어넣음)  
- 01~05 는 화살표/점프대 개수를 차례대로 나열한 것으로 a~y, A~Y (각각 25개)를 이용한 50진법으로 숫자를 표시함. 개수가 50이 넘으면 50을 빼고 50진법화시킴  
- 06는 0102의 개수가 50이 넘는지 안 넘는지 여부를 2진법으로 나타낸 문자임  
(예: 01= 15, 02= 52, 06= “01>50”+”02>50” = ”0”+”1” = 01 = 1)  
- 07은 03 ,04 ,05의 개수가 50이 넘는지 안 넘는지 여부를 2진법으로 나타난 문자임  
(예: 03= 2, 04= 62, 05­= 50, 07= “03>50”+”04>50”+”05>50” = “0”+”1”+”0” = 010 = 2)  
- 08은 시작 방향을 나타냄  
- 09는 벽 색깔을 나타냄

- 이렇게 해서 맵 코드가 만들어짐.   
- p.s. 중간에 대쉬(-)를 끼워넣는 이유는 보기 좋으라고 놓음

디코딩 원리 설명  
- 서로 다른 진법을 사용하니 그 진법에 맞는 0,1,01,2~9 데이터를 맵 배열에 집어넣음  
- 아이템 코드도 분해해서 데이터를 입력함

  
- 907 : 우측 상단의 문서 아이콘 클릭 시 인코딩하기  
- 909 : 디코딩할 코드 입력  
- 910 : 디코딩 버튼  
- 911 : 인코딩할 맵의 화살표/점프대 개수 입력  
- 912 : 맵에 있는 화살표/점프대 세서 911번째 줄 버튼에 입력  
- 913 : 인코딩 버튼  
- 914 : 코드화된 맵 코드 출력  
- 915 : 화살표/점프대 입력하는 예시 출력  
  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- encode() : map[x][y]에 있는 데이터 및 여러 데이터를 인코딩(코드로 만듬)하는 함수

- 916~923 : 인코딩에 쓸 여러가지 변수 초기화  
- 925~930 : 맵 전체 인코딩  
- 931 : 맵 인코딩 후 10(g0)을 전부 “.”로 바꿈  
- 932 : “.”을 세서 전부 A~P 16진법으로 바꿈  
- 933 : 아이템 코드를 인코딩 다 된 맵코드 뒤에 붙임  
- 934 : 벽 색깔도 맵코드 뒤에 붙임  
- 935 : 맵 코드에 “-“(대쉬) 추가 (안해도 됨)  
- 937 : 맵코드 출력  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- encode\_segm(x) : x를 받으면 그 x를 인코딩하는 일꾼 함수

- 941 : 현재 상태(0,1,…)가 받은 x랑 똑같으면 num의 값을 올림  
- 942~949 : 현재 상태가 받은 x랑 안 똑같으면 모아놓은 num의 값을 현재 상태에 맞게 16진법화시킨 맵코드 뒤에 붙임  
- 950~951 : 그 후 현재 상태를 x로 바꾸고 num도 초기화함  
p.s. 만약 while문을 써서 num을 올렸다면 좀더 보기 좋은 코드가 탄생했을 것….

스크린샷이(가) 표시된 사진

자동 생성된 설명  
- end\_of\_encode() : encode\_segm(x)는 구조상 마지막에 else쪽에 있는 구문을 한번 더 덧붙여줘야 되는 함수라 마지막에 else쪽에 있는 구문을 덧붙이기 위한 용도의 함수  
- 956~962 : encode\_segm(x)랑 동일

p.s. encode\_segm(x,y)의 941부분의 if (now\_state==x)를 if (now\_state==x && now\_state<2)를 추가할 것. 만약 특수 오브젝트가 2개 겹쳐있으면 인식을 못함

스크린샷이(가) 표시된 사진

자동 생성된 설명  
- dot\_to\_Aex() : “.”으로 바꾼 “g0”의 개수를 세서 A~P 16진법으로 바꾸는 함수

- 966 : 맵코드의 길이를 미리 저장  
- 967~972 : 맵코드의 i번째 글자가 점이면 점이 끝날 때까지 세다가 점이 끝나면 이때까지 센 점들을 16진법으로 만들어라는 신호를 replace\_start로 보냄  
- 974~976 : replace\_start가 1이면 점이 시작되기 전까지의 코드를 temp\_mapcode에, 점이 끝난 후의 코드를 mapcode에 저장함  
- 977~978 : i를 0으로 바꾸고 length를 mapcode의 길이로 바꿈 (다시 시작)  
- 980~983 : dot\_num를 1~f 16진법으로 바꾼 뒤 A~P 16진법으로 바꾼 후 temp\_mapcode에 붙여넣음  
- 989~1001 : dot\_to\_Aex()의 알고리즘이 encode\_segm(x)의 알고리즘을 차용한 거라 마지막에 한번 더 else쪽 구문을 실행시킴  
- 1002~1003 : 마지막에 temp\_mapcode 뒤에 mapcode를 붙이고 temp\_mapcode를 mapcode에 복사함  
스크린샷이(가) 표시된 사진

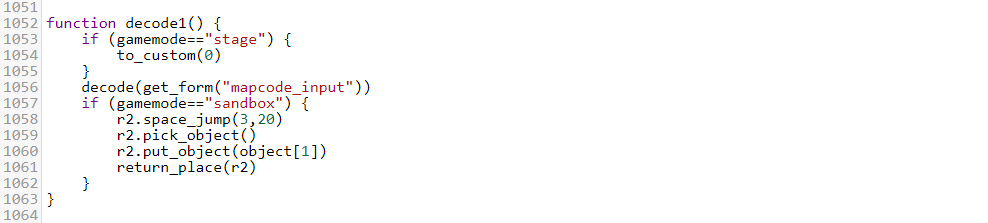
자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명  
- iteminfo\_encode() : iteminfo에 있는 정보와 시작 방향을 아이템코드에 담는 함수

- 1007 : 화살표/점프대 개수 입력창에 있는 숫자를 “,”를 기준으로 나누어 iteminfo\_input[]이라는 배열에 담음  
- 1009~1015 : iteminfo\_input[i]에 있는 숫자를 mod() 자체함수로 숫자가 50n+a이라고 칠 때 a를 꺼낸 뒤 그 a를 a~Y 50진법으로 바꾼 뒤 iteminfo\_code에 붙임. (1쪽 참고)  
- 1016~1017 : iteminfo\_input[0~4]가 50이 넘는지 안 넘는지 여부를 2진법으로 기록한 숫자 2개를 iteminfo\_code에 붙임. (1쪽 참고)  
- 1018 : 시작 방향을 iteminfo\_code에 붙임  
- 1020 : iteminfo\_code를 반환

스크린샷이(가) 표시된 사진

자동 생성된 설명  
- decode(str) : 받은 코드를 해독해 맵과 여러가지 변수에 옮기는 함수

- 1031: 필요없는 코드  
- 1032~1035 : 디코딩에 필요한 갖가지 변수 초기화  
- 1034~1038 : 코드에 있는 “-“(대쉬) 제거  
- 1040~1041 : 맵코드하고 아이템코드 분리  
- 1043~1046 : 맵코드를 해독해서 맵 배열에 옮김  
- 1047 : 아이템 코드도 해독해서 아이템 관련 변수/배열에 옮김  
- 1048~1049 : 맵 전체 매핑하고 아이템 개수도 표시함  
  
  
- decode1() : 디코딩 버튼 전용 디코딩 함수, 위 함수와 차이점은 decode()은 다용도 디코딩 함수라는 것  
- 1053~1055 : 게임모드가 스테이지면 게임모드를 커스텀으로 바꿈  
- 1056 : 맵코드 입력창에 있는 맵코드를 decode()를 써서 디코딩함  
- 1057~1062 : 게임모드가 샌드박스면 샌드박스 오브젝트 선택창에 있는 벽의 색깔을 바꿈  
  
p.s. 게임모드가 샌드박스일 시 오브젝트 선택창에 있는 시작방향 화살표를 바꾸는 코드도 후에 추가함  
  
  
  
  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- decode\_segm(char) 받은 char을 해독해 맵 배열에 옮기는 함수

- 1066 : 받은 char이 현재 상태에 속하면 temp\_decode에 char을 추가  
- 1067 : **<받은 char이 현재 상태에 속하지 않을 시>**- 1068~1073 : 현재 상태가 0이라면 포인터 ix, iy를 따라 빈 공간(0)을 temp\_decode를 해독한 수만큼 채워넣음  
- 1074~1080 : 현재 상태가 1이라면 포인터 ix, iy를 따라 벽(1)을 temp\_decode를 해독한 수만큼 채워넣음  
- 1081~1090 : 현재 상태가 2이라면 포인터 ix, iy를 따라 벽+빈공간(01)을 temp\_decode를 해독한 수만큼 입력함  
- 1091~1098 : 현재 상태가 3이라면 포인터 ix, iy를 따라 오브젝트(2~9)를 입력함  
- 1099~1102 : 현재 상태를 받은 char이 속하는 상태로 바꾼 뒤 temp\_decode를 초기화하고 받은 char을 temp\_decode에 입력함  
  


- decode\_identifying(num) : num이 어느 상태(디코딩용)에 속하는 지 알아내는 함수  
- 1106~1010 : 여기서 num이란 아스키 코드 숫자를 뜻함, 1~f는 0, g~v는 1, A~P는 2, S~Z는 3  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- degexnate(str) : g~v 16진법을 일반적인 16진법 함수로 되돌리는 함수

- 1114~1115 : temp\_arr[]에 인수로 받은 str의 각 알파벳을 입력하고 str을 초기화함  
- 1116~1118 : temp\_arr[]에 있는 str의 각 알파벳을 asc()를 이용해 일반적인 16진법으로 바꾸고 str에 추가함  
- 1119 : str을 반환  
  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- degexnate(str) : A~P 16진법을 일반적인 16진법 함수로 되돌리는 함수

- 1123~1124 : temp\_arr[]에 인수로 받은 str의 각 알파벳을 입력하고 str을 초기화함  
- 1125~1126 : temp\_arr[]에 있는 str의 각 알파벳을 asc()를 이용해 일반적인 16진법으로 바꾸고 str에 추가함  
- 1128 : str을 반환  
  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- end\_of\_decode() : 디코딩의 끝을 담당하는 함수  
- 1132~1163 : 이 함수 역시 encode이랑 같은 알고리즘을 사용해서 end\_of\_encode()랑 같은 이유로 추가한 함수  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- iteminfo\_decdoe(str) : 맵코드랑 따로 떨어져나온 아이템 코드를 디코딩하는 함수

- 1166 : temp\_arr에 인수로 받은 str의 각 숫자/알파벳을 입력함  
- 1167~1168 : temp\_arr[5~6]에 있는 숫자(문자열 상태)를 숫자(정수 상태)로 바꾼 뒤 2진법 문자열로 바꿈  
- 1169~1174 : temp\_arr[0~4]에 있는 50진법 알파벳을 10진법 숫자로 바꾼 뒤 temp\_arr[5~6]에 있는 50 초과 여부 정보에 따라 50을 더하고 iteminfo[x]에 입력함  
- 1175~1176 : 아이템코드 끝에 있는 시작방향과 벽 색깔을 마지막으로 start\_pd와 wall\_color에 입력함  
  
  
- dealphexnate(char) : 50진법 알파벳 char을 일반적인 10진법 숫자로 바꾸는 함수  
- 1180~1185 : 소문자면 asc(char)-97로 0~24로 변환, 대문자면 asc(char)-40으로 25~49로 변환

스크린샷이(가) 표시된 사진

자동 생성된 설명  
- count\_arrow() : 맵에 있는 화살표/점프대 개수를 세는 함수

- 1188 : 화살표/점프대를 셀 때 쓸 item\_count를 초기화시킴  
- 1190~1201 : 맵을 돌아다니며 화살표/점프대 개수를 item\_count[0~4]에 집어넣음  
- 1205~1208 : item\_count[0~4]에 있는 숫자를 item\_count[5] 문자열에 반점(,)과 함꼐 집어넣음  
- 1211~1212 : 화살표/점프대 입력창에 item\_count[5]를 입력 후 r2를 제자리로 돌려보냄  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- to\_hexstr(x) : 10진법 숫자 x를 16진법 숫자(문자열)로 바꾸는 함수  
- 1216 : x.toString(16)을 통해 16진법 숫자로 바꾼 뒤 String을 통해 문자열로 바꿈  
  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- str\_hexnate(str) : 16진법 숫자(문자열)을 일반적인 10진법으로 바꾸는 함수  
- 1220~1221 : 인수로 받은 숫자(문자열)이 10보다 크면 그대로 정수화 해서 보내고 그게 아니면 아스키코드 해독함수, asc()를 이용해 숫자로 변환   
p.s. 숫자(문자열)도 일반 숫자랑 비교가능  
  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- to\_gexstr(str,len,num) : 일반적인 16진법 숫자(문자열)을 g~v 16진법으로 바꿔서 돌려보내는 함수  
- 1225 : 재귀함수여서 마지막을 만들어줌  
- 1226 : len을 1만큼 감소시킴  
- 1227 : str은 받은 문자열을 뜻하고 len은 그 문자열의 길이, num은 몇번째 문자를 해독할지를 뜻하는 인수. str의 num번째 글자를 10진법으로 만든 다음 아스키코드 함수, chr()을 이용해 g~v 16진법으로 바꾼 뒤, 뒤에 n+1번째 글자를 해독하라는 재귀함수 호출  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- to\_Aexstr(str,len,num) : 일반적인 16진법 숫자(문자열)을 g~v 16진법으로 바꿔서 돌려보내는 함수  
- 1231~1234 : 전체적인 구성은 to\_gexstr과 같음  
  
스크린샷이(가) 표시된 사진

자동 생성된 설명  
- to\_specialstr(x) : 오브젝트 번호를 그에 해당하는 문자(S~Z)로 바꾸는 함수  
- 1237 : 아스키코드 문자 반환 함수, chr()을 이용해 문자로 반환함

p.s. to\_hexstr 종류의 함수는 split()을 썻었다면 약간 길어지더라도 깔끔한 코드가 나왔을 것