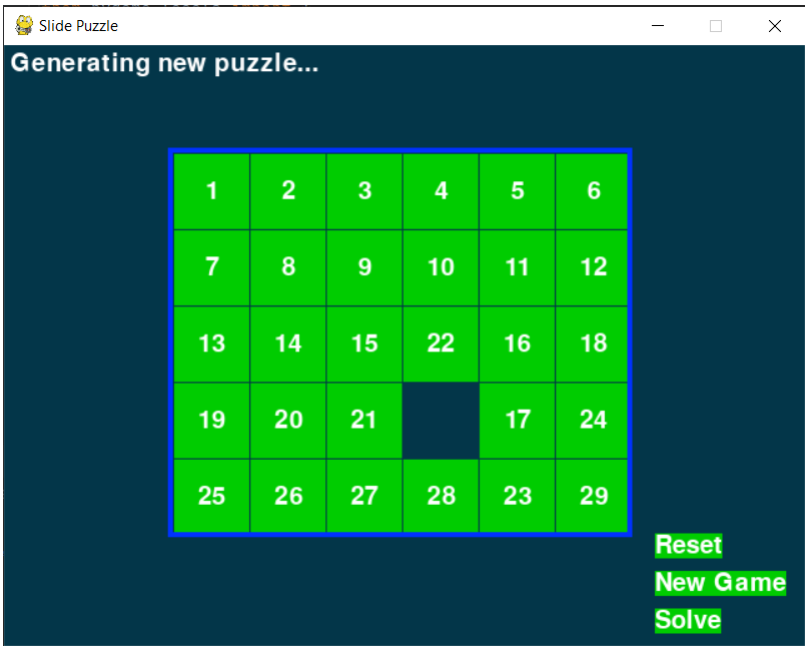
SlidePuzzle лабараториска вежба

1. *Бројот на полиња во редиците и колоните да биде различен.*

   Бројот на полиња го менуваме едноставно со менување на вредностите на BOARDWIDTH и BOARDHEIGHT дополнително со цел приказот да остане валиден односно во прозорецот да ги собере сите ново додадени полиња ја намалуваме големината на полињата односно вредноста на TILESIZE.

   ```
   BOARDWIDTH = 6  # number of columns in the board
   BOARDHEIGHT = 5 # number of rows in the board
   TILESIZE = 60
   ```



2. *Додадете ново копче "HELP". Ако играчот кликне на копчето, на екранот треба да му се прикаже порака со можна насока за придвижување во следниот чекор. Покрај тоа, сите соседни полиња на празното треба да се обоени во црвена боја.*

   За да додадеме ново копче дефинираме глобални променливи HELP_SURF и HELP_RECT кој ке ја чуваат површината на копчето и ги иницијализираме

   ```
   global HELP_SURF, HELP_RECT
   HELP_SURF, HELP_RECT = makeText('Help', TEXTCOLOR, TILECOLOR,
   WINDOWWIDTH - 120, WINDOWHEIGHT - 120)
   ```

   Дополнително во main функцијата креираме локална променлива која ке чува листа од означени (маркирани) полиња односно полиња кои сакаме да ги обоиме во црвена боја.

```
markedTiles = []
```

Потоа во main loop додаваме код кој што ќе проверува дали е кликнато копчето HELP и доколку е кликнато во променливата се зачувуваат полињата коишто може да се движат во следниот чекор.

```
elif HELP_RECT.collidepoint(event.pos):
        markedTiles = getHelpTiles(mainBoard)
```

полињата коишто може да се движат ги земаме со функција getHelpTiles која што проверува за секоја насока дели е валидно движењето доколку е координатите на тоа поле ги додава во листа која што потоа се враќа како резултат

```
def getHelpTiles(board):
        blankx, blanky = getBlankPosition(board)
        validMoves = []
        if isValidMove(board, UP):
                validMoves.append((blankx, blanky + 1))
        if isValidMove(board, DOWN):
                validMoves.append((blankx, blanky - 1))
        if isValidMove(board, LEFT):
                validMoves.append((blankx + 1, blanky))
        if isValidMove(board, RIGHT):
                validMoves.append((blankx - 1, blanky))
        return validMoves
```

За означените полиња да се исцртаат во црвена боја на екран функцијата drawBoard што ја црта таблата ја менуваме тако што ќе прима уште еден аргумент за листа од координати на полиња кои што се означени (маркирани) кој што има предефинирана вредност празна листа. И дополнително додаваме код кој што откако ќе се исцртаат сите полиња на таблата ги прецртува тие полиња кои што се означени

```
def drawBoard(board, message, markedTiles=[]):
        ...
        for tilex, tiley in markedTiles:
                drawTile(tilex, tiley, board[tilex][tiley], color=MARKED_TILE_COLOR)
        ...
        DISPLAYSURF.blit(HELP_SURF, HELP_RECT)
        …
```

Исто така за функцијата drawTile коа исцртува едно поле на екран правиме промена така што додаваме уште еден аргумент кој ја означува бојата на полето што се исцртува со предефинирана вредност на TILECOLOR

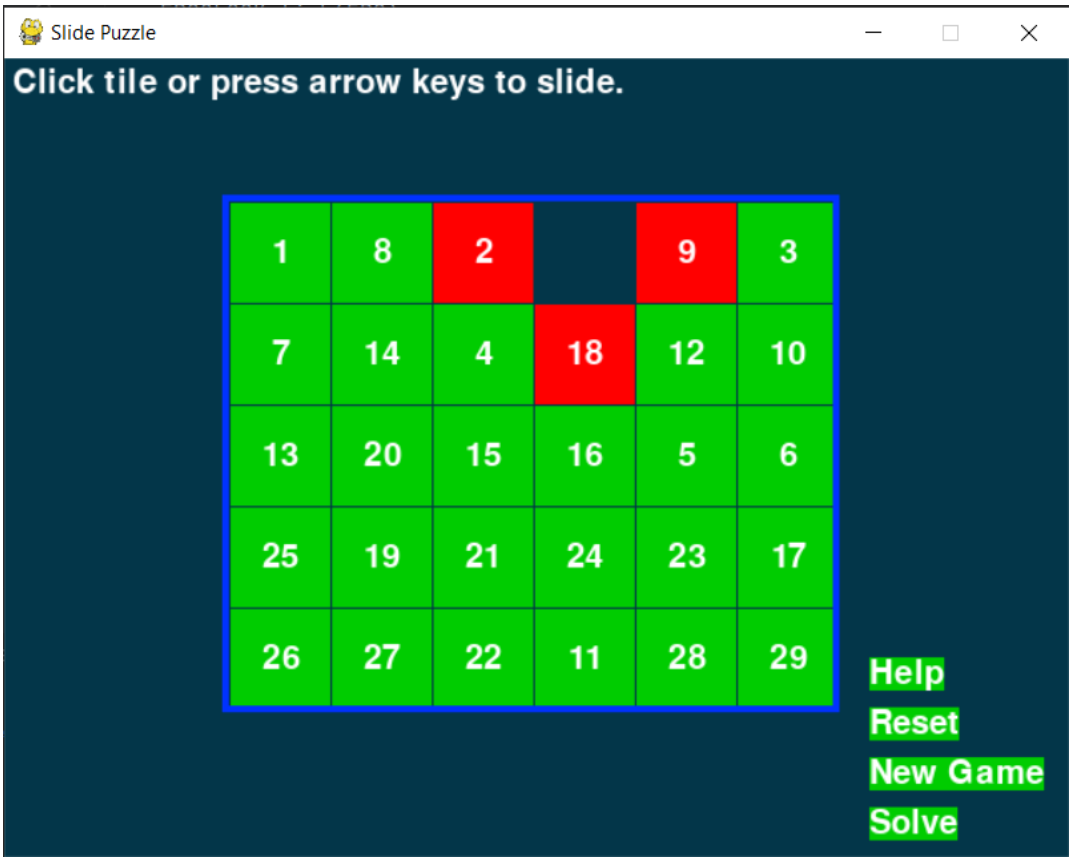
```
def drawTile(tilex, tiley, number, adjx=0, adjy=0, color=TILECOLOR):

        …
```

```
pygame.draw.rect(DISPLAYSURF, color, (left + adjx, top + adjy, TILESIZE, TILESIZE))
...
```

Дополнително во main loop при повик на функцијата drawboard додаваме параметар за означените полиња.

```
drawBoard(mainBoard, msg, markedTiles)
```

Исто така при движње или при клик на некое од останатите копчиња (reset, solve, new game) листата со означени полиња се поставува на празна листа.



3. *Додадете бројач на поместувањата што ги прави играчот. Ако корисникот успешно ја заврши играта, информирајте го за бројот на чекори што му бил потребен да ја заврши играта и бројот на чекори што би биле потребни за решавање на играта од страна на компјутерот (имајќи ја предвид логиката што за таа цел се користи при кликнување на копчето "Solve").*

   За да го постигнеме ова во main loop во делот каде се проверува дали играта е успешно завршена наместо да ја исмишеме пораката “Solved!” испишуваме порака

во која кашуваме колку движења направил играчот а колку му биле потребни на компјутерот.
Во променливата allMoves ги чуваме сите движења на играчот па едноставо со функцита len може да го дознаме бројот на движења додека пак во променливата solutionSeq ги чуваме движењата потребни за генерирање на загатката односно движењата потребни компјутерот да ја реши загатката па така со функцијата len моше да го дознаме број на движења за компјутерот да ја реши загатката

```
if mainBoard == SOLVEDBOARD:
        msg = 'Solved in {} moves the computer could solved it in {}' \
        'moves'.format(len(allMoves),len(solutionSeq))
```