1. First we change the kame movements to the keyarrows , as left is for cameraLeft , right is for cameraRigh and so on.

```
            playerMoveTo = DOWN

        # Set the camera move mode.
        elif event.key == K_LEFT:
            cameraLeft = True
        elif event.key == K_RIGHT:
            cameraRight = True
        elif event.key == K_UP:
            cameraUp = True
        elif event.key == K_DOWN:
            cameraDown = True
```

```
    elif event.type == KEYDOWN:
        # Unset the camera move mode.
        if event.key == K_LEFT:
            cameraLeft = False
        elif event.key == K_RIGHT:
            cameraRight = False
        elif event.key == K_UP:
            cameraUp = False
        elif event.key == K_DOWN:
            cameraDown = False
```

-For the keys to proceed and move the character, we use the numpad buttons so we make numpad key 4 to be left , 6 for right and etc etc.

```
    elif event.type == KEYDOWN:
        # Handle key presses
        keyPressed = True
        if event.key == K_KP4:
            playerMoveTo = LEFT
        elif event.key == K_KP6:
            playerMoveTo = RIGHT
        elif event.key == K_KP8:
            playerMoveTo = UP
        elif event.key == K_KP5:
            playerMoveTo = DOWN
```

2.For the random level picker when we click N , we makde that the currentLevelIndex is a random integer from 0 to the maxamount of levels in levels. Then we return the result to the game, so everytime

we press N as N is next, we get random lvl, if we solve it we go to next lvl

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


    if result in ('next'):
        currentLevelIndex = random.randint(0,levels.__len__())  #the index when we press N button will be a random from the amount
        #of total levels and 0 , so we return as a value the random amount so everytime is random level from 0 to length of levels
        result = runLevel(levels, currentLevelIndex)
    if result in ('solved', 'next'):
        # Go to the next level.
        currentLevelIndex += 1

    elif result == 'back':
        # Go to the previous level.
        currentLevelIndex -= 1
        if currentLevelIndex < 0:
            # If there are no previous levels, go to the last one.
            currentLevelIndex = len(levels)-1
    elif result == 'reset':
        pass # Do nothing. Loop re-calls runLevel() to reset the level
```

3. I made that the game renders one more three in the game , so we put the Tree_Trees jpg as novo drvo and we put it in the dictionary , then we mark it as index 5 , so we can use it in the rendering afterwards

```
                'ugly tree': pygame.image.load('Tree_Ugly.png'),
                'novo drvo': pygame.image.load('Tree_Treee.jpg')} #we load the new tree in the game

# These dict values are global, and map the character that appears
# in the level file to the Surface object it represents.
TILEMAPPING = {'x': IMAGESDICT['corner'],
               '#': IMAGESDICT['wall'],
               'o': IMAGESDICT['inside floor'],
               ' ': IMAGESDICT['outside floor']}
OUTSIDEDECOMAPPING = {'1': IMAGESDICT['rock'],
                      '2': IMAGESDICT['short tree'],
                      '3': IMAGESDICT['tall tree'],
                      '4': IMAGESDICT['ugly tree'],
                      '5': IMAGESDICT['novo drvo']}  #we put the new tree in dictionary
```

-So we see in the next image that the trees are rendered in the game at random spots so it works.