**ЛАБОРАТОРИСКА ВЕЖБА 4**

30.12.2020

**ПРОГРАМИРАЊЕ НА**

**ВИДЕО ИГРИ И СПЕЦИЈАЛНИ ЕФЕКТИ**

*Wormy*

Професор*:* д-р Катарина Тројачанец Динева

Студент*:* Кирил Зеленковски

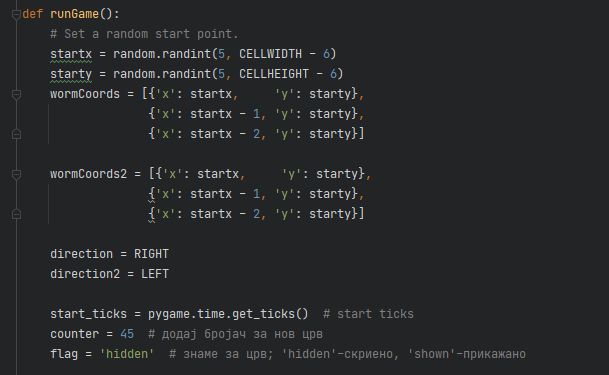
1. ***Барање 1***

По 45 секунди од почетокот на играта додадете друг црв. Оригиналниот црв треба да го избегнува вториот црв. Ако го допре со главата, неговото тело расте за еден сегмент. Ако вториот црв го допре оригиналниот, тогаш неговото тело се продолжува за еден сегмент. Движењето на вториот црв е случајно.

Пред се за ова барање потребен е тајмер од 45 секунди за прикажување на втор црв. Овој тајмер е потребно да започне во исто време кога играта започнува и потоа да го прикаже вториот црв кој има случајно движење.

Имплементацијата на тајмерот беше слична како барањето со јаболките (жолто и плаво) од работата на час. Започнуваме со знаменце за дали црвот е прикажан, на почетокот црвот е со знаме “hidden” и доколку поминат 45 секунди црвот почнува да се прикажува.

Ова барање може да се разложи на 3 компоненти:

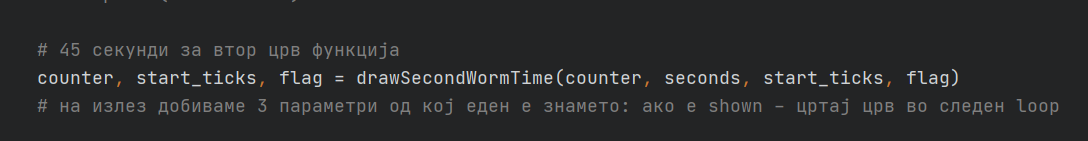
* Тајмер за втор црв
* Рандом движење на црв
* Колизија на црвите
  + Тип 1 колизија: Прв црв го допира втор црв со својата глава (расте сегмент)
  + Тип 2 колизија: Втор црв го допира првиот црв било каде по телото (расте сегмент)

## Тајмер за втор црв

Тајмерот е всушност бројач кој почнува од 45 и се движи до 0, кога доаѓа на нула се испишува само порака да се избегнува црвот.

Се дефинира пред се знамето на “hidden”, бројачот на 45 и се зимаат почетните ticks од почетокот на играта. Кога ќе дојдат на 45 се менува знамето.

Имплементацијата започнува во главниот game loop:



Функцијата враќа 3 аргументи кои ги процесира на следниот начин:



Оваа функција се состои од 2 главни проверки: каква е состојбата на знамето и дали секундите се еднакви на тајмерот. Доколку е состојба да криење (“hidden”), тогаш тајмерот врти додека не се исти. Секундата кога се исти преминува знамето во состојба прикажување (“shown”) и започнува да испишува “Avoid second worm!”. Знамето е многу битно и тоа ќе го објаснам зошто во следните страни.

## Рандом движење на втор црв

Ова движење зависи многу од знамето во претходната функција. Сега, веќе кога знамето е “shown” имаат поминато 45 секунди и можеме да прикажуваме црв. Ова го правиме така што црвот го претставуваме со листа од речници со x,y како клучеви и вредности кординати. Ваква листа има првиот црв, правиме со слично име за вториот која се вика **wormCoords2**.

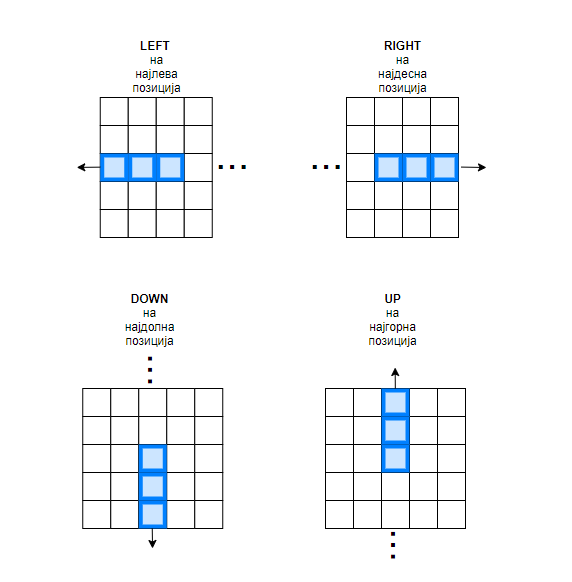
Иницирана е со исти почетни 3 сегменти дефинирани внатре во играта (исти како првиот).

Сега прво што напрвив е после прикажување на црвот со трите сегменту му давав рандом насока од листа со насоки, со random.choice(directions), каде directions е листа од сите можни насоки (LEFT, UP, ...) и потоа до движев слично како првиот што е придвижен од нашата насока.

Но, иако ми делуваше доста логично на прв поглед, овде има грешка доколку не правиме **проверки** црвот за многу кусо време ќе **излезе** од играта односно:

* x кординатите на сегментите ќе бидат:
  + Негативни, т.е. лево од играта
  + поголеми од ширината (CELLWIDTH), т.е. десно од играта
* y кординатите на сегментите ќе бидат:
  + негативни, т.е. под игратата
  + поголеми од висината (CELLHEIGHT), т.е. над од играта

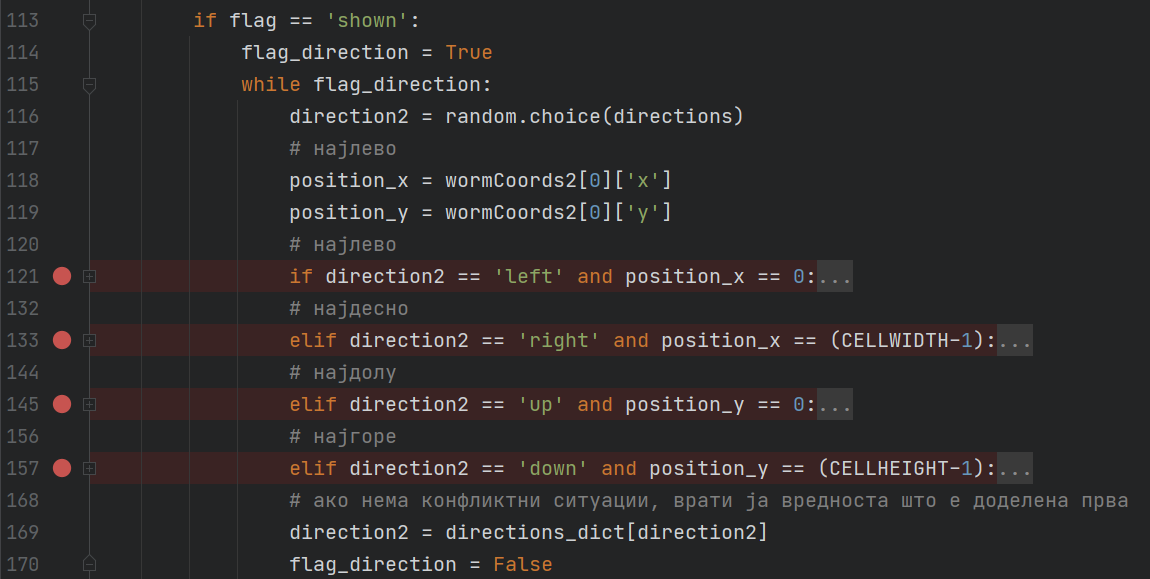
Тука стана доста зезнато за имплементација на движење на вториот црв, бидејќи доколку даваме рандом вредности од листа со движења можни веројатно е дека ќе го снема брзо. Поради овој проблем, пробав да направам проверки со цел “регулирано ” случајно да го движиме, проверките се засоноваат на каде е главата на вториот црв (првиот елемент од листата).

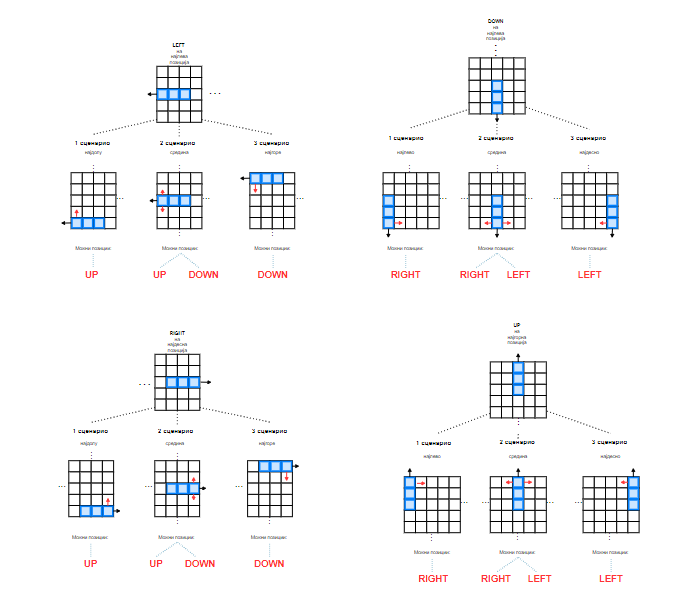
Секоја една “критична ситуација” ја разгранувам како сценарио кое резултира со можни насоки при таквии ситуации. Вакви “критични ситуации” се 4:

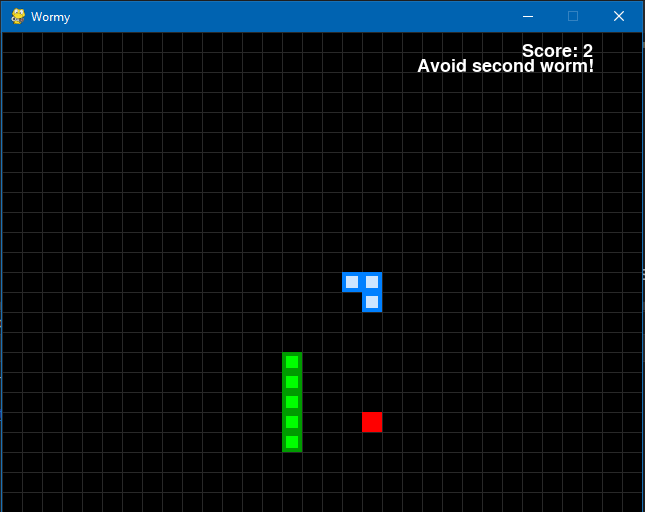
* Се наоѓаме најлево во игра а добиваме движење од случајен избор LEFT
* Се наоѓаме најдесно во игра а добиваме движење од случајен избор RIGHT
* Се наоѓаме најгоре во игра а добиваме движење од случајен избор UP
* Се наоѓаме најдолу во игра а добиваме движење од случајен избор DOWN

За цртање на сите следни цртежи како и овој користам: [draw.io](https://www.diagrams.net/)

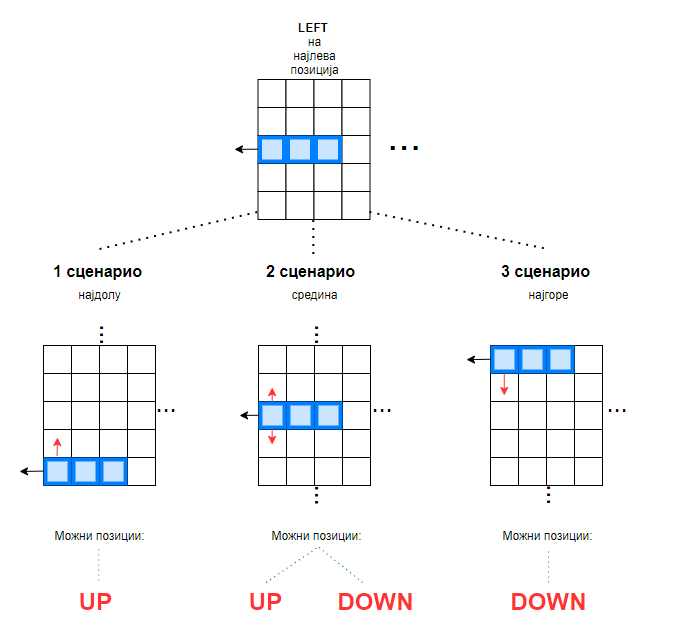
Сега овие 4 проверки се во самиот код имплементирани како функција која изгледа вака (ги направив како collapse за да може да се гледа идејата ќе ја објаснам во целост подолу):



Значи самата функција ги прави овие проверки и потоа секоја една од овие критични ситуации е исполнета внатре во секоја итерација имаме разгранување на 3 можни сценарија кој се повеќе околку местоположбата на самиот црв (дали доколку е надоле е во средина или на десен ќош бидејќи ако е пр во десен ќош нема да има логика да му дадеме десно да се движи ќе избега со главата и повторно лоша проверка). Со таа цел го направив ова поголемо дрво:



**4 УСЛОВИ. 12 Сценарија.**  
Ова се сите можни “сценарија” предизвикани од нашите 4 “критични ситуации”. Доаѓа нешто како дрво на одлука. (ЛЕВО)   
  
Црвот во скиците го направив да биде плав, гледав истата боја да е и во играта (ДЕСНО).

**1 УСЛОВ:** Добиена случајна избрана насока за втор црв LEFT а се наоѓа на најлева позиција

**1 УСЛОВ. 1 Сценарио:** *најдолу*

Ова е доколку црвот е најлево и во исто време најдолу, тука насока LEFT или DOWN нема логика бидејќи изглегува од игра, а насока RIGHT ќе има два дупликата сегмента и ја намалува должината на некој начин (барем визуелно). Од каде можна насока е **UP**

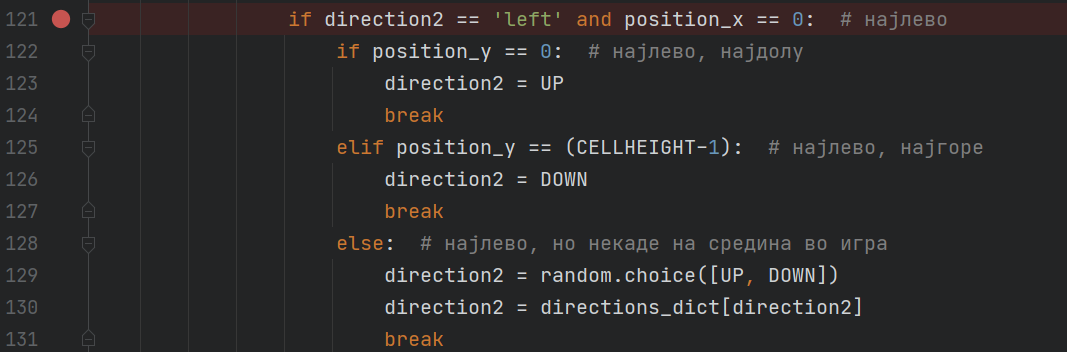
**1 УСЛОВ. 2 Сценарио:** *средина*

Ова е доколку црвот е најлево но не е ниту најгоре ниту најдоле. Тука врз основа на случаен избор се дава една од двете можни логички насоки: **UP** или **DOWN**

**1 УСЛОВ. 3 Сценарио:** *најдолу*

Идентично на првото само најгоре се наоѓа црвот. UP не е логично што значи единствена можна насока е **DOWN**

Овој услов искодиран изгледа вака:

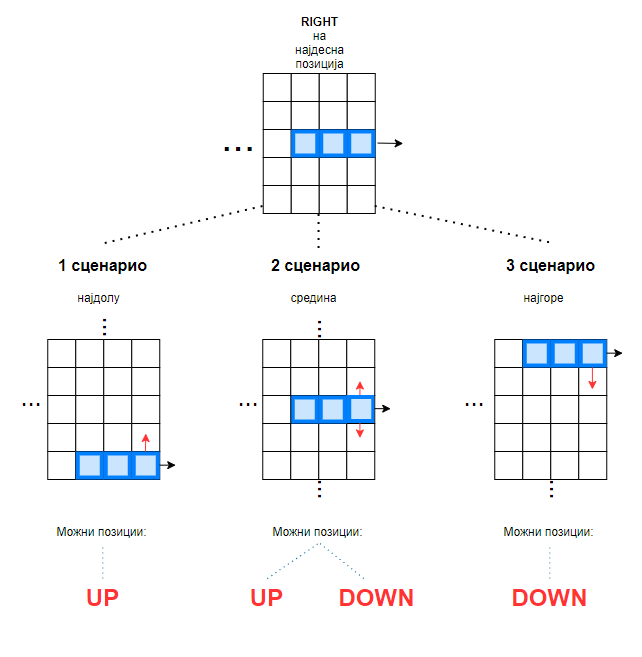


**1 УСЛОВ. 1 Сценарио:** Доколку сме најлево и најдолу даваме UP како следна насока и го прекинуваме циклусот кој е всушност додека не најдеме насока.

**1 УСЛОВ. 2 Сценарио:** Доколку сме во средина правиме листа од двете можни насоки за да имаме 50-50 шанси и со функцијата random.choice() добиваме една од тие две и повторно прекинуваме.

**1 УСЛОВ. 3 Сценарио:** Доколку сме најлево и најгоре даваме насока DOWN како следна и го прекинуваме циклусот.

**2 УСЛОВ:** Добиена случајна избрана насока за втор црв RIGHT а се наоѓа на најдесна позиција

**2 УСЛОВ. 1 Сценарио:** *најдолу*

Ова е доколку црвот е најдесно и во исто време најдолу, исто како 1 УСЛОВ, и тука насока LEFT или DOWN немаат логика бидејќи изглегува од игра, а насока LEFT ќе има два дупликата сегмента и ја намалува должината на некој начин (барем визуелно). Од каде можна насока е **UP**

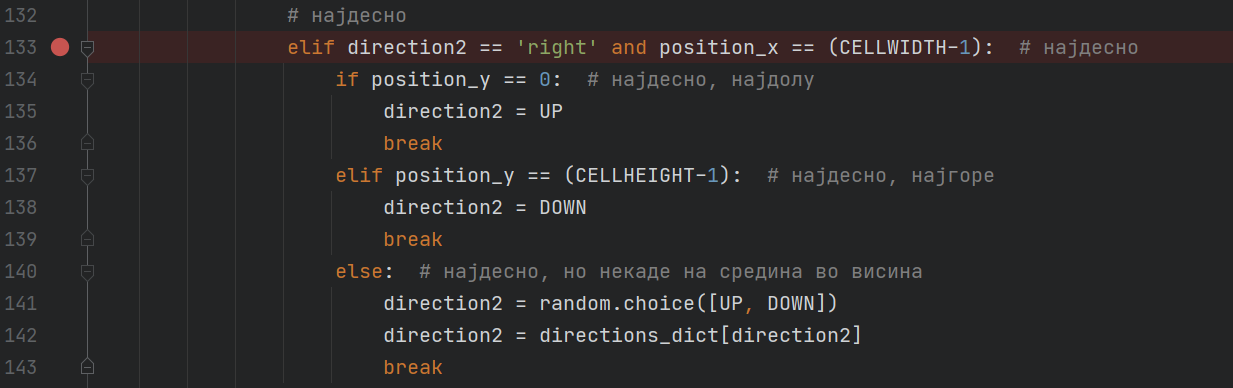
**2 УСЛОВ. 2 Сценарио:** *средина*

Ова е доколку црвот е најдесно но не е ниту најгоре ниту најдоле. Тука врз основа на случаен избор се дава една од двете можни логички насоки: **UP** или **DOWN**

**2 УСЛОВ. 3 Сценарио:** *најдолу*

Идентично на првото само најгоре се наоѓа црвот. UP не е логично што значи единствена можна насока е **DOWN**

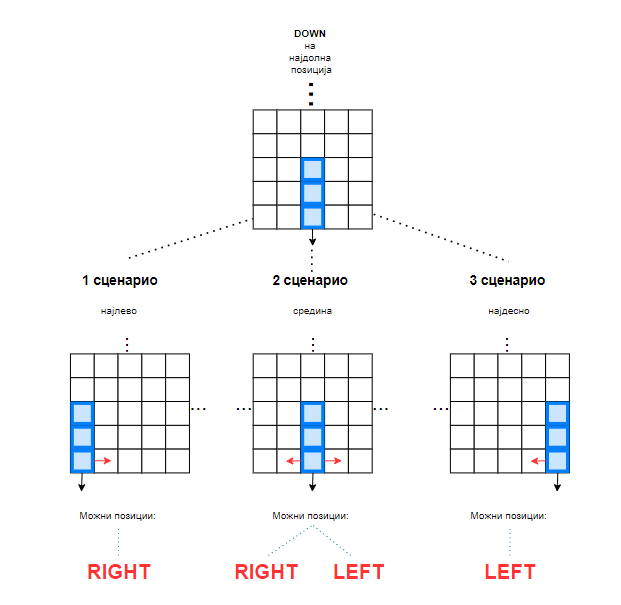
Овој услов искодиран изгледа вака:



**2 УСЛОВ. 1 Сценарио:** Доколку сме најдесно и најдолу даваме UP како следна насока и го прекинуваме циклусот.

**2 УСЛОВ. 2 Сценарио:** Доколку сме во најденсо во средина правиме листа од двете можни насоки, случајно бираме и повторно прекинуваме.

**2 УСЛОВ. 3 Сценарио:** Доколку сме најдесно и најгоре даваме насока DOWN како следна и го прекинуваме циклусот.

**3 УСЛОВ:** Добиена случајна избрана насока за втор црв DOWN а се наоѓа на најдолна позиција

**3 УСЛОВ. 1 Сценарио:** *најлево*

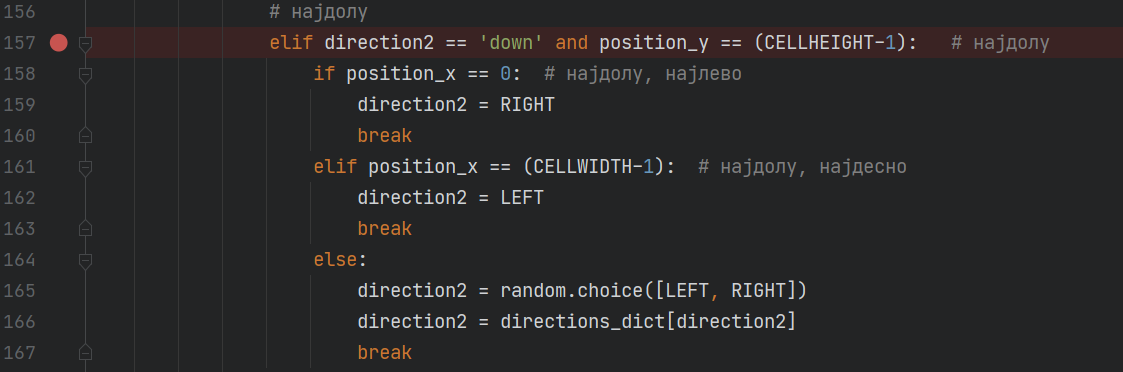
Ова е доколку црвот е најдоле и во исто време најлево слично како претходните услови ги бараме логичните наски тука е само една а таа е десно со што следна насока е **RIGHT**

**3 УСЛОВ. 2 Сценарио:** *средина*

Ова е доколку црвот е најдолу но не допира ѕидови и врз основа на случаен избор се дава една од двете можни логични насоки: **RIGHT** или **LEFT**

**3 УСЛОВ. 3 Сценарио:** *најдесно*

Идентично на првото само сме најдолу најдесно што значи единствена можна насока е **LEFT**

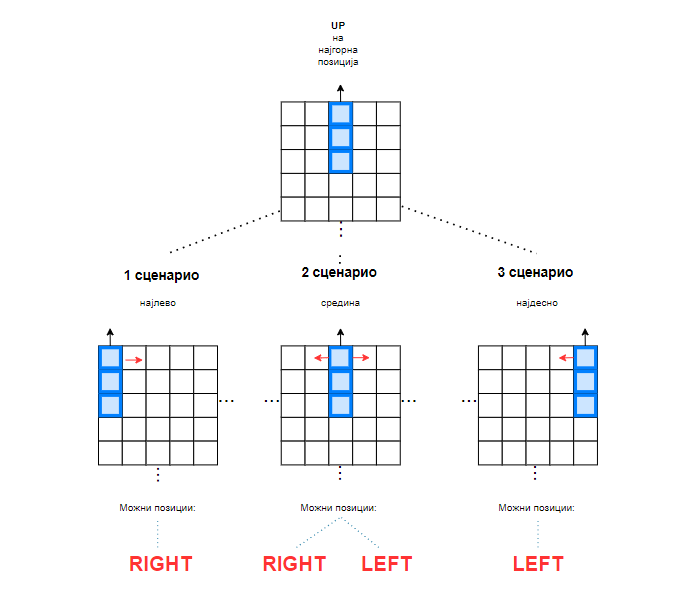
Овој услов искодиран изгледа вака:

Иако на прв поглед изгледа грешно, не е. Обратно се се горе и долу во кодирање, ова е зависност од како е поставен кординатниот ситем.

**3 УСЛОВ. 1 Сценарио:** Доколку сме најдолу и најлево даваме RIGHT како следна насока и го прекинуваме циклусот.

**3 УСЛОВ. 2 Сценарио:** Доколку сме во најдолу во средина правиме листа од двете можни насоки, случајно бираме и повторно прекинуваме (RIGHT или LEFT)

**3 УСЛОВ. 3 Сценарио:** Доколку сме најдолу и најдесно даваме насока LEFT како следна и го прекинуваме циклусот.

 **4 УСЛОВ:** Добиена случајна избрана насока за втор црв UP а се наоѓа на најгорна позиција

**4 УСЛОВ. 1 Сценарио:** *најлево*

Ова е доколку црвот е најгоре и во исто време најлево слично како претходниот услов ги бараме логичните наски тука е само една а таа е десно со што следна насока е **RIGHT**

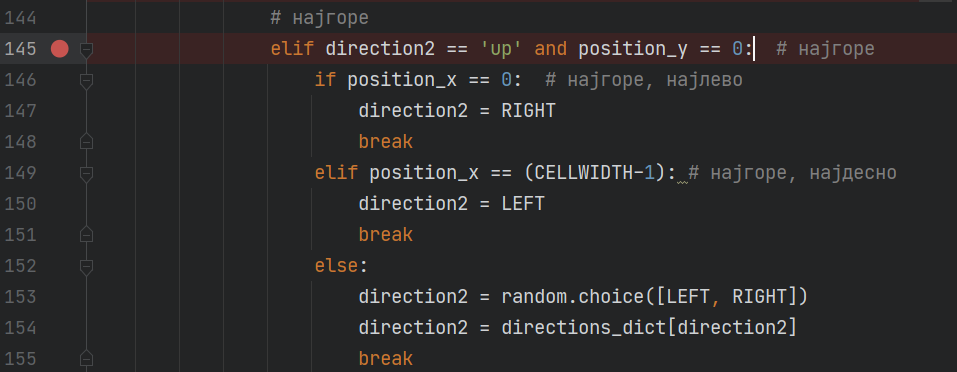
**4 УСЛОВ. 2 Сценарио:** *средина*

Ова е доколку црвот е најгоре но не допира ѕидови и врз основа на случаен избор се дава една од двете можни логични насоки: **RIGHT** или **LEFT**

**4 УСЛОВ. 3 Сценарио:** *најдесно*

Идентично на првото само сме најгоре најдесно што значи единствена можна насока е **LEFT**

Овој услов искодиран изгледа вака:

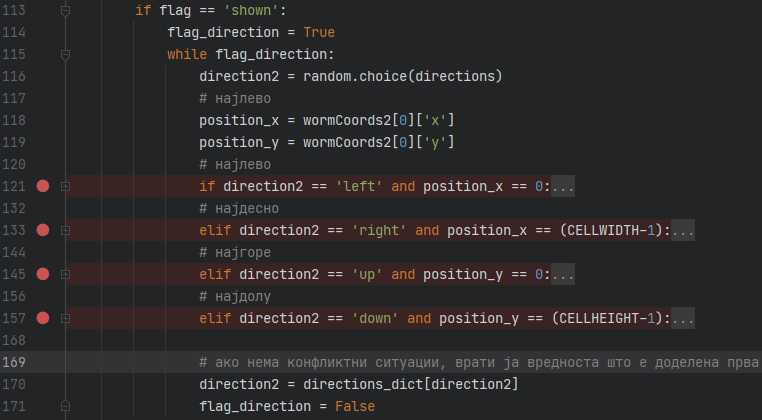


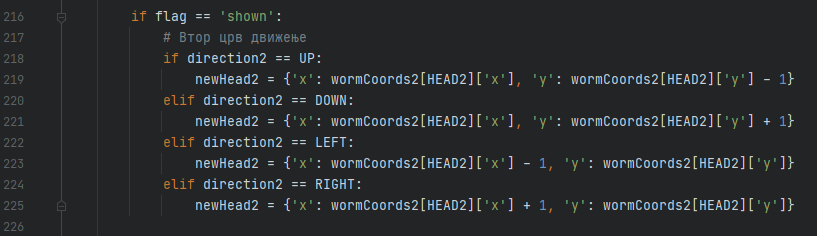
**4 УСЛОВ. 1 Сценарио:** Доколку сме најгоре и најлево даваме RIGHT како следна насока и го прекинуваме циклусот.

**4 УСЛОВ. 2 Сценарио:** Доколку сме во најгоре во средина правиме листа од двете можни насоки, случајно бираме и повторно прекинуваме (RIGHT или LEFT)

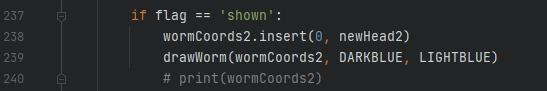
**4 УСЛОВ. 3 Сценарио:** Доколку сме најгоре и најдесно даваме насока LEFT како следна и го прекинуваме циклусот.

Со ова ги завршуваме сите можни 12 сценарија и доколку не влезе ни во еден од условите и не го прекине циклусот завршува со доделување на иницијалната насока од случајниот избор и потоа го прекинува циклусот (ред 170 – 171)



Црвот потоа се движи идентично како првиот црв со следниот блок комнади подоле во кодот:

И доколку е прикажан црвот се додава во листата со insert методот на почетокот новата глава доколку не се избрише сегмент (тоа го објаснувам подлоу)



Поголемиот дел од дебагирањето за дали е океј потегот го правев со принтање на листата од речници за црвот (ред 240).

## Колизија на црвите

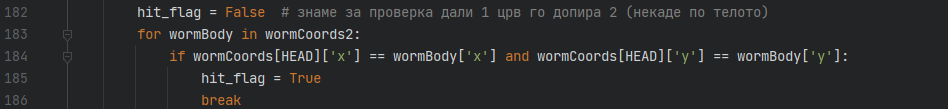
Следно за имплементација е што се случува доколку црвите имаат колизија некаква. Имаме од самиот опис на барањето 2 типа:

* + **Тип 1 колизија:** Главата од прв црв го допира втор црв (расте сегмент)
  + **Тип 2 колизија:** Втор црв го допира првиот црв со било кој сегмент било каде по телото на првиот (расте сегмент)

Притоа многу важно доколку имаме колизија од 1 тип, црвот расте но SCORE останува **непроменет**. Затоа подоцна имаме бројач кој брои дали имаме колизија од тип 1 за да минусира.

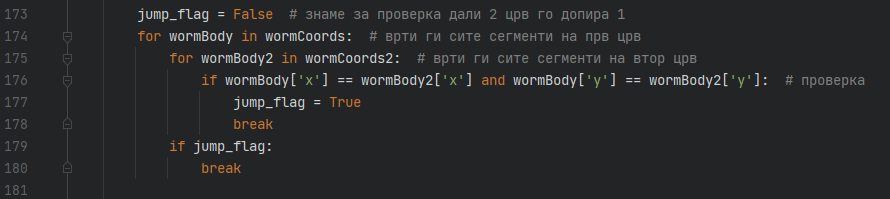
### Тип 1 колизија

Ова е доста едноставна проверка гледаме дали со својата глава привот црв го допира вториот некаде, доколку да поставуваме знамеце **hit\_flag = False**. Кога ќе првиот го допира со својата глава вториот тогаш знамето е **True**.

Искодирано ова изгледа вака:

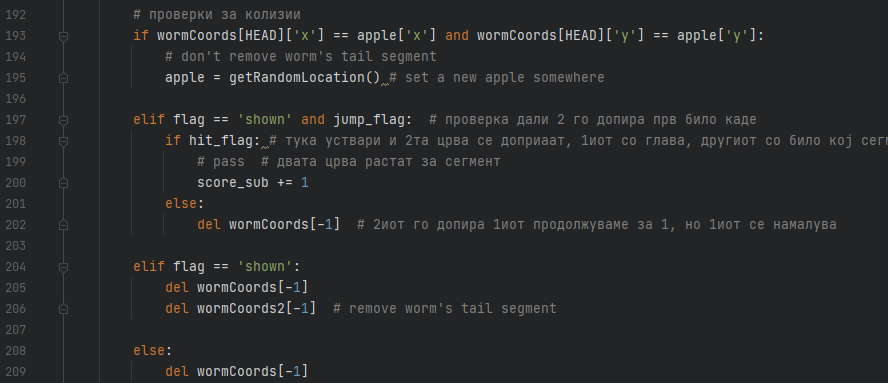
### Тип 2 колизија

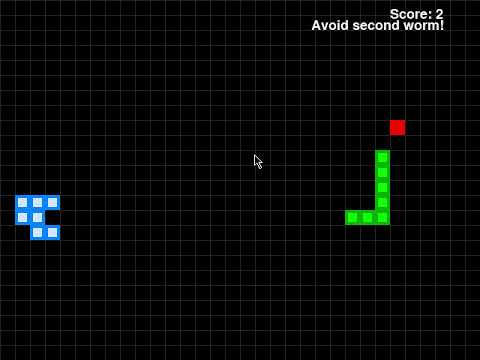
Оваа ја имав замислено како матрица каде гледаме дали секој сегмент од вториот црв е еднаков на секој сегмент од првиот црв (матрица на соседство ако би се проверувало кај графови). Ако видиме дека е еднаков тогаш ново знамеце jump\_flag од False го поставуваме на True.

Искодирано ова изгледа вака:

Сега зошто ни се потребни знамецата е повеќе кај проверките за бришење / оставање на сегменти. Имаме неколку можни ситуации:

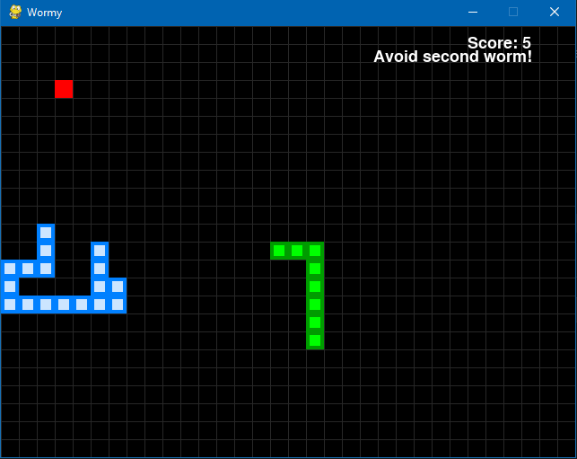
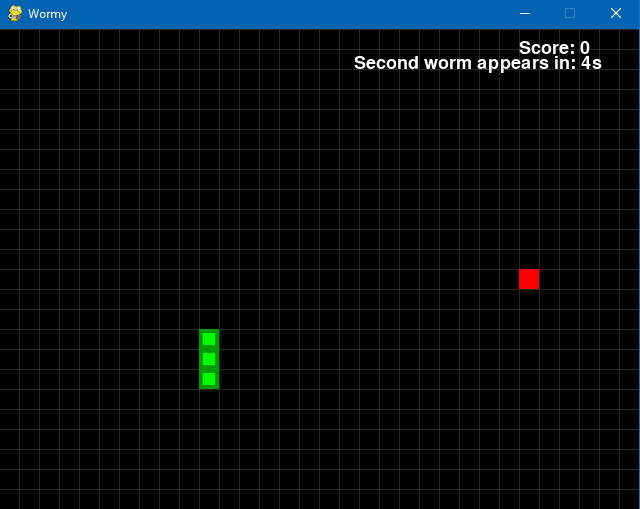
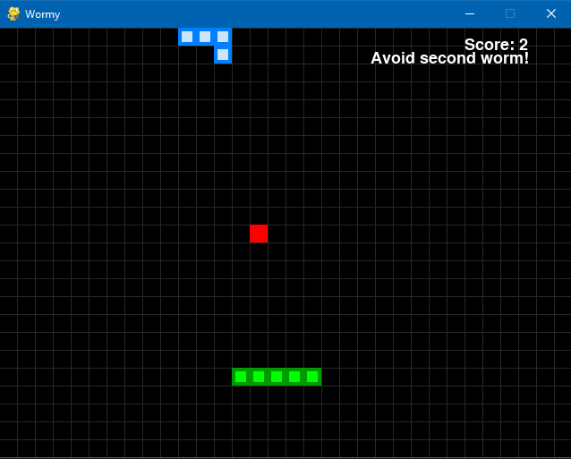
* Прв црв до допира втор црв со глава со што автоматски втроиот црв го допира првиот:   
  двата црва растат сегмент (т.е. не им се брише сегмент од крајот)
* Втор црв го допира првиот црв со своите други сегменти:   
  вториот расте сегмент (т.е. на првиот се намалува сегмент од крај)
* Ако е прикажан вториот и нема колизии:   
  ни еден не расте сегмент(т.е. и двата се намалуваат за сегмент)

Ова искодирано изгледа вака (**score\_sub:** се зголемува кога првиот со главата го допира вториот и притоа го намалува резултатот за толку со цел да ја долови суштината, плавиот расте зелениот расте но резултатот е ист):

[](https://www.youtube.com/embed/Z6ZnSMSbjZM?feature=oembed)Со цел визуелно да ви покажам како работи кодот, снимив видео и го прикачив на YouTube.   
Линкот до видеото е следен:

<https://youtu.be/Z6ZnSMSbjZM>

**NOTE:** Намалив тајмерот на видеото да почнува од 5 секунди место од 45 со цел да не прикачувам големо видео. Во крајниот код прикачувам со 45 секунди. Истотака на крајот од видеото по game over, повторно почнувам со цел да покажам дека секој пат се ресетира и големината на црвот и тајмерот за истиот.

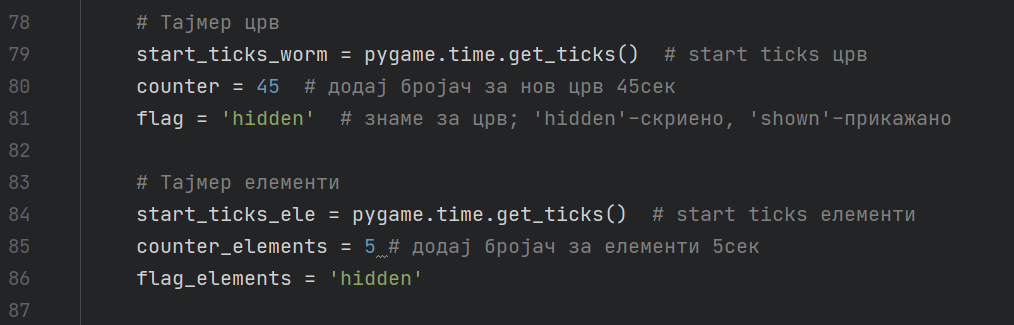
Неколку screenshots од ова барање се во прилог:

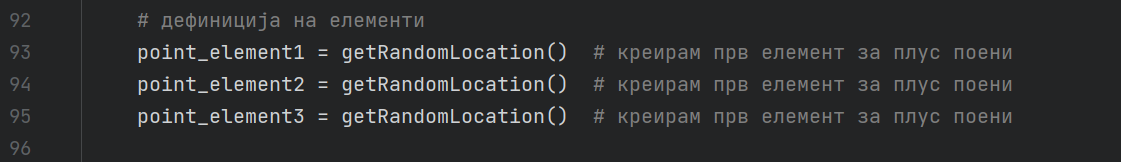
1. ***Барање 2***

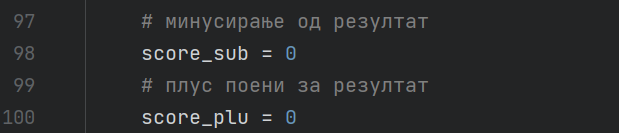
Додадете елементи што трепкаат (секој пат по три) на случајно избрани позиции со димензија од едно квадратче. Тие се појавуваат во времетраење од 5 секунди. Ако оригиналниот црв ги изеде, играчот добива дополнителни поени (по 2 за секој изеден елемент). Овие поени треба да се вклучат во конечниот резултат на начин што вие ќе го изберете. Треба да обезбедите објаснување за формулата што ја користите за пресметување на резултатот. Резултатот треба да се прикаже на екранот што се појавува кога играта ќе заврши.

Ова барање е слично (барем јас така го имплементирав) како жолтото јаболко и плавото јаболко само што наместо едно ќе имаме повеќе.

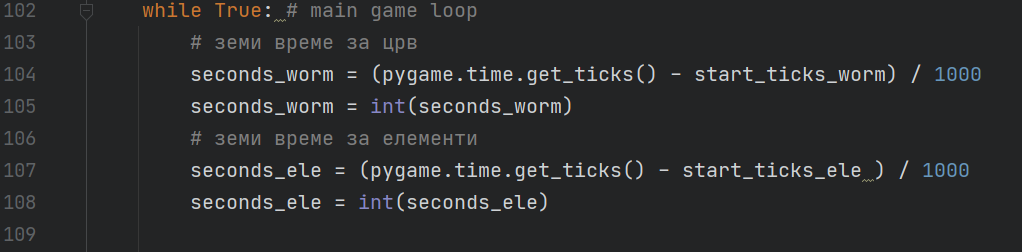
Идејата е да се има посебни бројачи, почетни времиња за двата тајмери. Секој пат кога ќе се изеде елемент се генерира нов и ова иде со трепкање од 5 секунди (5 покажува, 5 не покажува).

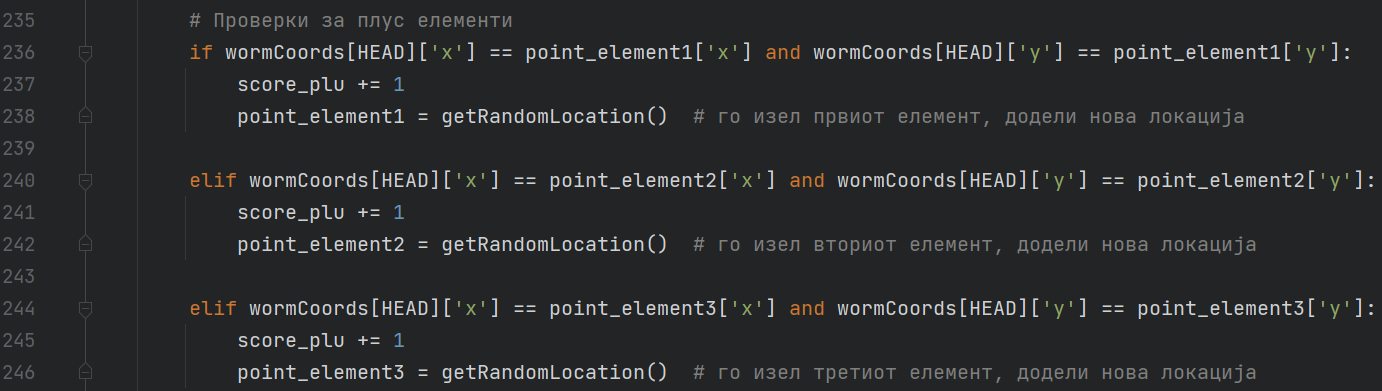
Земи почетно време за елементи, намести бројач и постави знаменце:

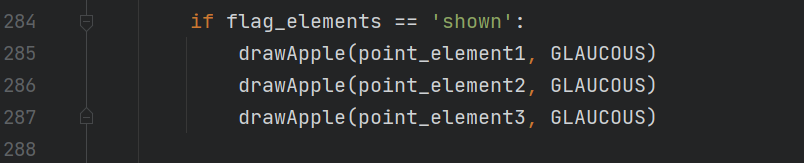
Земи три рандом променливи со рандом локации (технички треба да се проверува дали се покплопуваат две рандом локации но ја пуштив играта повеќе пати и немав 2 елемента на иста локација, ако има плус 4 поени за среќа):

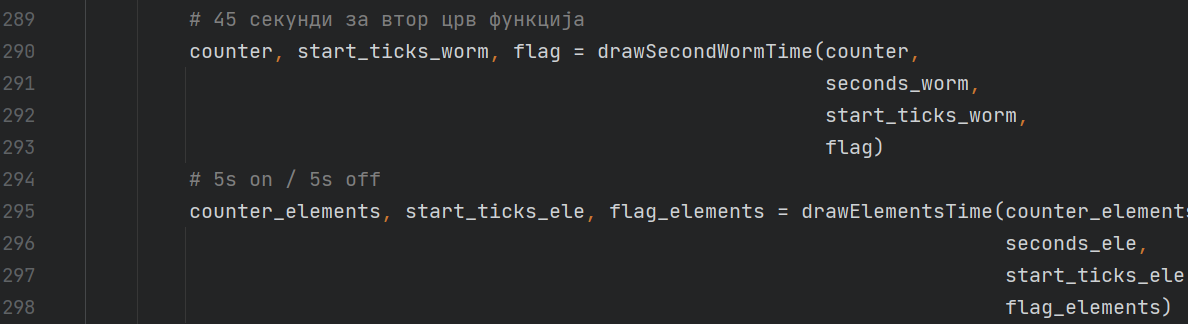
Дефинирај променливи за резултат:

* **Score\_sub:** брои колку пати сме се удриле со плав црв за да намали
* **Score\_plu:** брои колку плави елементи кои носат 2 поени сме изеле

Сега на самиот почеток од главниот циклус, ги зимаме времињата за тајмерот да работи. Притоа треба да се посебни бидејќи ако имаме start\_ticks за двата исто тогаш едниот од бројачите најчесто тој што прв истекол ќе го ресетира другиот.

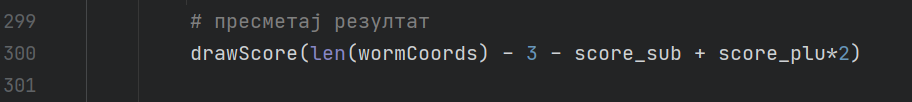
Потоа имаме проверка дали сме изеле едно од јаболките, ако сме изеле зголемуваме score\_plu += 1:

Потоа имаме проверка дали истекол тајмерот, ако да ги покажува ако не ги скока.

Ова е доста слично како со знамето на првиот црв, разликата е само оваа функција:

Разликата е внатре во самата drawElementsTime() e тоа што сакаме периодично прикажување а во drawSecondWormTime() чекаме 45 и потоа цело време го имаме.

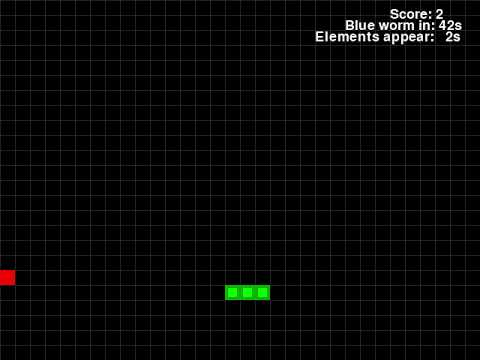
Формулата која е користам за пресметување на резултатот е следна:



Каде:

* **len(wormCoords):** Вкупна должина на црв
* **-3:** ова 3ка се почетните 3 сегменти од почетокот на играта
* **score\_sub:** ова е бројач колку сегменти се додадени каде играчот до удира плавиот црв
* **score\_plu\*2:** го множиме бројот на изедени елементи \* 2 за да го зголемуваме резултот за 2 на секој изеден елемент

На следниот линк има видео каде барање 1 и барање 2 ги испробав:

[](https://www.youtube.com/embed/4AVPHMAo6qE?feature=oembed)

<https://www.youtube.com/watch?v=4AVPHMAo6qE>

1. ***Барање 3***

На екранот што се појавува кога играта ќе заврши треба да се додадат две копчиња, "Start from the beginning" и "Quit". Кога играчот ќе кликне на првото копче, играта треба да почне од почеток (без да се појави почетниот екран). Кога играчот ќе кликне на второто копче, треба да се исклучи играта.

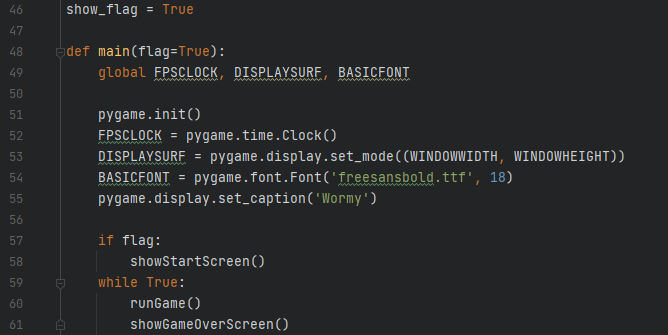
Копчињата ги додавам во главната игра во функцијата каде се испишува Game Over.

Слично како game и over, додавам surface и rect објекти и потоа проверувам слично како во Slide Puzzle дали имаме rect.colidepoint(event.pos) доколку имаме извршуваме што и да имаме во условот.

Модифицираната функција:

Во ред 509: ја повикувам главната функција но со flag. Ова знамеце игра улога во дали текстот што се врти ќе се испише или не.

Во ред 510: ја терминираме играта ако играчот кликне “Quit”

Знаменецто во главната функција го имплементирам вака:

Секој пат кога ќе биде кликнато “Start from beginning” ќе се смени вредноста и ќе го скока прикажувањето на текстот.



Како и за другите барање снимив видео демонстрација и за ова (иако е поедноставно):

<https://youtu.be/-sYOPGjvNxY>