Programko.sk

Python 4

ZAČÍNAME S PYGAME

Pygame je knižnica, vytvorená primárne na tvorbu hier. Je veľmi podobná knižnici tkinter (z kurzu python 3) no ponúka väčšie množstvo možností a zároveň je lepšie optimalizovaná, takže výsledné programy vyžadujú menší výkon ako programy vytvorené v tkinter.

Narozdiel od tkinter, pygame sa nám nestiahne spolu s pythonom, musíme ho nainštalovať samostatne.

INŠTALÁCIA PYGAME

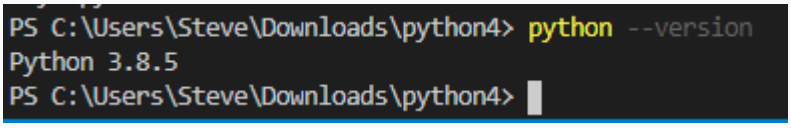
Ak máte v počítači nainštalovaní python, verziu 3.7.7 a vyššiu, inštalácia pygame je veľmi jednoduchá.

Otvoríme si powershell na windowse, bash na linuxe alebo terminal na macOS.

Pre zistenie verzie pythonu zadáme príkaz:

```
python3 --version
// prípadne, ak by prvý príkaz nefungoval
python --version
```

A mali by ste dostať podobný výstup. Ak máte verziu vyššiu ako 3.7.7, je všetko v poriadku. Inak poproste svojho lektora o pomoc pri inštalácii vyššej verzie



Ďalej spustíme príkaz, tomuto posielame niekoľko parametrov.

```
python3 -m pip install -U pygame --user
// alebo
python -m pip install -U pygame --user
```

Ako prvé zavoláme samotný python a povieme mu aby spustil pip.

Pip je package manažér, vďaka ktorému vieme do nášeho pythonu jednoducho pridávať nové knižnice, ako je napríklad pygame.

Teda **python -m pip install -U pygame** povie pythonu, aby stiahol knižnicu pygame (keďže ide o officiálnu knižnicu, vie odkiaľ má pygame stiahnuť iba podľa mena) a následne flag **--user** ktorá povie pip-u aby modul nainštaloval iba pre aktuálneho používateľa.

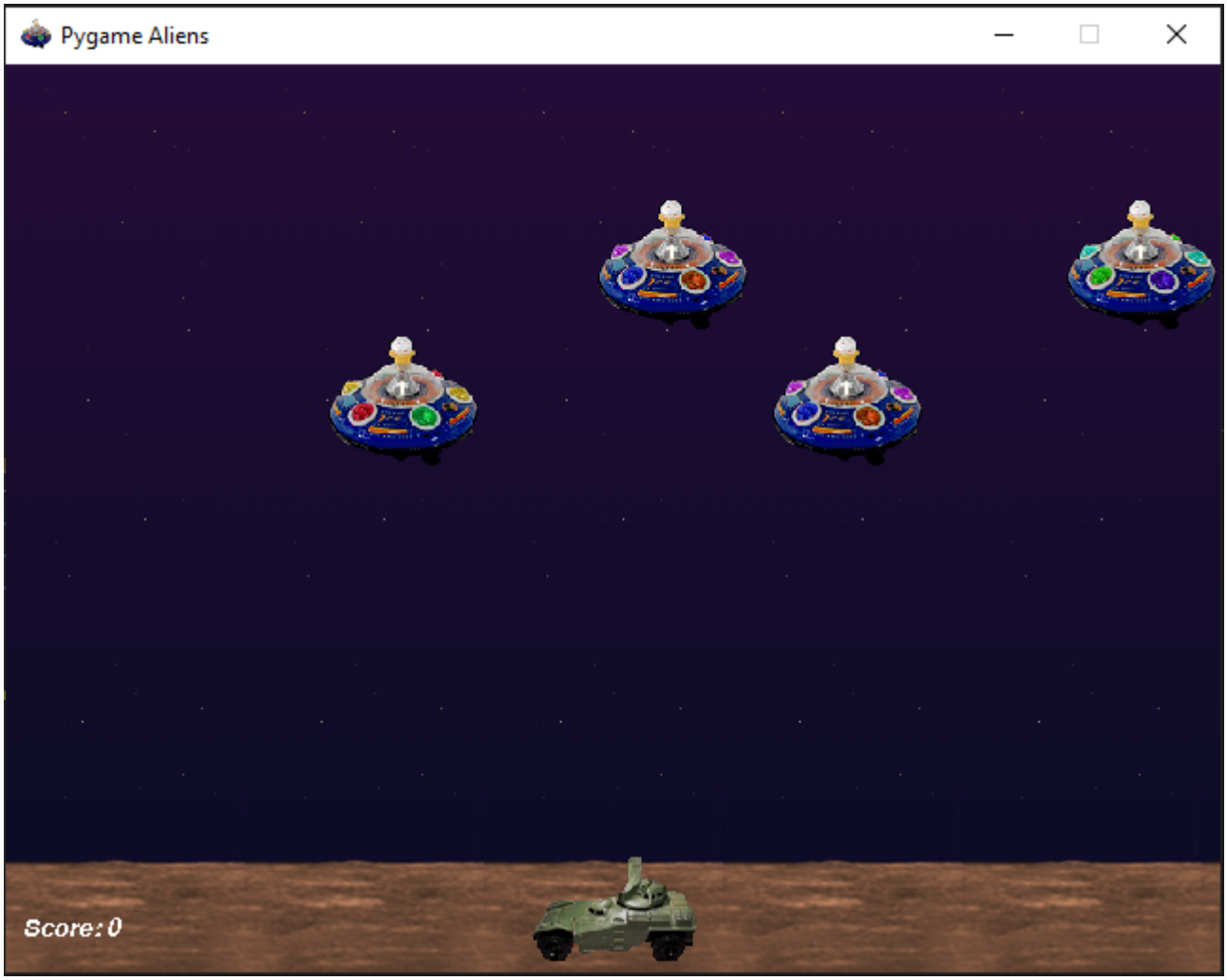
```
PS C:\Users\Steve\Downloads\python4> python -m pip install -U pygame --user
Requirement already satisfied: pygame in c:\users\steve\appdata\local\programs\python\py
Collecting pygame
  Using cached pygame-2.0.1-cp38-cp38-win32.whl (4.8 MB)
Installing collected packages: pygame
Successfully installed pygame-2.0.1
PS C:\Users\Steve\Downloads\python4>
```

Ak všetko prebehne správne, mali by ste mať na konci výstupu “Successfully installed pygame-2.x.y”.

Nakoniec nám zostáva si otestovať, či sa knižnica správne stiahla a teda funguje.

Na to nám stačí jednoduchý príkaz ktorý, v prípade že je všetko v poriadku spustí príklad hry vytvorenej v pygame

```
python3 -m pygame.examples.aliens
// alebo
python -m pygame.examples.aliens
```



Používanie pygame

Pre spustenie pygame sú najdôležitejšie dva príkazy a to:

pygame.init() # skrátené initialize, teda zapnutie knižnice

pygame.quit() # ukončenie programu

Všetok kód našej hry musí teda byť medzi týmito dvoma riadkami.

Ako prvé si teda vyskúšame či vieme knižnicu pygame importovať do nášho vlastného súboru

Vyskúšajte

```
import pygame

pygame.init()
# sem pôjde náš kód
pygame.quit()
```

V tomto programe importujeme knižnicu, zapneme ju a rovno ju vypneme. To znamená že sa nič nestane, ale len aby sme si overili že všetko funguje ako má, terminál by nám mal vypísať takýto (alebo podobný) text

```
pygame 2.0.1 (SDL 2.0.14, Python 3.8.5)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

Display - okno pygame

Aby sme mali kde našu hru vykreslovať, musíme si vytvoriť okno do ktorého budeme kresliť.

Okno alebo display nášho "inicializovaného" programu získame príkazom:

moje_okno = pygame.display.set_mode((1024, 720))

Kde parameter ktorý funkcii **set_mode** posielame je tuple (dvojica) čísel ktoré reprezentujú šírku a výšku nášho okna. Táto funkcia nám následne vracia objekt nášho okna, aby sme s ním mohli ďalej pracovať.

Vyskúšajte

```
import pygame

pygame.init()
moje_okno = pygame.display.set_mode((1024, 720))
pygame.quit()
```

Vyskúšajte si vytvoriť svoje vlastné okno, a program spustiť. Všimnite si že okno nám iba preblikne cez obrazovku, viete povedať prečo?

Všetky parametre ktoré môžeme set_mode poslať sú:

size=(0, 0), flags=0, depth=0, display=0, vsync=0

Size už funkcii posielame, depth, display a vsync. Flags môžeme poslať konštantu zadefinovanú pygame, ktorá zapne nejaké preddefinované nastavenia. Jedna zaujímavá flag je napríklad **pygame.FULLSCREEN,** ktorá zapne naše screen na celú obrazovku

Vyskúšajte

```
import pygame

pygame.init()
moje_okno = pygame.display.set_mode((1024, 720), pygame.FULLSCREEN)
pygame.quit()
```

Vyskúšajte čo sa stane ak zapneme display na celú obrazovku

Ostatné možnosti sú príliš zložité na tento kurz a nebudeme ich používať, v prípade záujmu si ich viete pozrieť na tomto odkaze:

https://www.pygame.org/docs/ref/display.html#pygame.display.set_mode

Základné game-loop

Na záver hodiny vytvoríme v našom kóde základný cyklus, ktorý udrží hru bežať až kým ju užívateľ nevypne.

```
import pygame

pygame.init()

screen = pygame.display.set_mode((576, 1024))

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break
    # kreslenie obrázkov, logika hry etc..
    pygame.display.update()

pygame.quit()
```

while True: zaručí aby hra neskončila až kým užívateľ nevypne hru sám

for event in pygame.event.get(): prechádza array aktuálne zavolaných eventov ktoré dostaneme od pygame, tieto si bližšie prejdeme neskoršie v kurze, nateraz nám stačí ak zistíme či je niektorý z eventov typu **pygame.QUIT,** vtedy vieme že máme hru zastaviť, teda nastavím running na False a vyjdeme z nášeho while cyklu.

pygame.display.update() tento príkaz povie pygame displayu, obnovil vykreslené objekty (zatiaľ nerobí nič, keďže nič nekreslíme, no budeme ho potrebovať neskôr)

Kreslenie a pohyb geometrických tvarov

Kreslenie tvarov pygame je jednoduché, a veľmi podobné knižnici tkinter.
Vyuzívame na to funkcie:

```
pygame.draw.rect(surface, color, rect)
pygame.draw.circle(surface, color, center, radius)
pygame.draw.polygon(surface, color, points)
pygame.draw.line(surface, color, start_pos, end_pos, width)
```

(Existujú aj ďalšie zložitejšie, ktoré nie sú súčasťou kurzu, ak by ste ale chceli, sú pekne rozpísané v pygame dokumentácii **https://www.pygame.org/docs/ref/draw.html)**

Všimnite si, že všetky štyri majú spoločné atribúty surface a color.

color

Je farba našeho útvaru, túto farbu môžeme reprezentovať viacerými spôsobmi, v tomto kurze budeme používať formát trojice čísel ktoré reprezentujú rgb hodnoty teda napríklad (255, 255, 255).
Skratka rgb znamená red green blue a teda každé číslo ktoré v trojici máme reprezentuje jednu z týchto farieb. Čísla môžu mať hodnoty medzi 0-255.
Napríklad (255, 0, 0) je čisto červená, pretože hovoríme že chceme maximálnu hodnotu červenej a žiadnu hodnotu zelenej a modrej.

surface

Surfaces (po slovensky povrchy) budeme prechádzať v neskoršej kapitole, ide ale v podstate o povrchy ktoré sú postupne na sebe naskladané a vieme na ne kresliť, či už útvary alebo obrázky.
Nateraz nám stačí ak budeme ako surface posielať náš screen

rect

rect je štruktúra ktorú vieme dostať od knižnice pygame a to takto:
stvorec = pygame.Rect(left, top, width, height)

Kde stačí ak pošleme je vzdialenosť od ľavého okraja, vzdialenosť od vrchu a jeho šírku a výšku, teda napríklad:

stvorec = pygame.Rect(10, 20, 50, 50)
Bude štvorec vzdialený 10 pixelov zprava, 20 pixelov z vrchu a bude široký a vysoký 50 pixelov.

Ostatné premenné sú už samostatné pre každý útvar, prejdeme si ich teda postupne.

Vyskúšajte

Využite základný kód, ktorý sme si pripravili na prvej hodine a pridajte do neho vytvorenie a vykreslenie štvorca

```
import pygame

pygame.init()

screen = pygame.display.set_mode((1024, 720))

running = True
stvorec = pygame.Rect(100, 100, 100, 100)

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break
    pygame.draw.rect(screen, (255, 0, 0), stvorec)
    pygame.display.update()

pygame.quit()
```

Výsledok by mal vyzerať nasledovne

Pohrajte sa s premennými color a Rect, a skúste si váš štvorec, presunúť, zmenšiť/zväčšiť alebo mu zmeniť farbu, aby ste si osvojili ako tieto parametre fungujú.

Vyskúšajte

Pridajte do vašeho kódu kruh, polygon (mnohouholník) a čiaru

```
import pygame

pygame.init()

screen = pygame.display.set_mode((1024, 720))

running = True
stvorec = pygame.Rect(100, 100, 100, 100)

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break
    pygame.draw.rect(screen, (255, 0, 0), stvorec)
    pygame.draw.circle(screen, (0, 255, 0), (100, 100), 20)
    # kruhu posielame screen, farbu, dvojicu čísel
    # ktoré ukazujú, kde má kruh stred a nakoniec jedno
    # číslo ktoré určuje jeho polomer

    pygame.draw.polygon(screen, (0, 0, 255), [(200, 200), (250, 200),
(320, 225), (200, 225)])
    # mnohouholníku posielame screen, farbu, list dvojíc čísel
    # ktoré určujú jeho body

    pygame.draw.line(screen, (0, 0, 255), (500, 500), (550, 550), 10)
    # mnohouholníku posielame screen, farbu, dve dvojice čísel,
    # kde prvá dvojica je jej prvý bod, druhá druhý bod
    # a nakoniec hrúbku čiary
    pygame.display.update()

pygame.quit()
```

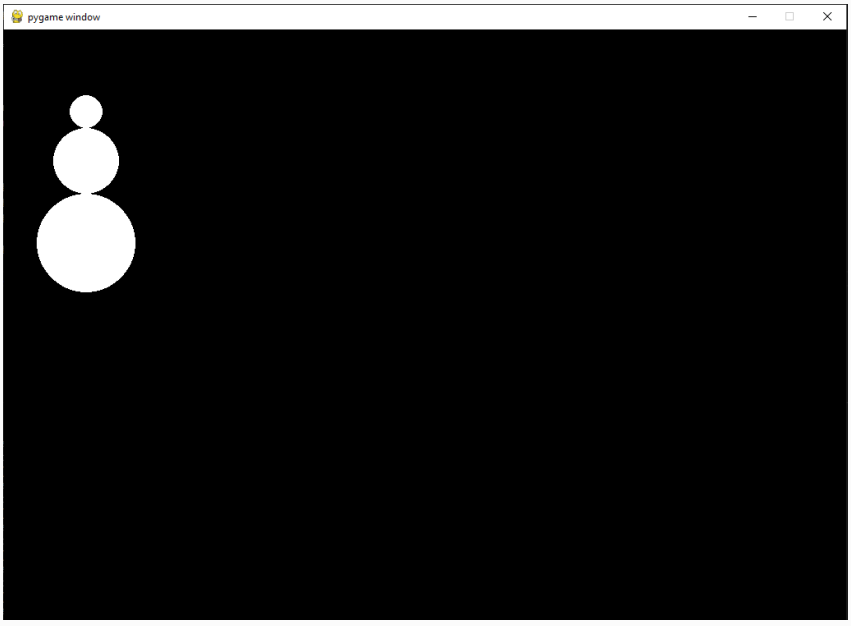
Znova sa skúste pohrať a pomeňte premenné ktoré sa funkciám posielajú, aby ste sa s nimi viac oboznámili

2.1 Úloha
Nakreslite pomocou jednoduchých útvarov, ktoré sme sa dnes naučili jednoduchý domček

Hint: Použite rect a polygon

2.2 Úloha

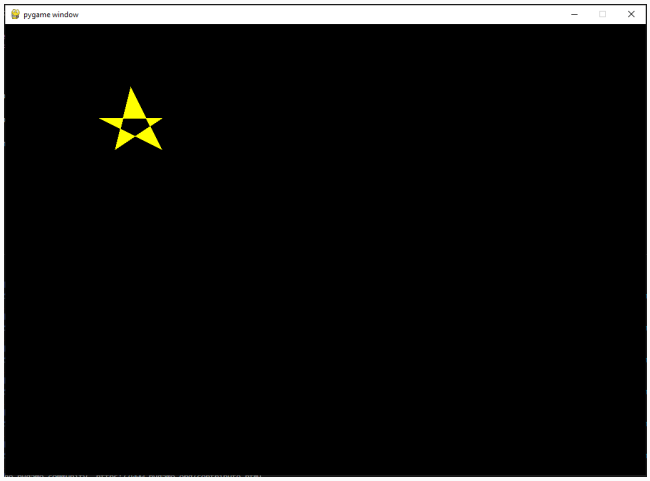
Nakreslite snehuliaka



2.3 Úloha

Nakreslite jednoduchú hviezdu pomocou polygonu

Bonus: Ak ste sa už učili o funkciách sínus a cosínus, vedeli by ste túto hviezdu spraviť presne pomocou týchto funkcií?



Kreslenie obrázkov

Kreslenie obrázkov na obrazovku je v pygame veľmi zjednodušené. V zásade nám bude treba vedieť dva príkazy. Načítanie obrázka a následne jeho vykreslenie.

obrazok = pygame.image.load('./NejakyObrazok.png')

Tento príkaz načíta obrázok s daným menom a uloží ho do premennej obrazok. Navyše môžeme zavolať ešte funckiu convert_alpha.

obrazok = pygame.image.load('./NejakyObrazok.png').convert_alpha()

Convert nespraví žiadnu viditeľnú zmenu, iba v pozadí zmení obrázok na typ, s ktorým vie pygame ľahšie pracovať a teda bude naša hra bežať rýchlejšie.

Následne vieme zavolať funkciu blit na našom screen ktoré sme si vytvorili už skôr

screen.blit(obrazok, (100, 100))

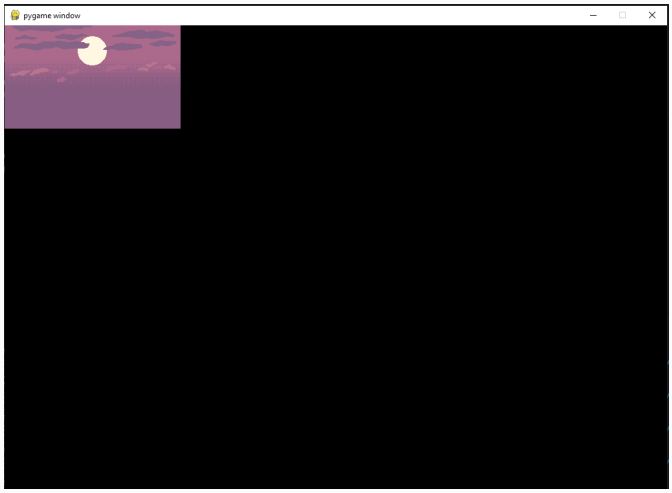
Tejto funkcii posielame náš obrázok a dvojicu čísel ktoré jej ukážu kde sa má na obrazovke obrázok vykresliť.

Vyskúšajte

```
import pygame

pygame.init()

screen = pygame.display.set_mode((1024, 720))

running = True
sky = pygame.image.load('./Sky.png').convert()

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break

    screen.blit(sky, (0, 0))
    pygame.display.update()

pygame.quit()
```

Lektor vám pošle potrebné súbory pre túto hodinu, prípade si ich viete stiahnuť z našeho verejného git repozitára
Pozor: Ak chcete načítať obrázok týmto spôsobom, musí sa nachádzať v rovnakom priečinku ako váš súbor .py

Výsledok by mal vyzerať nasledovne.



Asi ste si všimli že obrázok je je iba v rohu obrazovky. Musíme ho teda nejak upraviť aby sa roztiahol na celú šírku.

Takéto zmeny obrázka sa nazývajú transformácií. Základné transformácie ktoré budeme používať sú scale (zmena veľkosti), rotate (otáčanie) a flip (prevrátenie).

Scale

sky = pygame.transform.scale(sky, (1024, 602))

Zavoláme nasledovne a posielame mu obrázok ktorý chceme transformovať, dvojicu čísel ktoré reprezentujú výslednú veľkosť a výsledok uložíme znova do premennej

Okrem funkcie scale môžeme použiť aj funkciu scale2x, ktorá nepotrebuje výslednú veľkosť, jednoducho zväčší obrázok 2 krát.

sky = pygame.transform.scale2x(sky)

Rotate

sky = pygame.transform.rotate(sky, 90)

Funkcia rotate je ešte jednoduchšia, tej jednoducho pošleme uhol o ktorý chceme obrázok otočiť v smere hodinových ručičiek.

Flip

sky = pygame.transform.flip(sky, True, False)

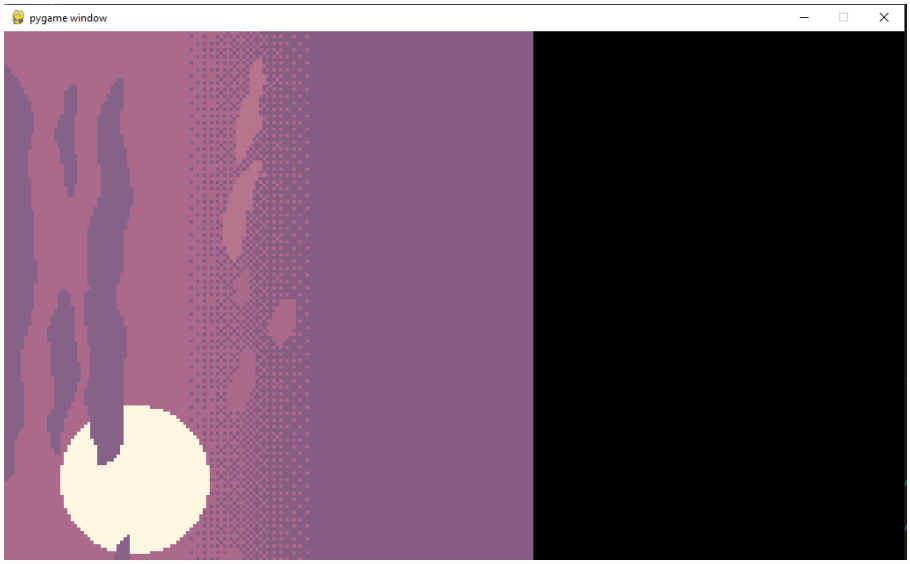
Funkcii flip zas posielame okrem obrázka dva parametre, prvý značí či chceme obrázok prevrátiť horizontálne a druhý vertikálne.

Daný príklad teda obrázok prevráti horizontálne.

Vyskúšajte

```
import pygame

pygame.init()

screen = pygame.display.set_mode((1024, 602))

running = True
sky = pygame.image.load('./Sky.png').convert()
sky = pygame.transform.scale(sky, (1024, 602))
sky = pygame.transform.rotate(sky, 90)
sky = pygame.transform.flip(sky, False, True)

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break

    screen.blit(sky, (0, 0))
    pygame.display.update()

pygame.quit()
```

Zoberte predošlú ukážku a vyskúšajte na neho aplikovať rôzne transformácie, aby ste sa utvrdili v tom ako fungujú.
(Pre tento príklad sme zmenili rozlíšenie našej aplikácie na 1024x602, aby pomer strán sedel k naškálovanému obrázky oblohy)



Clock alebo kontrola FPS (frames per second)

Veľmi dôležitou funkcionalitou pri tvorbe hier, je kontrola fps, alebo rýchlosti, s akou sa obrázky na obrazovku vykresľujú. Pygame na takúto kontrolu našťastie ponúka veľmi zjednodušenú funkcionalitu a nemusíme sa trápiť detailami.

Pointou je aby naša hra nešla ako rýchlo len môže, teda napríklad by ste sa vedeli hýbať rýchlejšie na rýchlejšom počítači, ale aby sledovala ako dlho trvá kým sa vykreslí naša hra jeden krát, a následne počkala, aby sa vykreslila ďalší krát až po stabilnom intervale.

Ak ste niekedy hrali nejaké videohry, určite aspoň zbežne viete čo fps znamená :) Je to vlastne presne to čo sa v názve po anglicky píše, tj koľkokrát sa hra vykreslí za sekundu.

Týmto dosiahneme že nezávisle od výkonu našeho počítača sa budú všetky veci hýbať rovnako rýchlo.

Vyskúšajte

```
import pygame

pygame.init()

screen = pygame.display.set_mode((1024, 602))

running = True
sky = pygame.image.load('./Sky.png').convert()
sky = pygame.transform.scale(sky, (1024, 602))
sky_x = 0

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break

    sky_x += 1
    screen.blit(sky, (sky_x, 0))
    pygame.display.update()

pygame.quit()
```

Pre ilustráciu si uložíme pozíciu našej oblohy do premennej sky_x
a v každom cykle danú premennú zväčšíme o jedna.
Všimnite si že pohyb je veľmi rýchly a v niektorých prípadoch seká,
nie je pravidelný (závisí od vašeho počítača)

Na kontrolu FPS musíme do našeho kódu pridať dva príkazy:

clock = pygame.time.Clock() # nám vráti objekt od pygame, ktorý fps za nás kontroluje

clock.tick(1) # posielame číslo, koľko frames za sekundu chceme

clock.tick() musíme volať na konci nášho while cyklu, pretože táto funkcia po každom vykreslení zráta aký čas prešiel a zastaví program na čas, ktorý treba aby boli frames per second stabilné.

Vyskúšajte

```
import pygame

pygame.init()

screen = pygame.display.set_mode((1024, 602))
clock = pygame.time.Clock()

running = True
sky = pygame.image.load('./Sky.png').convert()
sky = pygame.transform.scale(sky, (1024, 602))
sky_x = 0

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break

    sky_x += 1
    screen.blit(sky, (sky_x, 0))
    pygame.display.update()

    clock.tick(1)

pygame.quit()
```

Všimnite si že ak nastavíme fps, alebo tick() na jedna,
obrázok sa posunie iba raz za sekundu.
Navyše sa hra zle ukončuje, keďže sa naše eventy (zatvorenie) prezerajú iba raz za sekundu, tj. ak klikneme na zatvorenie hry, musíme počkať na ďalšiu sekundu, aby sa hra zatvorila.

Skúste tick() nastaviť na 60, prípadne 120, keďže to sú ideálne hodnoty,
v akých by hry mali bežať. Výsledkom bude oveľa plynulejší pohyb, ale aj okamžité zatvorenie po kliknutí X.

3.1 Úloha

Načítajte viaceré obrázky ktoré ste dostali a skúste ich vykresliť tak aby ste vytvorili scenériu. Nebojte sa využiť rôzne transformácie, či hýbať s danými obrázkami :)

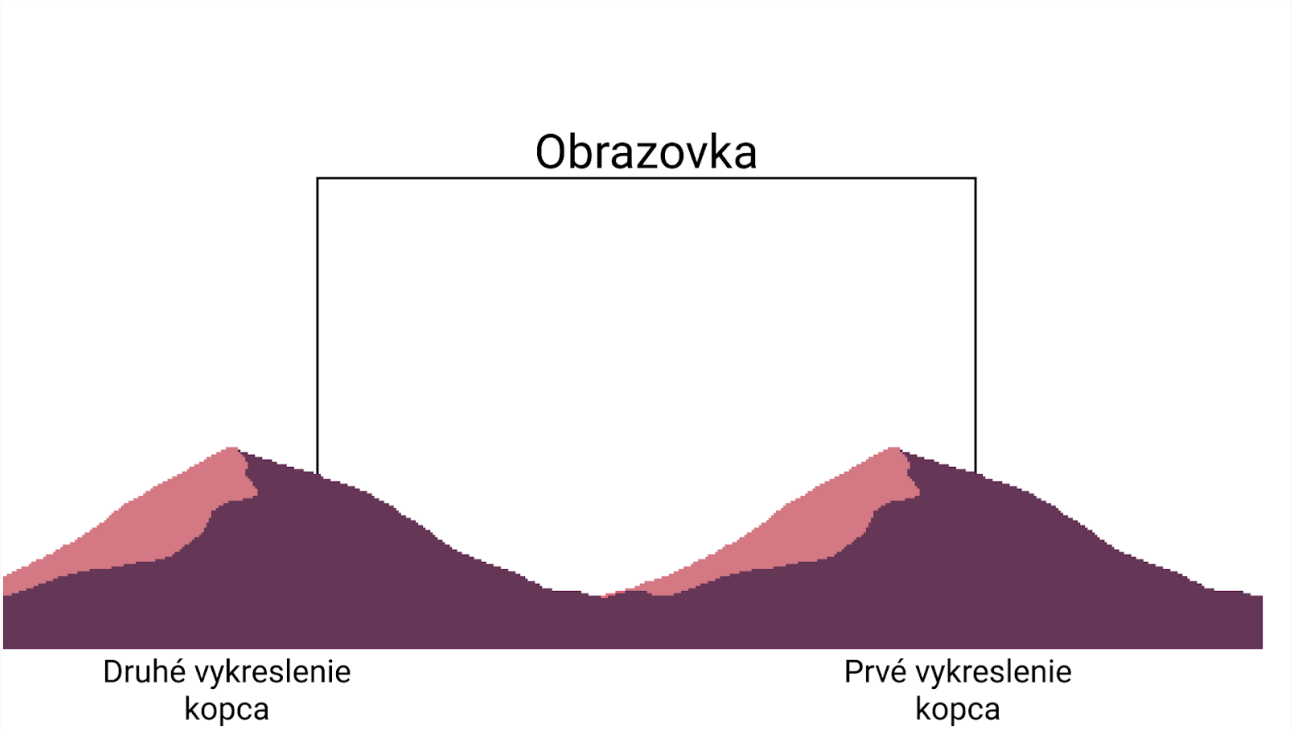
Pozor: záleží na poradí v akom obrázky vykreslujete, obrázok ktorý vykreslíte ako prvý bude úplne na spodku a ostatné obrázky ho prekryjú

3.2 Úloha

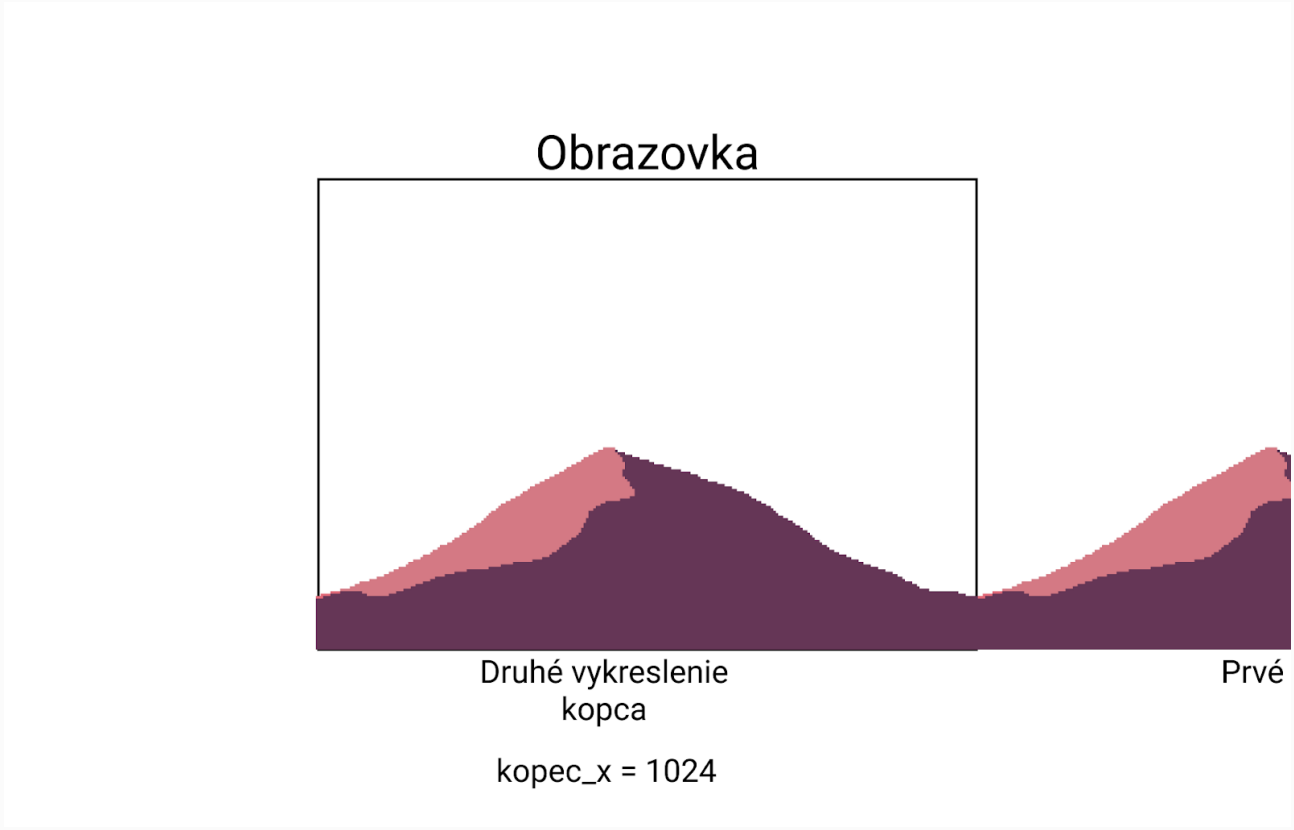
Načítajte obrázky “Mountain.png” a “MountainsBackground.png” a skúste im nastaviť rôzne rýchlosti pohybu aby ste vytvorili efekt vzdialenosti.

Všimnite si že horami síce hýbeme, no postupne nám zmízajú z obrazovky. Tento problém vieme vyriešiť jednoduchým trikom ktorý sa nazýva parallax.

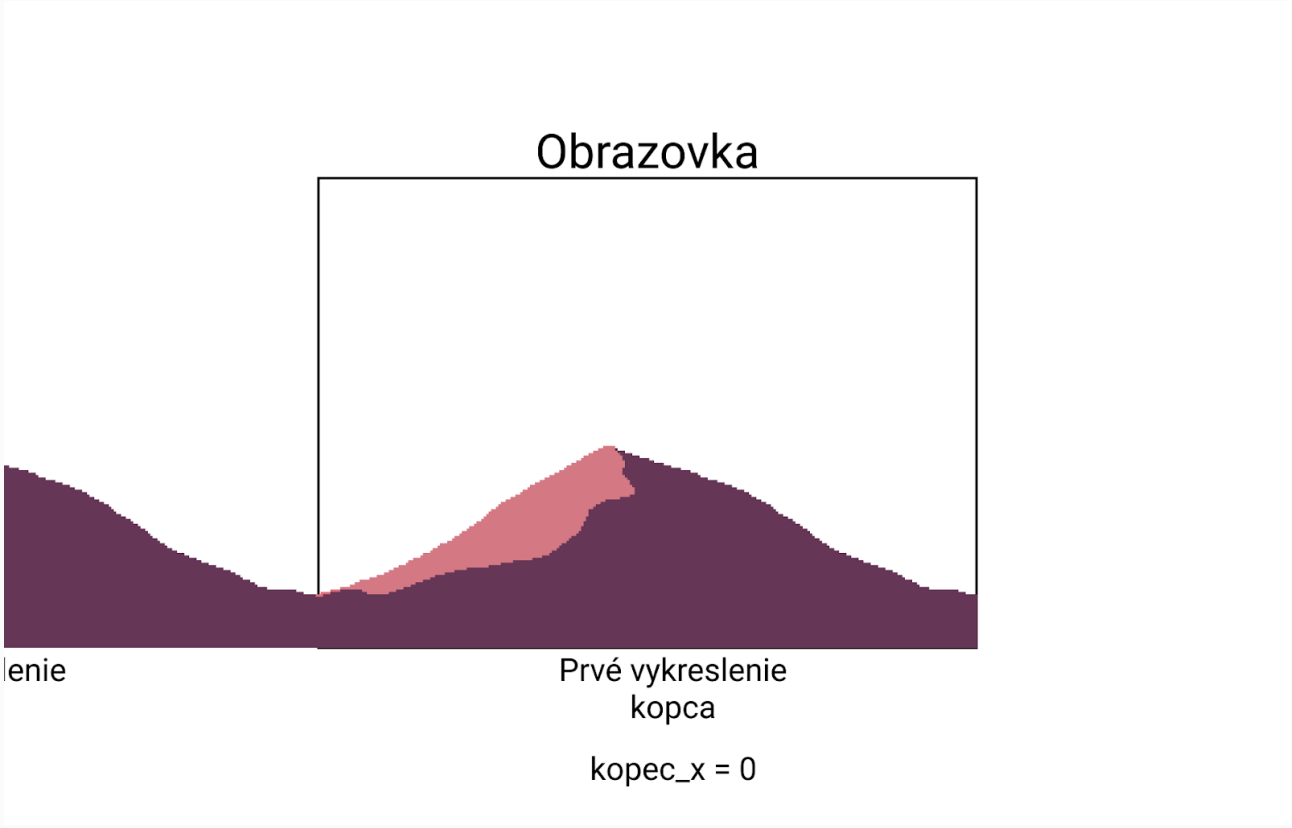
Ide o jednoduchý trik kde vykreslíme dvakrát ten istý obrázok za seba a budeme oba obrázky hýbať spoločne tak, aby to vyzeralo že sa obrázok navždy opakuje.



Len čo prídeme na koniec, tj. druhý kopec bude v strede našej obrazovky. Oba obrázky presunieme na ich základnú pozíciu, zmeníme ich x pozíciu znova na 0.



↓



Vyskúšajte

```
import pygame

pygame.init()

screen = pygame.display.set_mode((1024, 602))
clock = pygame.time.Clock()

running = True
sky = pygame.image.load('./Sky.png').convert_alpha()
kopec = pygame.image.load('./Mountain.png').convert_alpha()

sky = pygame.transform.scale(sky, (1024, 602))
kopce = pygame.transform.scale(kopce, (2048, 602))
kopec = pygame.transform.scale(kopec, (1024, 602))

kopec_x = 0

def nakresli_kopce():
    screen.blit(kopec, (kopec_x, 0))
    screen.blit(kopec, (kopec_x - 1024, 0))

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break

    kopec_x += 0.25
    if (kopec_x > 1024):
        kopec_x -= 1024 # aby sa naša pozícia nezväčšovala donekonečna
        # budeme ju po každom prechode vracať o jednu obrazovku späť

    screen.blit(sky, (0, 0))
    nakresli_kopce()

    pygame.display.update()

    clock.tick(60)

pygame.quit()
```

Pre väčšiu prehľadnosť sme si vytvorili funkciu **nakresli_kopce()**, ktorá nám kopce za nakreslí presne za sebou (posun je o šírku jednoho kopca, tj 1024)

Pohrajte sa s rôznymi nastaveniami, či rýchlosťami pohybu, aby ste si osvojili ako to presne funguje.

Vstup užívateľa

Aby sme hru mohli nejako ovládať, musím registrovať vstup od užívateľa.
Na toto využijeme list events (po slovensky udalosti), z ktorých získavame aj zatvorenie hry.

Tentokrát ale budeme sledovať či je typ eventu KEYDOWN, teda:

if event.type == pygame.KEYDOWN:

Následne, aby sme nemali v našom hlavnom cykle neporiadok si vytvoríme funkciu **spracuj_keydown(event)**, ktorej budeme posielať náš event.

Ak teda nastane event typu keydown, zavoláme našu funkciu a pošleme jej aktuálny event, ktorý má v sebe uložené o ktorú klávesu ide.

Na to aby sme zistili ktoré tlačidlo užívateľ stlačil, má pygame pripravené premenné s hodnotami s ktorými môžeme porovnávať. Pre všekty písmená a čísla na klávesnici sú tieto premenné:

pygame.K_a	pre klávesu a (a podobne pre ostatné písmená)
pygame.K_0	pre číslo 0 (a podobne pre ostatné čísla)
pygame.K_SPACE	pre klávesu space
pygame.K_UP	šípka hore
pygame.K_DOWN	šípka dole
pygame.K_RIGHT	šípka doprava
pygame.K_LEFT	šípka doľava

Tieto nám budú v dnešnej hodine stačiť, v prípade že by ste si chceli pozrieť aj iné klávesy, môžete ich nájsť tu: **https://www.pygame.org/docs/ref/key.html**

Ak ste zatiaľ zmätený, nebojte sa len čo dáme kód dokopy, mal by vám začať dávať zmysel. Znova môžeme použiť kód z predošlej hodiny, kde sme si poriadne nastavili cyklus.

(pridáme do kódu funkciu **screen.fill((255, 255, 255))** ktorý vyplní obrazovku bielou (alebo ak farbu zmeníte hociakou) farbou, aby sme videli naše obrázky)

Vyskúšajte

```
import pygame

pygame.init()
screen = pygame.display.set_mode((1024, 720))
clock = pygame.time.Clock()

running = True
player = pygame.image.load('./Player.png').convert_alpha()

player_x = 0
player_y = 0

def spracuj_keydown(event):
    global player_x, player_y
    if event.key == pygame.K_w:
        player_y -= 1
    if event.key == pygame.K_s:
        player_y += 1
    if event.key == pygame.K_a:
        player_x -= 1
    if event.key == pygame.K_d:
        player_x += 1

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break
        if event.type == pygame.KEYDOWN:
            spracuj_keydown(event)

    screen.fill((255, 255, 255))
    screen.blit(player, (player_x, player_y))

    pygame.display.update()
    clock.tick(60)
pygame.quit()
```

Ako môžete vidieť jednoducho, do forcyklu kde prechádzame eventy pridáme podmienku, keď je typ našej udalosť **KEYDOWN** a v nej zavoláme funkciu **spracuj_keydown(event)**, v ktorej už môžeme pracovať s jednotlivými tlačidlami.
V príklade sa na pohyb používa “wasd” no neváhajte si klávesy zmeniť napríklad na šípky.

Obrázok hráča si môžete stiahnuť z materiálov pre túto hodinu, alebo kľudne použite vlastný obrázok. Pamätajte ale že musí byť v rovnakom priečinku ako váš py súbor.

Pohyb hráča nám funguje, no všimnite si že musíte opakovane stláčať klávesu na to, aby ste sa vedeli plynule hýbať. Preto musím náš kód trocha upraviť a pridať nášmu hráčovi rýchlosť.

Pre predstavu si teda vytvoríme premennú pre rýchlosť pohybu a následne štyri premenné pre smer ktorým ideme. Ak stlačíme napríklad “w” premenná pohyb_hore sa nastaví na true.

Následne v našom while cykle budú štyri podmienky, jedna pre každý smer. Ak je niektorý smer true pohneme hráča do daného smeru o jeho rýchlosť.

Vyskúšajte

```
import pygame

pygame.init()
screen = pygame.display.set_mode((1024, 720))
clock = pygame.time.Clock()

running = True
player = pygame.image.load('./Player.png').convert_alpha()

# moje premenné
player_x = 0
player_y = 0
rychlost = 2
pohyb_hore = False
pohyb_dole = False
pohyb_doprava = False
pohyb_dolava = False

def spracuj_keydown(event):
    global pohyb_hore, pohyb_dole, pohyb_doprava, pohyb_dolava
    if event.key == pygame.K_w:
        pohyb_hore = True
    if event.key == pygame.K_s:
        pohyb_dole = True
    if event.key == pygame.K_a:
        pohyb_dolava = True
    if event.key == pygame.K_d:
        pohyb_doprava = True

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break
        if event.type == pygame.KEYDOWN:
            spracuj_keydown(event)

    if (pohyb_hore):
```

```
        player_y -= rychlost
    if (pohyb_dole):
        player_y += rychlost
    if (pohyb_dolava):
        player_x -= rychlost
    if (pohyb_doprava):
        player_x += rychlost

    screen.fill((255, 255, 255))
    screen.blit(player, (player_x, player_y))

    pygame.display.update()
    clock.tick(60)
pygame.quit()
```

Hráč sa nám teraz hýbe plynulo do smeru ktorý klikneme, no niekde je chyba, nezastavuje sa.

Na to aby sa zastavil musíme pridať do našich events podmienku pre typ **pygame.KEYUP**, aby sme zistili či užívateľ klávesu pustil. Pre keyup si vytvoríme funkciu **spracuj_keyup(event)** ktorá bude v podstate rovnaká ako pre keydown, ibaže budeme hodnoty nastavovať na **False**.

Vyskúšajte

```
import pygame

pygame.init()
screen = pygame.display.set_mode((1024, 720))
clock = pygame.time.Clock()

running = True
player = pygame.image.load('./Player.png').convert_alpha()

# moje premenné
player_x = 0
player_y = 0
rychlost = 2
pohyb_hore = False
pohyb_dole = False
pohyb_doprava = False
pohyb_dolava = False

def spracuj_keydown(event):
    global pohyb_hore, pohyb_dole, pohyb_doprava, pohyb_dolava
    if event.key == pygame.K_w:
        pohyb_hore = True
    if event.key == pygame.K_s:
        pohyb_dole = True
```

```
    if event.key == pygame.K_a:
        pohyb_dolava = True
    if event.key == pygame.K_d:
        pohyb_doprava = True

def spracuj_keyup(event):
    global pohyb_hore, pohyb_dole, pohyb_doprava, pohyb_dolava
    if event.key == pygame.K_w:
        pohyb_hore = False
    if event.key == pygame.K_s:
        pohyb_dole = False
    if event.key == pygame.K_a:
        pohyb_dolava = False
    if event.key == pygame.K_d:
        pohyb_doprava = False

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break
        if event.type == pygame.KEYDOWN:
            spracuj_keydown(event)
        if event.type == pygame.KEYUP:
            spracuj_keyup(event)

    if (pohyb_hore):
        player_y -= rychlost
    if (pohyb_dole):
        player_y += rychlost
    if (pohyb_dolava):
        player_x -= rychlost
    if (pohyb_doprava):
        player_x += rychlost

    screen.fill((255, 255, 255))
    screen.blit(player, (player_x, player_y))

    pygame.display.update()
    clock.tick(60)
pygame.quit()
```

4.1 Úloha

Vytvorte funkciu **pohyb_hraca()** ktorú budeme volať vo while cykle a presuňte do nej naše štyri podmienky pre pohyb.

4.2 Úloha

Pridajte obmedzenie, aby sa hráč nevedel "odísť z našej aplikácie" teda aby jeho pozícia nebola nikdy väčšia ako šírka/výška našej aplikácie, alebo menšia ako nula

Hint: Treba upraviť našu funkciu **pohyb_hraca()** a pridať do nej 4 vnorené podmienky

4.3 Úloha

Z materiálov si stiahnite obrázok "Sword.png", načítajte ho a vykreslite ho vedľa hráča, tak aby to vyzeralo že ho hráč nesie. (Tento budeme potrebovať na budúcej hodine na porazenie duchov)

4.4 Úloha

Pridajte možnosť švihať mečom. Na teraz stačí jednoducho, ak užívateľ stlačí SPACE, meč sa začne točiť (použite transform.rotate), keď SPACE pustí, prestane sa točiť.

4.5 Bonusová úloha

Všimnite si že funkcie **spracuj_keydown** a **spracuj_keyup** sú v podstate rovnaké až na nastavenie na T**rue/False**. Skúste funkcie spojiť do jednej funkcie **spracuj_key(...)**, náš kód bude tak oveľa prehladnejší.

Kolízie a pygame.Rect

Doteraz sme obrázky pri rotácii či škálovani menili priamo. Tento spôsob ale nie je optimálny, je viac zaťažujúci na počítač a spôsobuje ďalšie problémy v prípade že chcem ten istý obrázok vykresliť viac krát, v rôznych pozíciách. Navyše sa takto kolízie počítajú veľmi zložito, a vďaka funkcionalite pygame si vieme prácu o dosť zjednodušiť.

Funkcie pygame.Rect

Rect obsahuje viaceré funkcie ktoré budeme postupne používať.

Hlavné z nich ktoré budú zaujímať nás sú:

rect.move(x, y) # posunie rect o dané x/y a vráti nový, posunutý rect

rect.update(left, top, width, height) # nastaví pozíciu a veľkosť rect

rect.x, y, width, height # vráti hodnotu pozície, alebo veľkosti

rect.centerx a **.centery** # vráti pozície stredu našeho štvoruholníka

rect.colliderect(iny_rect) # vráti True alebo False, podľa toho či sa rect a iny_rect prekrývajú, touto funkciou budeme v našej hre testovať kolízie.

Začneme tým že namiesto premenných player_x a player_y si vytvoríme jednu premennú player_rect.

player_rect = player.get_rect(center = (0, 0))

player (náš obrázok) nám vďaka tejto funkcii vráti štvoruholník, rovnaký ako keď sme napríklad vykreslovali rect na druhej hodine, a vycentruje ho nastaví mu pozíciu na 0,0.

Ak teraz chceme vykresliť hráča, nemusíme už vytvárať tuple z jeho pozície, môžeme jednoducho zavolať, nemusíme už používať jednotlivé premenné:

screen.blit(player, player_rect)

alebo pre jeho meč

screen.blit(sword, player_rect.move(20, 0))

Keďže funkcia **player_rect.move(20, 0)** nám vráti rect hráča posunutý o 20 pixelov na osy x, ale originál zachová bez posunu.

5.1 Úloha

Zoberte si posledný kód z predošlej hodiny a nahraďte premenné **player_x** a **player_y** objektom rect tak ako sme si ukázali vyššie.

Nakoniec ešte upravte funkciu **pohyb_hraca()** tak aby využívala rect, namiesto týchto dvoch premenných.

5.2 Bonusová úloha

Všimnite si že teraz sa hráč dokáže pohnúť iba po okraj vľavo a hore, no vie úplne odísť z obrazovky vpravo a dole. Viete prečo to tak je? Skúste tento problém vyriešiť.

Hint: Treba použiť **rect.centerx** a **centery** vo funkcii **pohyb_hraca()**

Kolízie

Kolízie v hrách fungujú v celku jednoducho. Každý objekt ktorý ich potrebuje, má okolo seba nejaký priestor (vo väčšine prípadov kvôli jednoduchosti štvorec v 2d, alebo kocku v 3d, no zložitejšie hry môžu mať toto okolie presnejšie) pre ktorý sa v každom cykle hry pozerá či nenarazil na iný objekt, teda či sa nepretínajú. Tento priestor sa volá bounding box, no niekedy sa nazýva “hitbox”, možno ste sa už v niektorých videohrách s týmto pojmom aj stretli.

Pre zistenie či sa dva štvorce (alebo rect-y) pretínajú sa používa celkom jednoduchá geometria, táto nie je obsahom tohto kurzu, keďže ste sa ju možno ešte neučili, no pygame tento problém rieši za nás funkciou **colliderect()**, ktorá v jednoduchosti vráti hodnotu **True**, alebo **False**, podľa toho či sa dané štvorce prekrývajú/dotýkajú.

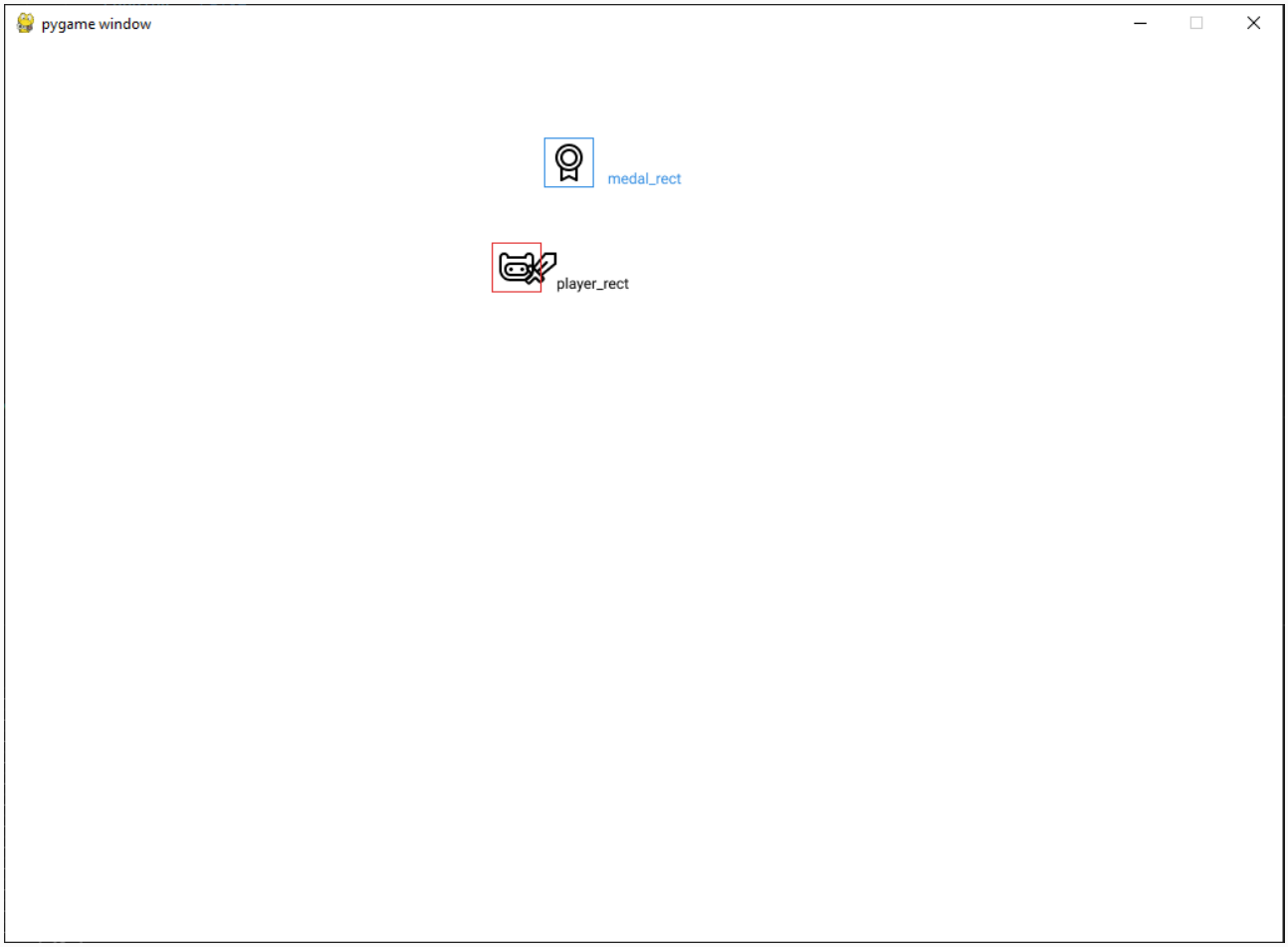
Vyskúšajte

Skúsme do hry nahrať obrázok Medal.png a vykresliť ho na náhodnej pozícii. Následne vytvoríme funkciu **check_collision()** ktorá nám nateraz vypíše či hráč a medaila “narazili”. Nakoniec nám stačí túto funkciu už iba zavolať v našom cykle.

```
import pygame
from random import randint

pygame.init()
screen = pygame.display.set_mode((1024, 720))
clock = pygame.time.Clock()

running = True
player = pygame.image.load('./Player.png').convert_alpha()
sword = pygame.image.load('./Sword.png').convert_alpha()
```

```
medal = pygame.image.load('./Medal.png').convert_alpha()

# moje premenné
player_rect = player.get_rect(center = (0, 0))

# položíme medailu na náhodné miesto, vrámci našej obrazovky
medal_rect = medal.get_rect(center = (randint(0, 1024), randint(0, 720)))

rychlost = 5
pohyb_hore = False
pohyb_dole = False
pohyb_doprava = False
pohyb_dolava = False
# funkcia na zistenie či sme narazili na medailu
def check_collision():
    global player_rect, medal_rect
    if player_rect.colliderect(medal_rect):
        print("Získal si medailu!")

def spracuj_key(event, hodnota):
   ... # nemeníme

def pohyb_hraca():
    ... # nemeníme

while running:
    for event in pygame.event.get():
        ... # nemeníme

    pohyb_hraca()
    # voláme v každom cykle po pohybe hráča
    check_collision()

    screen.fill((255, 255, 255))
    screen.blit(player, player_rect)
    # pridáme vykreslenie medaile
    screen.blit(medal, medal_rect)
    screen.blit(sword, player_rect.move(20, 0))

    pygame.display.update()
    clock.tick(60)
pygame.quit()
```

Pre lepšie pochopenie tu máte ilustráciu ako bounding boxy fungujú, jednoducho obkolesujú náš obrázok a my v každom cykle voláme funkciu ktorá sleduje či sa náhodou neprekrývajú.



5.3 Úloha

Vytvorte premennú pre počítanie medailí a rátajte koľko ich hráč po zbieral. Aby nestačilo stáť na mieste, pri zobraní medaile ju presuňte na nové miesto pomocou funkcie **update()**

5.4 Úloha

Pridajte viacej medailí, počet záleží na vás (netreba odznova načítavať obrázok Medal, stačí ak pre každú medailu vytvoríte jej rect) a pre všetky sledujte kolízie a ich zber.

5.5 Úloha

Pridajte do hry nepriateľov do ktorých ak hráč narazí, odoberie mu to jednu medailu a vráti ho to naspäť na pozíciu 0,0. Po náraze sa nepriateľ takisto presunie na novú náhodnú pozíciu. (Môžete použiť obrázok “Ghost.png”)
Môžete vytvoriť novú funkciu pre kolízie s nepriateľmi, alebo jednoducho rozšírte funkciu **check_collision()**

Animácie

Animácie v 2d videohrách fungujú vcelku jednoducho. Stačí ak máte pripravené obrázky, ktoré reprezentujú jednotlivé snímky pohybu, a tie postupne vymieňate, takže to vyzerá že sa obrázok hýbe.

Tieto obrázky, nazývame frames a pre každú animáciu definujeme ako dlho bude jeden frame vidieť.

Na uloženie obrázkov nám stačí použiť list a pre zistenie či môžeme použiť funkcionalitu pygame nazvanú userevent. Takýto event vytvoríme a následne vieme kontrolovať či sa deje rovnako ako napríklad KEY_DOWN.

(Budeme využívať nové obrázky dinosaura)

Vyskúšajte

Použite kód zo štvrtej hodinya vyskúšajte vykreslenie statického obrázku, nahradiť kreslením jednotlivých frames

```
import pygame

pygame.init()
screen = pygame.display.set_mode((1024, 720))
clock = pygame.time.Clock()

running = True
idle_frames = []
for i in range(1, 5):

idle_frames.append(pygame.image.load('./idle/idle0'+str(i)+'.png').convert_alpha(
))


# moje premenné
current_frame = 0
player_rect = idle_frames[current_frame].get_rect(center = (0, 0))

ANIMATION_EVENT = pygame.USEREVENT
pygame.time.set_timer(ANIMATION_EVENT, 200)

rychlost = 5
pohyb_hore = False
pohyb_dole = False
pohyb_doprava = False
pohyb_dolava = False
```

```
def spracuj_key(event, hodnota):
    # funkcia spracuj_key..

def pohyb_hraca():
     # funkcia pohyb_hraca..

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            break
        if event.type == pygame.KEYDOWN:
            spracuj_key(event, True)
        if event.type == pygame.KEYUP:
            spracuj_key(event, False)
        # zisťujeme či nastal event na zmenu animácie
        if event.type == ANIMATION_EVENT:
            current_frame += 1
            if (current_frame > 3):
                current_frame = 0

    pohyb_hraca()

    screen.fill((255, 255, 255))
    screen.blit(idle_frames[current_frame], player_rect)

    pygame.display.update()
    clock.tick(60)
pygame.quit()
```

Ďalším krokom je prepínanie medzi animáciou kráčania a státia.

Vyskúšajte si tento krok ako úlohu samostatne. Ak by ste mali problémy, neváhajte si pomôcť vzorovým riešením, prípadne sa spýtať lektora.

6.1 Úloha

Pridajte do vašej hry animáciu kráčania.

Hint: aby ste nemuseli mať dve premenné pre ukladanie aktuálnej frame, skúste porozmýšľať či by ste nevedeli jednoducho využiť modulo

6.2 Úloha

Skúste dosiahnuť to, aby sa dinosaurus otáčal do smeru do ktorého kráča (stačí doľava a doprava)

6.3 Úloha

Podobne ako na minulej hodine, pridajte do hry zberateľné coins na náhodných miestach, no spravte ich animované

6.3.1 Úloha - Bonus

Pridajte animáciu zobratia mince

6.3.2 Úloha - Bonus

Pridajte na ľubovoľné tlačidlo možnosť kopnúť (animácia v priečinku kick) a takto zobrať mincu zo zeme

Hint: Treba pridať stav kopnutie, vtedy prehrať animáciu kopnutia a iba počas tohoto stavu sledovať kolízie okolo seba

Opakovanie - Flappy Bird

7.1 Úloha

Vytvorte nový súbor pre hru (nebojte sa pomôcť si predošlými hodinami), ktorá bude mať obrazovku v rozmeroch 288x512 a na pozadí jeden obrázok “background-day.png”

7.1.1 Úloha

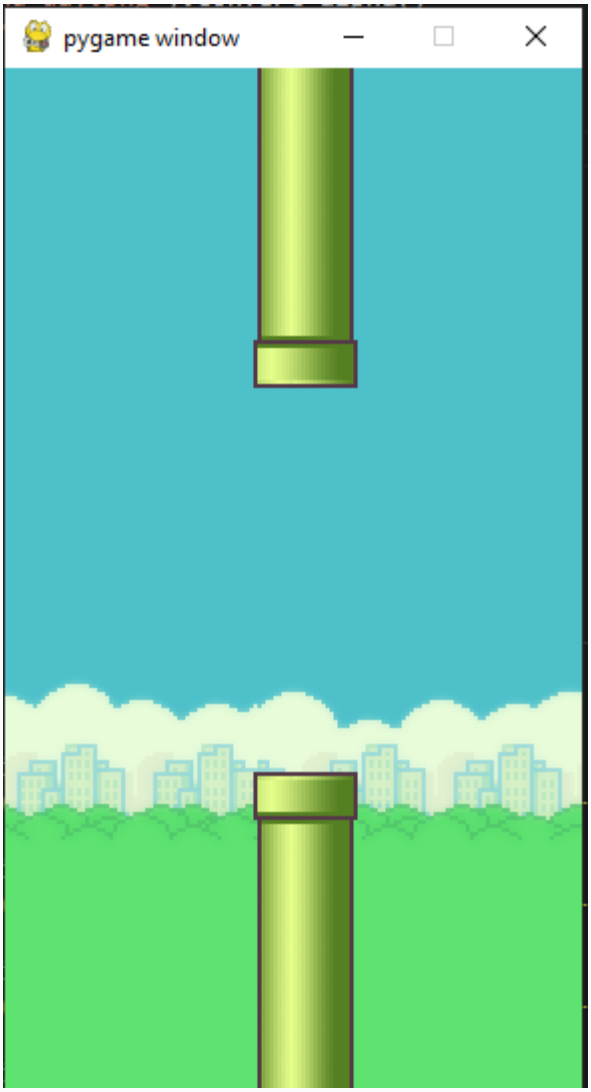
Vytvorte s pozadia “nekonečný” parallax hýbajúci sa doľava.

7.2 Úloha

Pridajte do hry potrubie tak aby bolo na obrazovke aj z vrchu aj zospodu tak ako v originálnej hre.

Použite rovnaký asset, a pomocou transform si vytvorte kópiu ktorú otočíte buď pomocou flip alebo rotate. Pre vykreslenie na určitej pozícii použite Rect vygenerovaný z obrázkov, ktorý neskôr použijeme na kolízie.

Výsledok by mal vyzerať podobne:



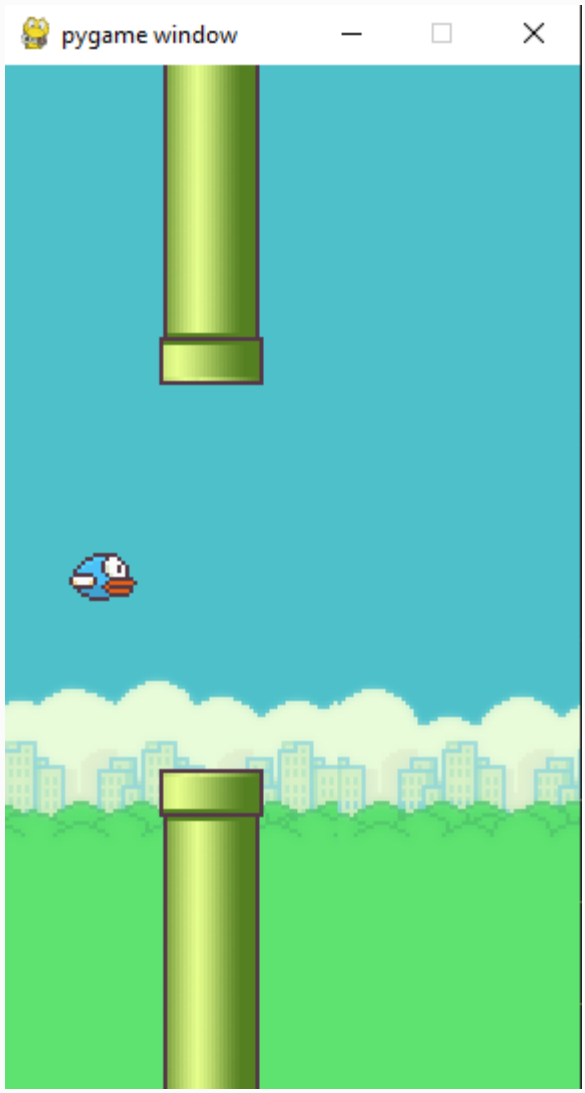
7.3 Úloha

Vytvorte premennú v ktorej bude uložená x pozícia trubiek a podobne ako pozadie ju posúvajte doľava. Po prechode nulou ju presuňte na začiatok, tak aby to vyzeralo že sa vytvorili nové

Bonus: Skúste ich po každom “teleportovaní” na začiatok posunúť o kúsok vyššie alebo nižšie pomocou knižnice random

7.4 Úloha

Pridajte do scény flappy birda ktorý bude jednoducho stáť na mieste (alebo letieť vďaka našemu pozadiu) a vždy keď narazí do niektorej z pipes tak vypíšte “Ouch”



7.5 Úloha

Jednoduchá verzia: Umožnite hráčovi chodiť birdom hore a dolu pomocou kláves “**w**” a “**s**”

Bonusová verzia: Pridajte do hry gravitáciu, takže bude bird stále padať dolu, no akonáhle hráč stlačí “**space**”, posunie sa o trocha vyššie. Teda vytvorte jednoduchú kópiu pohybu v originálnej hre

7.5.1 Úloha

Ak bird zletí príliš nízko, takisto vypíšte ouch (prípadne ukončite hru zmenou premennej running na **False**)

7.6 Úloha

Využite obrázky “**bluebird-downflat**” “**bluebird-midflap**” a “**bluebird-upflap**” (prípadne inú obľúbenú z farieb) a animujte bird-ovi mávanie krídel