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)

```
10
11  WINWIDTH = 960  # width of the program's window, in pixels
12  WINHEIGHT = 480  # height in pixels
13
```

2. 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.

```
21  CAMERASLACK_X = 90     # how far from the center the squirrel moves before moving the camera (along x axis)
22  CAMERASLACK_Y = 80     # how far from the center the squirrel moves before moving the camera (along y axis)
```

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 у користиме **CAMERASLACK_Y** како константа во условите.

```
173         # adjust camerax and cameray if beyond the "camera slack"
174         playerCenterx = playerObj['x'] + int(playerObj['size'] / 2)
175         playerCentery = playerObj['y'] + int(playerObj['size'] / 2)
176         if (camerax + HALF_WINWIDTH) - playerCenterx > CAMERASLACK_X:
177             camerax = playerCenterx + CAMERASLACK_X - HALF_WINWIDTH
178         elif playerCenterx - (camerax + HALF_WINWIDTH) > CAMERASLACK_X:
179             camerax = playerCenterx - CAMERASLACK_X - HALF_WINWIDTH
180         if (cameray + HALF_WINHEIGHT) - playerCentery > CAMERASLACK_Y:
181             cameray = playerCentery + CAMERASLACK_Y - HALF_WINHEIGHT
182         elif playerCentery - (cameray + HALF_WINHEIGHT) > CAMERASLACK_Y:
183             cameray = playerCentery - CAMERASLACK_Y - HALF_WINHEIGHT
```

3. 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:

```
    'health' - an integer showing how many more times the player can be hit by a larger squirrel before dying.
Enemy Squirrel data structure keys:
    'surface' - the pygame.Surface object that stores the image of the squirrel which will be drawn to the screen.
    'movex' - how many pixels per frame the squirrel moves horizontally. A negative integer is moving to the left, a positive to the right.
    'movey' - how many pixels per frame the squirrel moves vertically. A negative integer is moving up, a positive moving down.
    'width' - the width of the squirrel's image, in pixels
    'height' - the height of the squirrel's image, in pixels
    'bounce' - represents at what point in a bounce the player is in. 0 means standing (no bounce), up to BOUNCERATE (the completion of the bounce)
    'bouncerate' - how quickly the squirrel bounces. A lower number means a quicker bounce.
    'bounceheight' - how high (in pixels) the squirrel bounces
Grass data structure keys:
    'grassImage' - an integer that refers to the index of the pygame.Surface object in GRASSIMAGES used for this grass object
"""
```

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 у оска на случаен начин (со веројатност од 50% горе/доле -rand.int(0,1), првото е откако се креираат верверичките во главниот циклус:

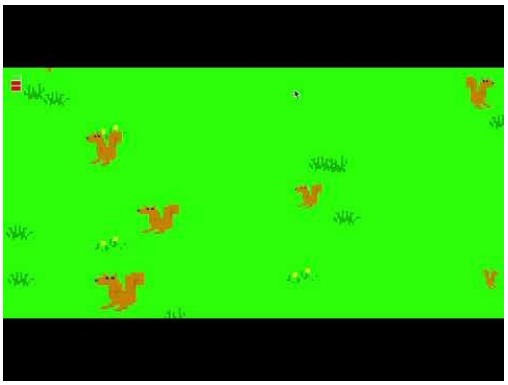
```
149             # random chance they change direction
150             if random.randint(0, 99) < DIRCHANGEFREQ:
151                 sObj['movex'] = getRandomVelocity()
152                 sObj['movey'] = abs(getRandomVelocity())
153                 if sObj['movex'] > 0: # faces right
154                     sObj['surface'] = pygame.transform.scale(R_SQUIR_IMG, (sObj['width'], sObj['height']))
155                 else: # faces left
156                     sObj['surface'] = pygame.transform.scale(L_SQUIR_IMG, (sObj['width'], sObj['height']))
```

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

```
360  def makeNewSquirrel(camerax, cameray):
361      sq = {}
362      generalSize = random.randint(5, 25)
363      multiplier = random.randint(1, 3)
364      sq['width']  = (generalSize + random.randint(0, 10)) * multiplier
365      sq['height'] = (generalSize + random.randint(0, 10)) * multiplier
366      sq['x'], sq['y'] = getRandomOffCameraPos(camerax, cameray, sq['width'], sq['height'])
367      sq['movex'] = getRandomVelocity()
368      sq['movey'] = abs(getRandomVelocity())
369      if sq['movex'] < 0: # squirrel is facing left
370          sq['surface'] = pygame.transform.scale(L_SQUIR_IMG, (sq['width'], sq['height']))
371      else: # squirrel is facing right
372          sq['surface'] = pygame.transform.scale(R_SQUIR_IMG, (sq['width'], sq['height']))
373      sq['bounce'] = 0
374      sq['bouncerate'] = random.randint(10, 18)
375      sq['bounceheight'] = random.randint(10, 50)
376      return sq
```

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://youtu.be/KBfSwlGUTxk

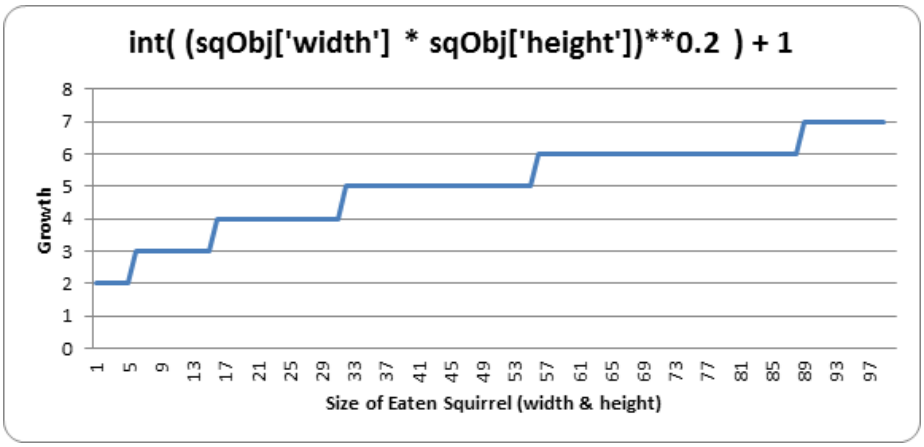


4. Барање 4

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:

```
282                 if sqObj['width'] * sqObj['height'] <= playerObj['size']**2:
283                     # player is larger and eats the squirrel
284                     playerObj['size'] += int( (sqObj['width'] * sqObj['height'])**0.3 ) + 1
285                     del squirrelObjs[i]
```

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):

```
281
282                 if sqObj['width'] * sqObj['height'] <= playerObj['size']**2:
283                     # player is larger and eats the squirrel
284                     playerObj['size'] -= int( (sqObj['width'] * sqObj['height'])**0.3 ) - 1
285                     del squirrelObjs[i]
286
287                     if playerObj['facing'] == LEFT:
288                         playerObj['surface'] = pygame.transform.scale(L_SQUIR_IMG, (playerObj['size'], playerObj['size']))
289                     if playerObj['facing'] == RIGHT:
290                         playerObj['surface'] = pygame.transform.scale(R_SQUIR_IMG, (playerObj['size'], playerObj['size']))
291
292                     if playerObj['size'] < WINSIZE:
293                         winMode = True # turn on "win mode"
```

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

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