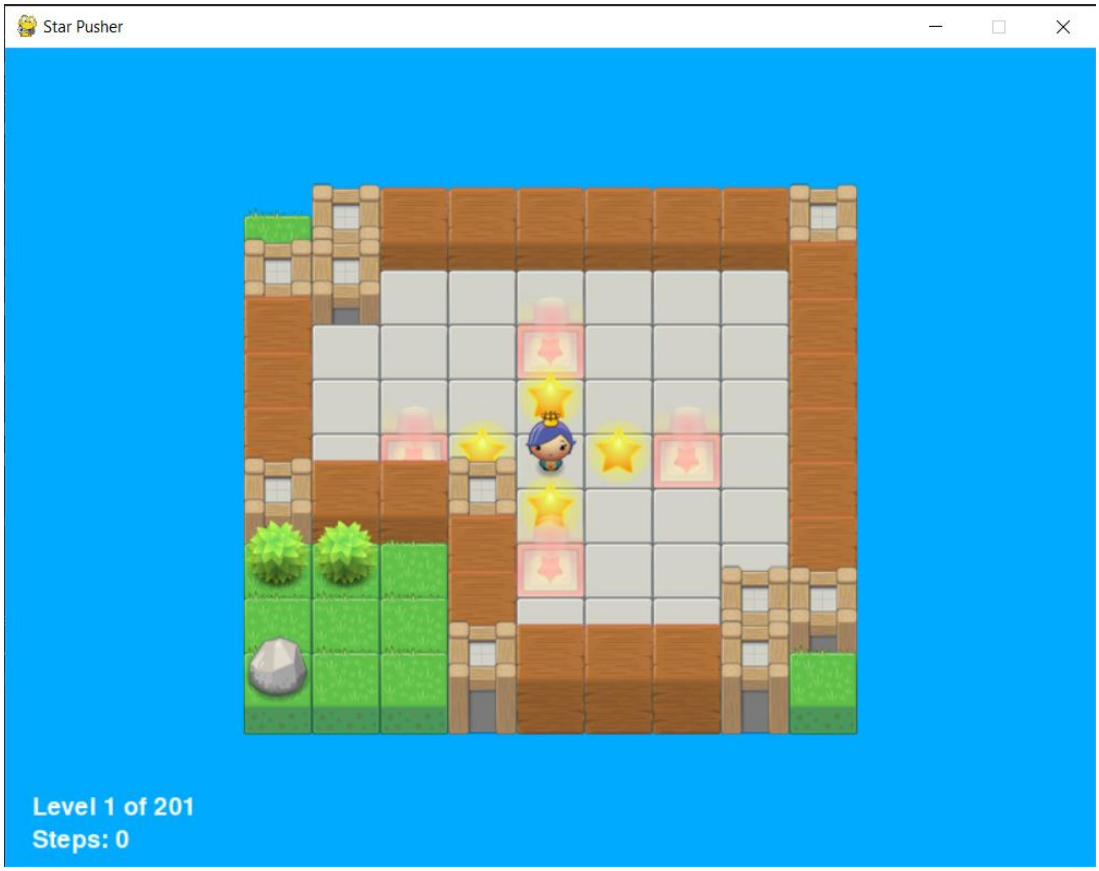
Starpusher работа на час

1. Слика од почетен екран



2. Make an evidence of the time that the player needs to finish each level.

```
def runLevel(levels, levelNum):
    ...
    # Change for requirement 2:
    start_time = time.time()
    time_taken = 0  # time taken is 0 at the start
```

Во функцијата runLevel додаваме променливи кои ќе го означуваат времето на почеток на ниво и колку време играчот го решава тоа ниво.

```
if isLevelFinished(levelObj, gameStateObj):
    # level is solved, we should show the "Solved!" image.
    levelIsComplete = True
    keyPressed = False
    # Change for requirement 2:
    time_taken = time.time() - start_time
```

Во main loop во делот каде се проверува дали нивото е решено додаве код со кој го пресметуваме времето кое му било потребно на играчот да го реши нивото.

```
# Change for requirement 2:
if not levelIsComplete:
    time_taken = time.time() - start_time
# draw the time taken message
timeSurf = BASICFONT.render('Time taken: %ss' % (round(time_taken,
1)), 1, TEXTCOLOR)
timeRect = timeSurf.get_rect()
timeRect.bottomleft = (20, WINHEIGHT - 60)
DISPLAYSURF.blit(timeSurf, timeRect)
```

Потоа додаваме код со кој времето поминато на нивото го исцртуваме на екран. Доколку нивото не е решено го прикажуваме времето изминато од почетокот на нивото, а доколку нивото е решено го прикажуваме времето потребно за да се реши.

3. Change the number of tiles that have additional decoration on them.

```
# Change for requirement 3:  increasing the chance an outside tile to
be decorated
# OUTSIDE_DECORATION_PCT = 20
OUTSIDE_DECORATION_PCT = 50
```

Ја зголемуваме вредноста на константата која означува шанса надворешно поле да биде декорирано.

4. Change the game so that the next level is available only if the current is solved.

```
while True:  # main game loop
    # Run the level to actually start playing the game:
    result = runLevel(levels, currentLevelIndex)

    # Change for requirement 4:
    # if result in ('solved', 'next'):
    if result == 'solved':
        # Go to the next level.
        currentLevelIndex += 1
        if currentLevelIndex >= len(levels):
            # If there are no more levels, go back to the first one.
            currentLevelIndex = 0
    elif result == 'back':
        # Go to the previous level.
        currentLevelIndex -= 1
        # Change for requirement 4:
        if currentLevelIndex < 0:
            # currentLevelIndex = len(levels) - 1
            # If there are no previous levels, reset the level.
            currentLevelIndex = 0
```

Во main() во main loop го менуваме кодот така што преминување на следно ниво ќе се случува само ако нивото е решено односно result има вредност 'solved', дополнително, доколку кликнеме back а сме на почетното ниво наместо да се пушти последното ниво ќе се пушти почетното ниво од почеток. Со ова обезбедуваме дека нема да има скокања на нивоа, доколку не се поминати односно решени.

```
# Change for requirement 4:
# don't allow the player to skip levels
# elif event.key == K_n:
#     return 'next'
```

Во main loop во runLevel во делот каде што се обработуваат настаните (event handling loop) ги отстрануваме(коментираме) наредбите со кои се преминува на следно ниво.

5. Add a new object within the map (an image that represents a rock) on a random position. This object should block the movement on that position.

```
def runLevel(levels, levelNum):
    global currentImage
    levelObj = levels[levelNum]
    mapObj = decorateMap(levelObj['mapObj'],
levelObj['startState']['player'])
    # Change for requirement 5:
    mapObj = addRandomRock(mapObj, levelObj['startState']['player'],
levelObj['startState']['stars'], levelObj['goals'])
```

во runLevel по повикот на функцијата decorateMap повикуваме функција addRandomRock која додава објект камен во мапата го означуваме со '?'.

```
# Change for requirement 5:
# put one rock on a random position in the inside of the map
def addRandomRock(map, player, stars, goals):
    tiles = []  # get all inside floor tiles in the map
    for x in range(len(map)):
        for y in range(len(map[0])):
            if map[x][y] == 'o':
                tiles.append((x, y))
    emptyTiles = []
    # make sure the rock won't be placed on the same tile as a star,a
goal or the player
    for cords in tiles:
        if not ((cords in stars) or (cords in goals) or (cords in
[player])):
            emptyTiles.append(cords)
    rockCordx, rockCordy = random.choice(emptyTiles)  # get a random
tile where to put the rock from emptyTiles
    map[rockCordx][rockCordy] = '?'  # mark the tile as rock

    return map
```

Креираме функција addRandomRock која како аргументи ги прима мапата после извршената декорација, координатите на играчот, листа со координати на ѕвездите и листа со координати на целите. Најпрви во листа ги додаваме сите полиња кои се наоѓаат во внатрешноста односно се означени со 'o', а потоа во нова листа ги додаваме само тие полиња на кои што не се наоѓа друг објект (ѕвезда,цел или играч). Следно од

оваа листа по случаен избор избираме поле и конечно тоа поле во мапата го означуваме со ‘?’ (камен).

```
def isWall(mapObj, x, y):
    """Returns True if the (x, y) position on
    the map is a wall, otherwise return False."""
    if x < 0 or x >= len(mapObj) or y < 0 or y >= len(mapObj[x]):
        return False  # x and y aren't actually on the map.
    # Change for requirement 5:
    # treat rock object '?' as wall
    # elif mapObj[x][y] in ('#', 'x'):
    elif mapObj[x][y] in ('#', 'x', '?'):
        return True  # wall is blocking
    return False
```

Ја менуваме функцијата isWall така што и полињата означени со ‘?’ односно како камења ги третираме како ѕид, играчот неможе да ги турка или да поминува низ нив.

```
# Change for requirement 5:
# the rock object has an inner tile as a base tile
if mapObj[x][y] == '?':
    baseTile = TILEMAPPING['o']
# First draw the base ground/wall tile.
mapSurf.blit(baseTile, spaceRect)
```

```
# Change for requirement 5:
# draw the rock
elif mapObj[x][y] == '?':
    mapSurf.blit(IMAGESDICT['rock'], spaceRect)
```

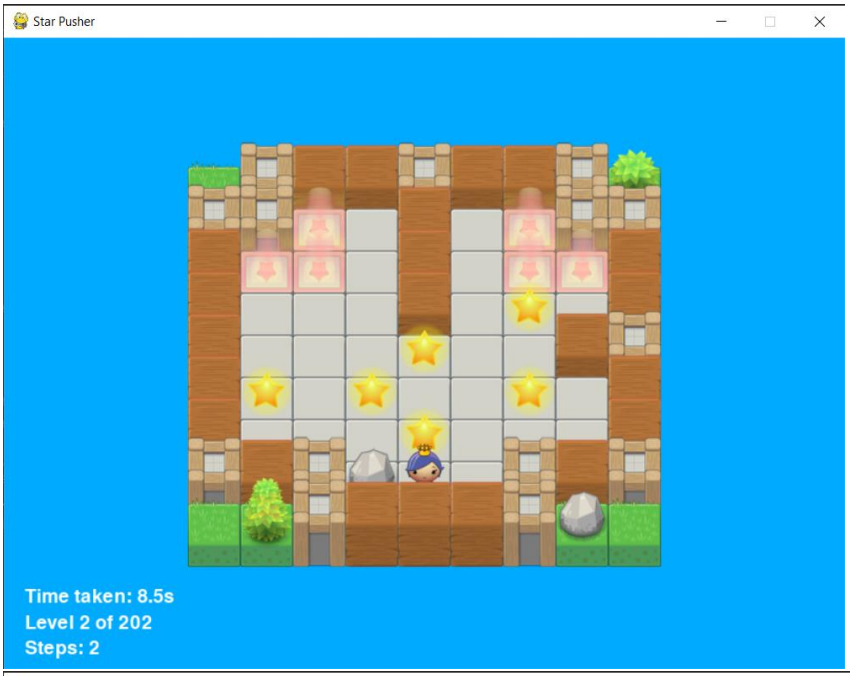
Во функцијата drawMap додаваме код со кој ги исцртуваме полињата означени како камења (‘?’) така што како позадинска слика е сликата од внедрешно поле (inner tile) а потоа врз неа се исцртува сликата на камен (rock).

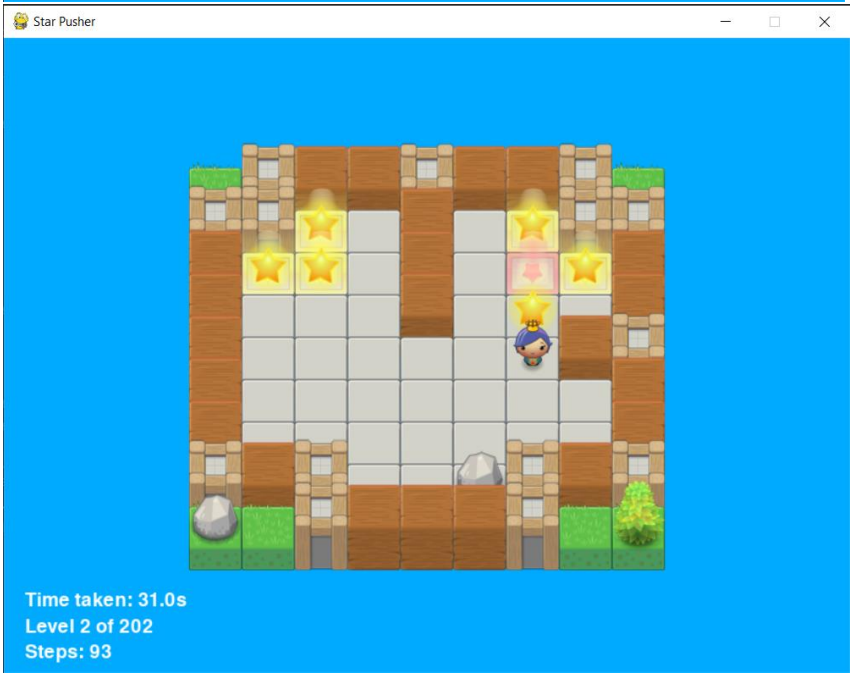
6. Add a new level map in the appropriate file as a second level and update the game appropriately.

   ; Change for requirement 6:

   ```
    #######
   ##. # .##
   #.. # ..#
   #   # $ #
   #   $  ##
   #$ $  $ #
   #   $   #
   ### @ ###
    #####
   ```

   Промената ја правиме само во starPusherLevels фајлот со нивоа.


Star Pusher
Time taken: 8.5s
Level 2 of 202
Steps: 2


Star Pusher
Time taken: 31.0s
Level 2 of 202
Steps: 93