Ťahák 3

Hra a svety

Aby naša hra dobre fungovala, tak okrem objektov potrebujeme aj nejaký **svet**, alebo viaceré svety, do ktorých tie objekty dáme. Napríklad akvárium v našej minulej hre bolo svet. Svet má svoje metódy `krok()`, `nastav()` a `nakresli()`, ktoré fungujú rovnako ako u objektov. Okrem toho má aj vlastnosť `cas`, ktorú objekty nemali, a v ktorej je uložený počet milisekúnd od začatia tohoto sveta.

Taký svet ale nemusí byť v našej hre len jeden. Môžeme do našej hry pridať nový svet, ktorý sa nám zobrazí počas pauzy, alebo taký, čo sa nám zobrazí keď prehráme. O to, aby sa svety správne otvárali, zatvárali a pauzovali sa stará `hra` (a tým myslíme premennú programu, ktorá volá napríklad funkcie `start()` a `koniec()`). Hra má svety, ktoré sme otvorili, uložené v zásobníku. To znamená, že keď zatvoríme posledný otvorený svet, tak sa nám nezobrazí prvý otvorený svet, ale posledný otvorený svet. Môžeme si to predstaviť tak, akoby ich jednoducho ukladal na seba, pričom tie, čo nie sú na vrchu sú pauznuté.

- `hra.start(svet)` – Spustí svet, ktorý mu zadáme a postará sa o to, aby hra začala – aby sa nám otvorilo nové okno a spustilo sa všetko čo sa má.
- `hra.otvorSvet(svet)` – Spustí svet, ktorý mu zadáme. Tento príkaz použijeme vtedy, keď chceme otvoriť nový svet, pričom už nejaký otvorený máme. Toto môžeme využiť na pauznutie hry – otovríme svet, v ktorom iba vypíšeme slovo „Pauza“.
- `hra.zatvorSvet()` – Zatvorí posledný otvorený svet, v prípade ak bol tento svet posledný, tak aj ukončí program. Napríklad ak sa chceme vrátiť z pauzy späť do hry.
- `hra.nahradSvet(svet)` – Nahradí posledný otvorený svet svetom, ktorý mu zadáme, toto je veľmi praktické najmä, vtedy, keď chceme zatvoriť posledný svet a po ňom otvoriť ďalší tak, aby sa nám nezatvorila hra. Toto vieme použiť pri prehraní hry. Nahradí svet hry svetom, kde je vypísané „Koniec hry“.
- `hra.koniec()` – Ukončí program.

Udalosti

Občas by sa nám zišlo, aby sme mohli v rámci programu „zakričať“ všetkým objektom, že sa niečo stalo a oni by sa už podľa toho rozhodli, čo majú robiť. Práve to môžeme spraviť pomocou udalostí. S jednou špeciálnou udalosťou ste sa už dokonca stretli, s udalosťou `priZrazke`. Táto je špaciálna tým, že ste ju nevytvárali vy, ale o jej správne volanie sa starala sama hra.

Ostatné udalosti môžete vytvárať pomocou volania metódy, ktorú má svet: `svet.nastalaUdalost(„udalost")`. String „udalost“ je názov udalosti, ktorá sa má stať. Ak nechceme, aby udalosť nastala hneď, ale miesto toho chceme, aby sa vykonala až neskôr, napríklad si chceme nastaviť budík, tak to môžme spraviť pomocou príkazu `nacasujUdalost(milisekundy,„udalost")`, kde milisekundy je počet milisekúnd, za ktoré sa má udalosť stať. Program však nečaká kým sa udalosť stane, normálne pokračuje vo vykonávaní a až po *milisekundy* milisekundách vytvorí príslušnú udalosť.

Na udalosti potom môžu objekty alebo svet reagovať pomocou `@priUdalosti(„udalost")`. Tento príkaz funguje tak, že ho dáme tesne nad funkciu, ktorá sa má, v prípade, že nastane daná udalosť, vykonať. Rovnako funguje aj `@priZrazke(trieda)`, až na to, že sa funkcia pod tým nevykoná vtedy keď nastala nejaká nami vytvorená udalosť, ale vtedy, keď sa objekt, ku ktorému to patrí zrazí s objektom, ktorý patrí do triedy, ktorú sme mu zadali.

Keby sme napriklad pri spustení hry nastavili budík na 3 sekundy a naša postava, by sa naňho mala zobudiť tak by to mohlo vyzerať približne takto:

```
class Postava(Vec):
    def nastav(self):
        self.zobudeny = False

    @priUdalosti("ZvoniBudik")
    def zobudzamSa(self):
        self.zobudeny = True

class MojSvet(Svet):
    def nastav(self):
        self.nacasujUdalost(3000,"ZvoniBudik")
```

Ešte jedna špeciálna udalosť, ktorú môžeme použiť sa volá „KLAVES DOLE“. Táto udalosť sa volá vtedy, keď sa stlačí nejaký kláves a posúva nám aj dva parametre – prvý z nich nám dá kláves vo formáte `pygame.K_nazovKlavesy` (tak ako ste ich používali v akváriu) a druhý sa používa vtedy, keď sú to nejaké rozumné znaky (napr. písmená) a je to len jeden znak reprezentujúci daný kláves (napr. ak sa stlačilo A, tak `klaves==pygame.K_a` a `unicode=='a'`).

Keby sme napísali kód tak, ako tesne pod týmto odstavcom, tak by sme vo funkcii klaves mohli pristupovat k prvému parametru pomocou premennej *klaves* a k druhému pomocou premennej *pismeno*.

```
@priUdalosti("KLAVES_DOLE")
def klaves(self, klaves, pismeno):
    pass
```

Zoznamy

Niekedy bude pre nás praktické mať zoznam nejakých objektov alebo premenných, napríklad, zoznam vecí v našom inventáry. Ak chceme vytvoriť zoznam s nejakými konkrétnymi objektami, tak ho môžeme vytvoriť tak, že do hranatých zátvoriek dáme čiarkami oddelené jednotlivé objekty, napr: `A = [objekt1, objekt2, objekt3]`, prázdny zoznam môžeme vytvoriť takto: `A = []`.

K jednotlivým objektom potom môžeme pristupovať podľa ich poradia v tomto zozname, napríklad, ak by sme chceli tretí prvok zoznamu A priradiť do premennej *treti*, tak to môžme spraviť takto: `treti = A[2]`. Naozaj je to $A[2]$ a nie $A[3]$, lebo prvky sú číslované od 0.

Do zoznamu môžeme na jeho koniec pridávať ďalšie objekty pomocou metódy `zoznam.append(objekt)` a naopak, vymazať nejaký prvok môžeme pomocou metódy `zoznam.remove(objekt)`.

```
A = ["moj","novy","zoznam"]
print A[1] #vypise novy
A.append("nieco")
print A[3] #vypise nieco
A.remove("novy")
print A[1] #vypise zoznam
```

Veľakrat máme zoznam vecí kvôli tomu, aby sme s každým jeho prvkom niečo spravili. Na prechádzanie zoznamu slúži `for` a `in`. Nasledujúca časť programu sčíta všetky čísla v zozname:

```
sucet = 0
cis = [1, 5, 8, -4]
for cis in cisla:
    sucet += cis
print sucet # vypise 10
```

To, čo sa nachádza za `for` (v tomto prípade *cis*), je názov premennej, do ktorej sa nám postupne vloží každý prvok tohto poľa a vykoná sa s ním to, čo je v cykle (v tomto prípade sa to, čo je v premennej *cis* pripočíta do premennej *sucet*)

Ešte jedna vec, ktorá sa nám bude hodiť je príkaz `break`. Ten slúži na to, aby sme mohli z `for` cyklu predčasne odísť. Predstavme si, že máme zoznam cifier nejakého čísla, ktoré ale začina nulami a my by sme chceli vedieť na akú cifru (inú ako nula) začína. V nasledujúcom príklade budeme prechádzať pole, až kým nenájdeme prvú cifru inú ako nula a vypíšeme ju.

```
for cifra in cislo:
    if cifra != 0:
        print cifra
        break
```