**ЛАБОРАТОРИСКА ВЕЖБА 2**

02.12.2020

**ПРОГРАМИРАЊЕ НА**

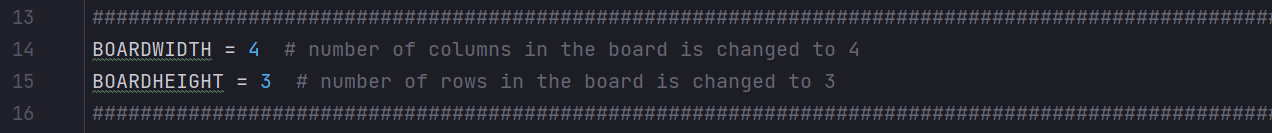
**ВИДЕО ИГРИ И СПЕЦИЈАЛНИ ЕФЕКТИ**

*Slide puzzle game*

Професор*:* д-р Катарина Тројачанец Динева

Студент*:* Кирил Зеленковски

1. ***Барање 1***

Бројот на полиња во редиците и колоните да биде различен (на пример 8 на 6)

1. ***Барање 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.

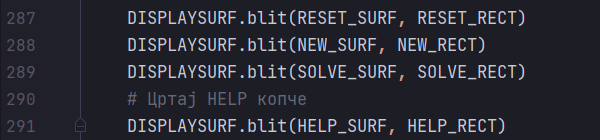
Новото копче го правам на сличен начин како што се првобитните копчиња:   
**Reset**, **New** **Game** и **Solve**

Сите три копчиња е потребно да се нацртаат користејќи ја функцијата **DISPLAYSURF.blit().** Оваа функија на влез прима тоа што сите три кочиња имаат заедничко (две променливи):

* **Surface** променлива (**RESET\_SURF, NEW\_SURF, SOLVE\_SURF**)
* **Rectangle** променлива (**RESET\_RECT, NEW\_RECT, SOLVE\_RECT**)

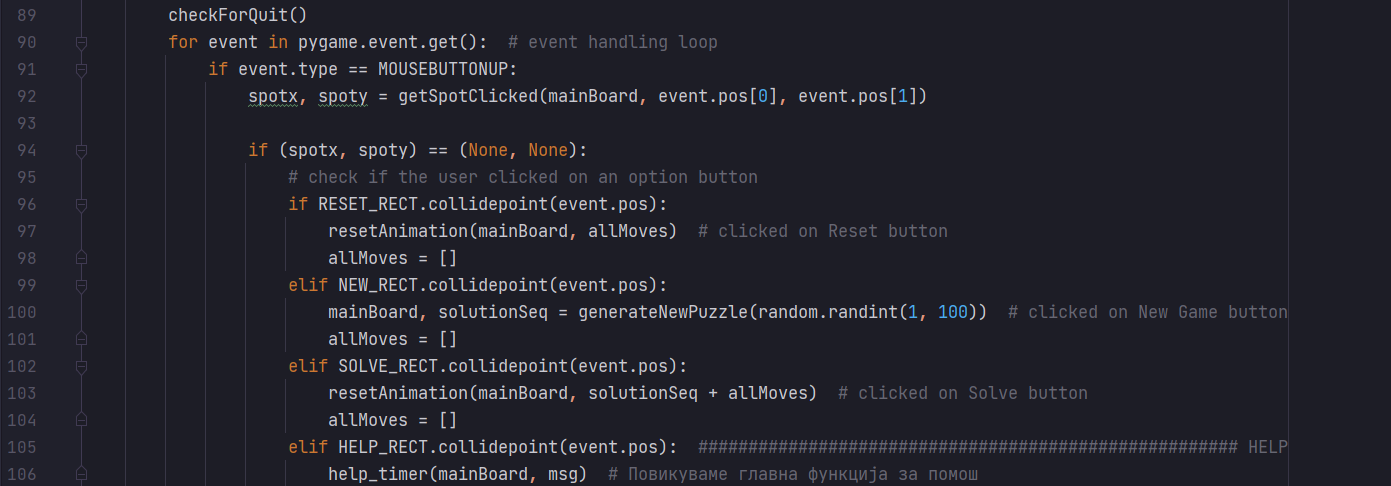
Со таа цел дефинирам две нови променливи за новото копче (многу важно и двете ги поставувам да бидат „глобални“ бидејќи ги користиме надвор од функцијата main() )

Со цел да се исцрта копчето потребно е да се употреби погоре споменатата функција која како влезни параметри и ги задаваме новите променливи:

Table

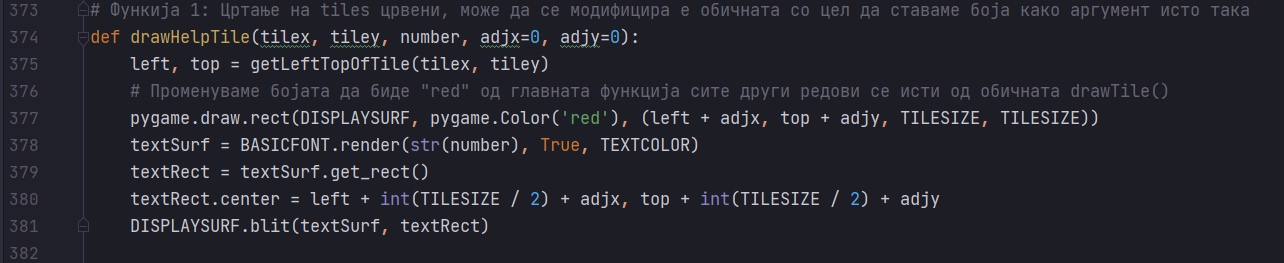
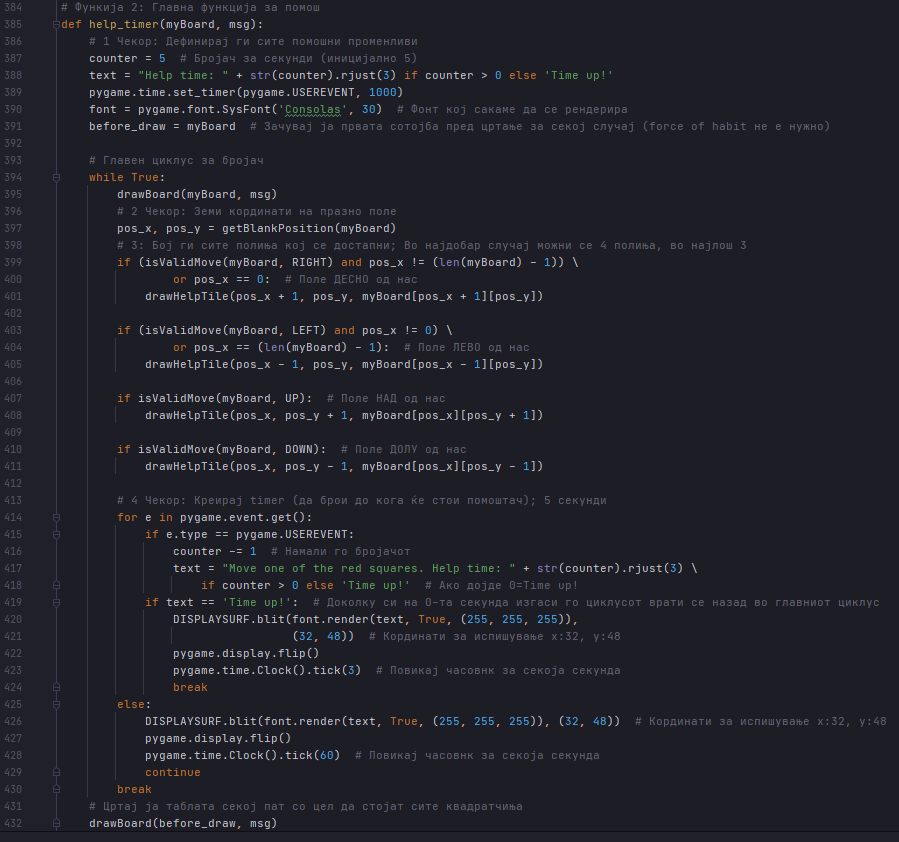
Description automatically generated

Со овие команди до оваа фаза во прозорецот од играта ќе гледаме копче за помош, меѓутоа истото нема никаков “*listener*” и нема повратна функција.

Со цел да оневозможиме некаква функција на копчето “**HELP**” мора да поставиме некој вид на “*listener*” кој кога копчето ќе се кликне ќе ги исцрта соседните полиња црвени.   
(Тоа го правиме во 97 ред каде правиме проверка и потоа во следниот повикуваме ф-ја)

Слично како сите event *listeners* за другите 3 копчиња таков е и за 4-тото копче. Задачата иако беше само да се обојат црвени можните чекори, јас сакав да имитирам нешто слично на игра и да има како тајмер за колку време помошта ќе биде на екран и ова го реализирав внатре во функцијата help\_timer().

За овој тајмер користам две функции:

* Функција 1   
  **drawHelpTile()**: За цртање на tiles црвени. Идејата е да имаме посебна функција но и обичната е океј ако се модифицира да има некој влезен параметар за боја бидејќи обичната drawTile() ги бои со предефинирана боја
* Функција 2  
  help\_timer(): Главна функција, која бои коцки во околина (десно, лево, горе доле доколку се валидни црвени) и во исто време почнува тајмер од 5 секунди со цел да имитира време за помош

Table

Description automatically generated

Слика 1: Додадено HELP копче

Слика 2,3: По кликнување на HELP копчето

Chart, treemap chart

Description automatically generatedChart, treemap chart

Description automatically generatedОвој бројач одбројува (5 – 1, кога доаѓа 0 вика “*Time* *up*!”) и се враќа на Слика 1.

1. ***Барање 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").

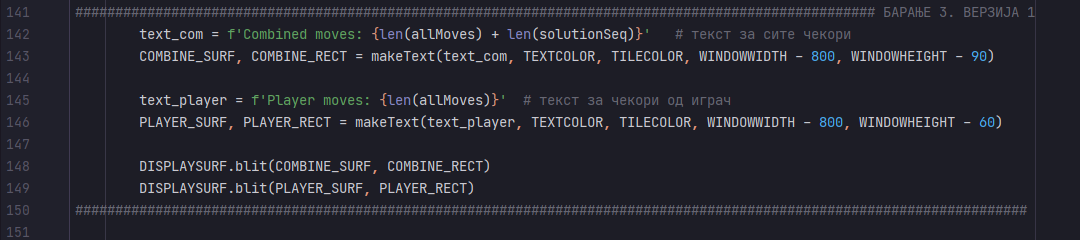
Како што пишува во барањето ова решение ја согледува логиката на “Solve” потребни counters за:

* Бројот на чекори направени од почеток до решавање
* Бројот на чекори од страна на компјутер

Верзија 1:

**Бројач на чекори од компјутер**

Овие се добиваат како збир од листата **allMoves** (каде се наоѓаат во обратен редослед сите преместување од играта што играчот ги прави) и листата од чекори **solutionSeq** каде се наоѓаат чекорите од тој “random scrabble” што самиот компјутер ги прави кога почнува играта со цел да ја размести од почетната состојба.

**Бројач на чекори од играч**

Овие се добиваат со читање на должината на **allMoves** листата.

\*\*\*И двете се претставуваат како текст со surface\*\*\*

идејата за вака е слична од копчето за “**HELP**”   
може и со обичен фонт но, вака повеќе ми се допаѓаат и брзо ги направив

Text, table

Description automatically generated

Text, table

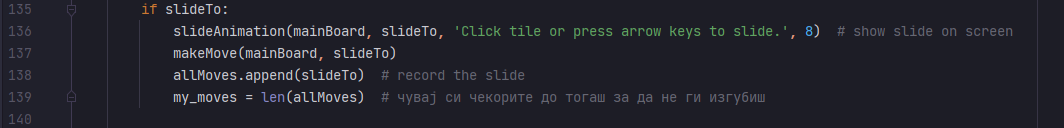
Description automatically generatedСлика 4: Почетна состојба на бројачи

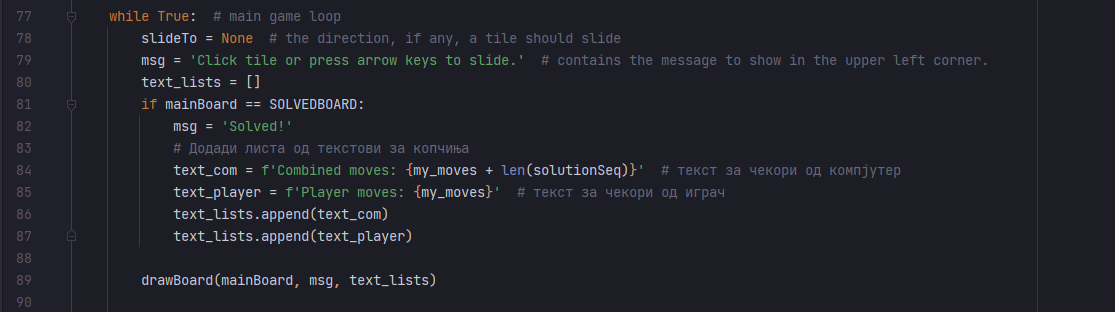
Слика 5: Состојба на бројачи по едно поместување

Верзија 2:

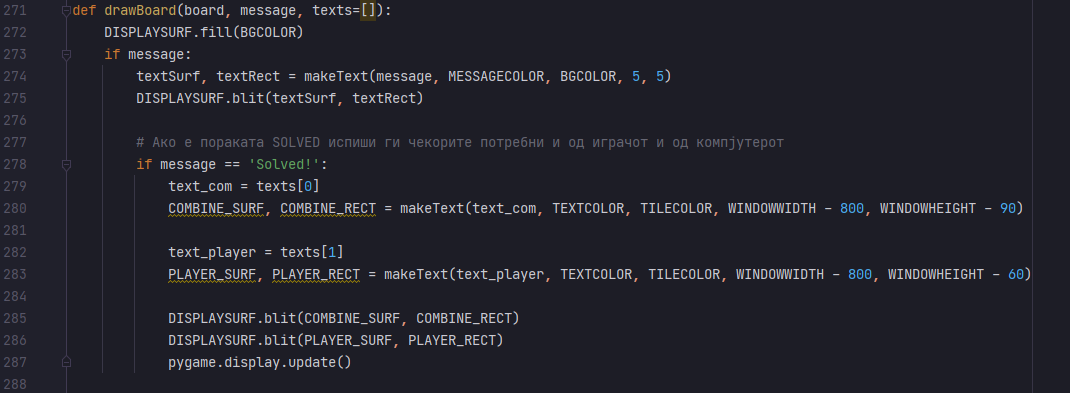
Бидејќи не бев сигурен дали треба да се испише текстот на крај или цело време да се прикажуваат го модифицирав кодот и го прикачив со оваа втора верзија. Тука ги испишувам чекорите само кога е решена играта.

За оваа верзија сепак бројач беше нужен бидејќи при ресетирање на играта сите чекори од allmoves листата се бришат (од каде и ми дојде логиката дека тоа што го има во барањето има смисла со бројач а не само со len(allmoves)).

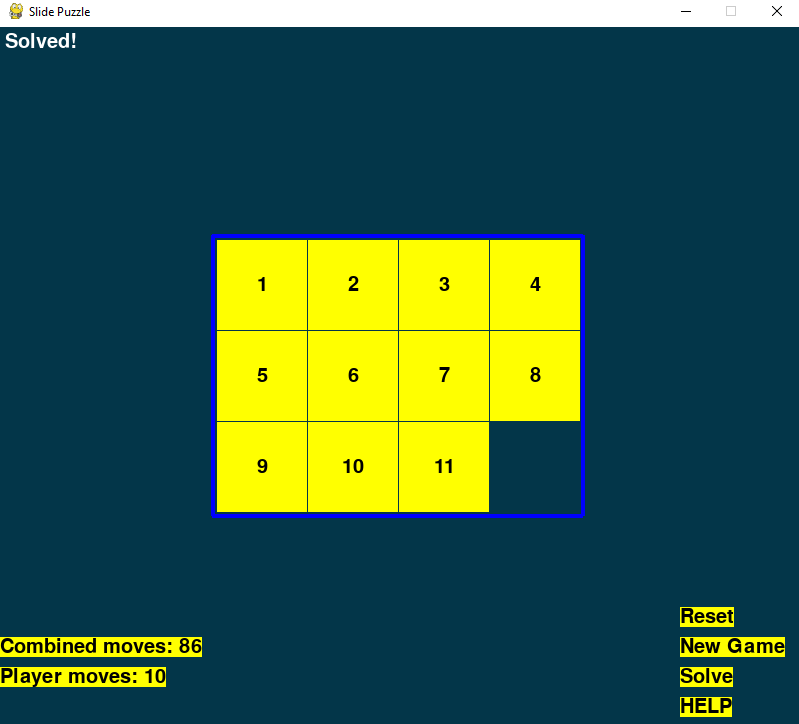
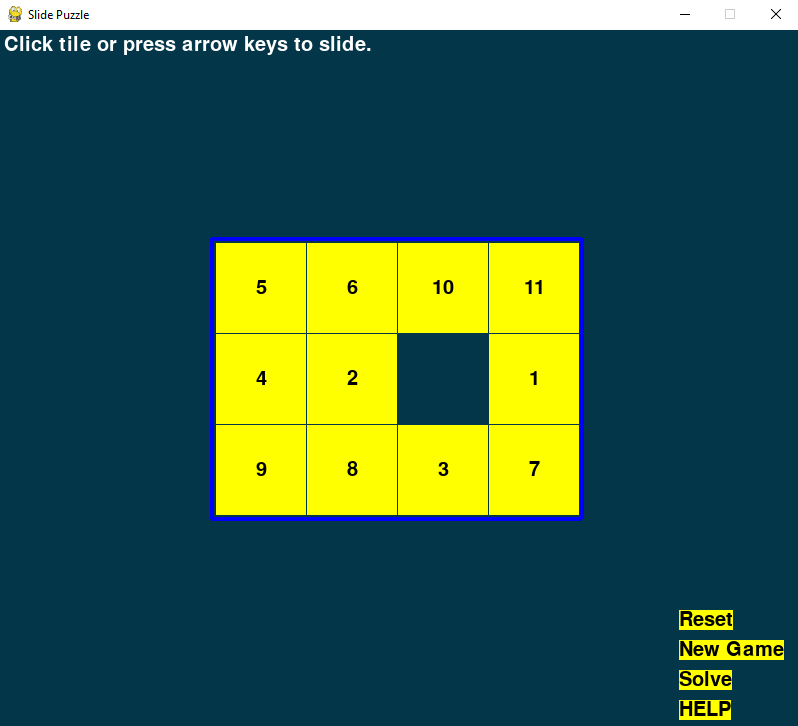
Бројачот не е класичен туку секој пат кога се додава чекор, го иницирам да биде еднаков на должината од листата allmoves во тој момент.

Оваа верзија малку покомплицирано ја направив од што треба но навистина сакав да видам дали можам и на крајот на играта да испишувам.

Правам листа од двете пораки и истите ги праќам во главната функција бидејќи секогаш се повикува. Тука само исто како во верзија 1 ги испишувам повторно.



Резултатот е тоа што при играње не ги гледаме но ако ја решиме играта (или одиме со solve) ќе ги видиме сите чекори потребни.



Слика 6, 7: Состојба без чекори (лево). Состојба по 10 чекори и кликнување на Solve (десно)