**ЛАБОРАТОРИСКА ВЕЖБА 6**

10.01.2021

**ПРОГРАМИРАЊЕ НА**

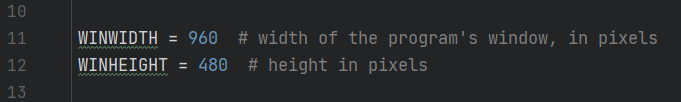
**ВИДЕО ИГРИ И СПЕЦИЈАЛНИ ЕФЕКТИ**

*Squirrel eats squirrels*

Професор*:* д-р Катарина Тројачанец Динева

Студент*:* Кирил Зеленковски

1. ***Барање 1***

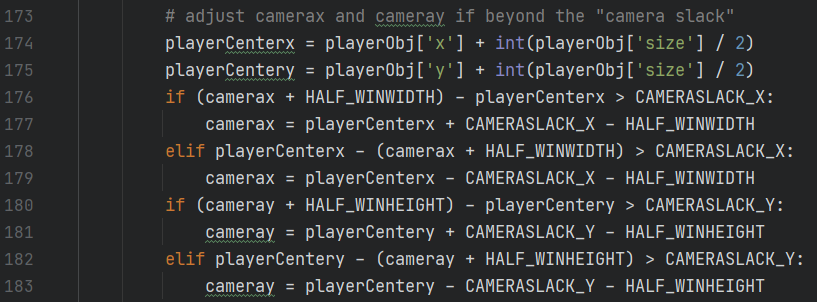
 Increase the camera area of the game to 960x480 pixels (1 point)

1. ***Барање 2***

Define two new constants for the CAMERASLACK constants for the horizontal and vertical direction appropriately (1 point)

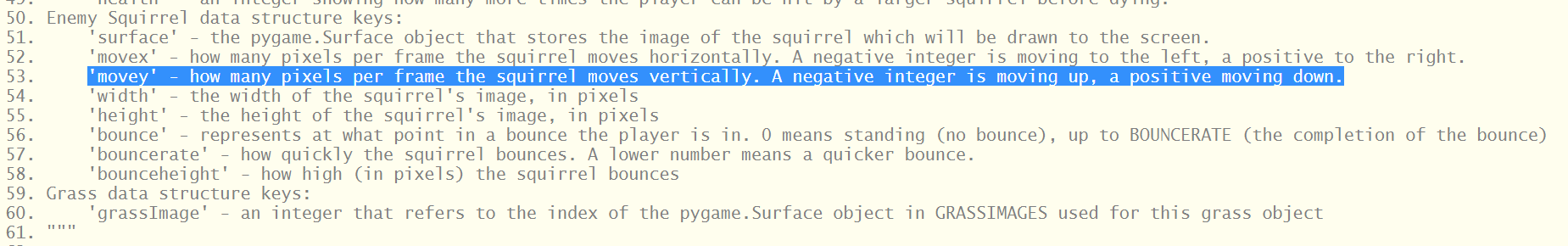
Ова барање се сведуваше на позицијата на камерата (што се чува како camerax и cameray). Ова треба да се апдејтира кога играчот оди преку.

Тука е всушност концептот на **CAMERASLACK** променливата: идејата зад оваа константа е да го наполно опредлува бројот на пиксели што играч може да се движи додека да се апдејтира. Во ред 19 (од оргиниалнот код) се поставува како 90, што значи дека нашата програма е таква така што играчот верверичка може да се движи 90 пиксели од центарот пред позицијата на камерата да се апдејтира. Потоа има равенки кои не се нужни за ова барање.

Поставуваме две променливи **CAMERASLACK\_X** и **CAMERASLACK\_Y**. Двете ги поставуваме различни и подоле во кодот каде се менува X користиме **CAMERASLACK\_X** а пак каде се менува и проверува y користиме **CAMERASLACK\_Y** како константа во условите.

1. ***Барање 3***

Introduce the possibility an enemy squirrel to bounce downwards (the squirrel could bounce only in one direction only upwards or only downwards). The direction is determined when creating an enemy squirrel.

Верверичките непријатели се составени од следните променливи:

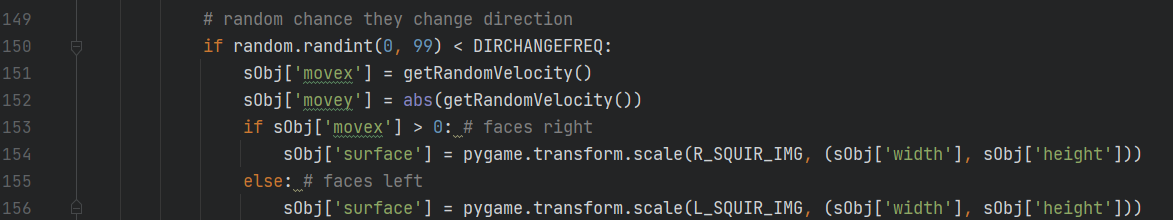
Секоја верверичка е дефинирана како речник со овие клучеви. Клучеви кои се поврзани со движење се:

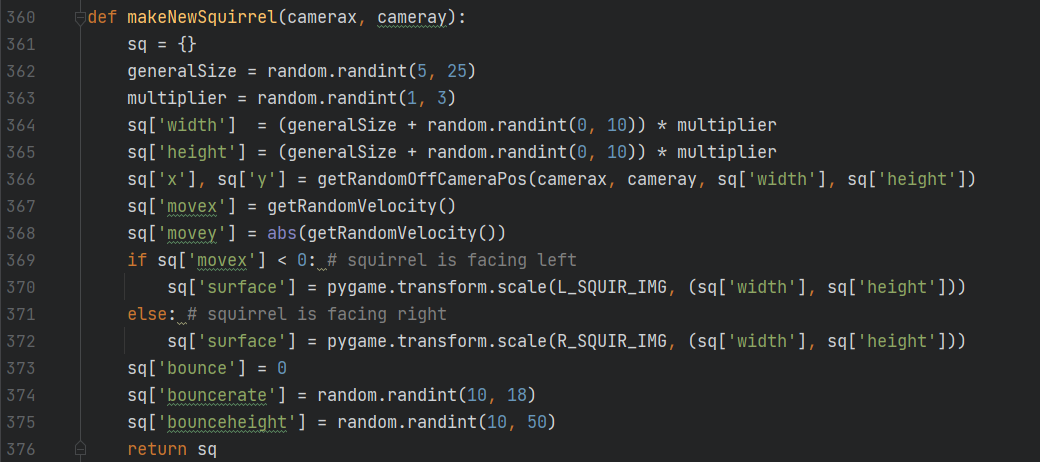
* **movex:** која ја движи верверичката **лево** или **десно** (за + движи десно, за - движи лево)
* **movey:** која ја движи верверичката **горе** или **доле** (за + движи доле, за - движи горе)

Доколку сакаме секогаш непријателските верверичките да се движат надолу (downwards) нужно е секој пат кога ја пресметуваме или доделуваме movey при преметки да ја правиме позитивна, т.е. да ја зимаме нејзината апсолутна вредност. Оваа променлива или е позитивна или е негативна, за позитивни бројки (>0) ја движи **надолу** верверичката а за вредности негативни (<0) ја движи **нагоре**.

Друго решение за овој проблем може да е и дигање на степен па зимање корен, но ова е доста полесно за имплементација во Python. Ја користиме вградената функција во интерпетерот abs().

Главното менување на вредностите за **movex** и **movey** се враќа како вредност од функцијата **getRandomVelocity()**. Оваа функција зима рандом целобројен број во одреден опсег и го враќа.

Имаме две појави каде се зимаат вредности за движењеата по y оска на случаен начин (со веројатност од 50% горе/доле -rand.int(0,1), првото е откако се креираат верверичките во главниот циклус:

А другата појава на вакво случајно доделување на променливи се случува кога се креираат новите верверички и промената ја правиме во ред 368, каде зимаме само апсолутна вредност од брзината која е со 50% горе или доле ние ја правиме да биде 100% позитивна – надолу.

Бидејќи со слика не се гледаше добро дали секогаш надолу се движат, снимив кратко видео за да го покажам барањето достапно на следниот линк:

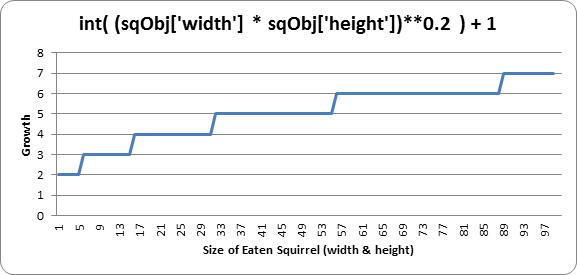
[](https://www.youtube.com/embed/KBfSwlGUTxk?feature=oembed)

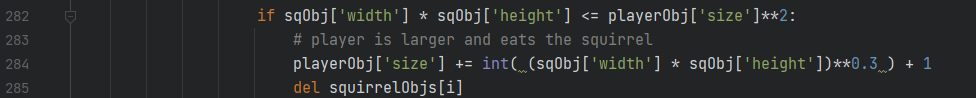
<https://youtu.be/KBfSwlGUTxk>

1. ***Барање 4***

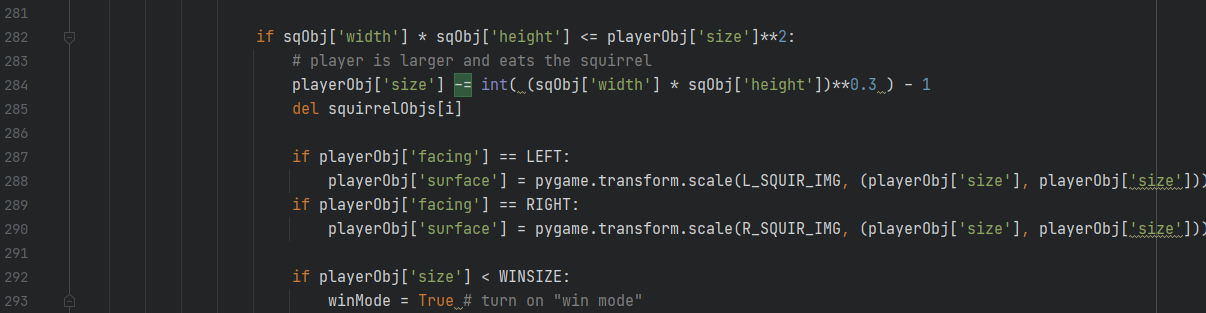
Change the logic of the game. If the squirrel is hit by the larger animal it becomes smaller using the same logic as for the getting bigger at the current game. The game is over when the squirrel grows to the WINSIZE or LOSTSIZE (5 points)

Логиката зад играта моментално е колку повеќе верверички јаде за одреден фактор се зголемува големината на верверичката во пиксели, доколку оваа големина ја достигне онаа зададена од самата игра (по default е 200) тогаш верверичката станува Omega и играта завршува.

Овој фактор за кој верверичката се зголемува се пресметува со помош на формула, која во оргиналната сркипта се наоѓа во ред 280 и идејата може да се опише со следниот график (извор: <http://inventwithpython.com/pygame/chapter8.html>):

Доколку играчот е еднаков или поголем од непријателкса верверичка и се судира со истата, тогаш играчот (верверичката) јас јаде истата и расте. Бројот за растење се доделува како вредност на клучот ‘size’ од речникот на главната верверичка (овој број е растот за верверичката) и се пресметува со помош на линија 280:

Графикот горе покажува како верверичка расте од различно изедени верверички (со разни големини) и според него ако пр главниот играч јаде верверичка со ширина и висина од 45 тогаш ќе порасне 5 пиксели во ширина и висина.

Сега, од тука ако овој фактор се “reverse engineer” – не, можеме да заклучиме дека доколку ја промениме оваа формула да одзима за одреден број (да ја смалува верверичката) тогаш ќе резултираме со обратно сценарио. За таа цел ја менуваме оваа линија код и ја менуваме логика за крај на игра (доколку е ПОМАЛО наместо поголемо од WINSIZE = 20):

Со цел визуелно да го претставам решението за барањето го снимив следното видео:

[](https://www.youtube.com/embed/9hSp-H90Bcw?feature=oembed)

<https://youtu.be/9hSp-H90Bcw>