5.4 O'zingizga rasm chizish taxtasini yasang - Hodisa

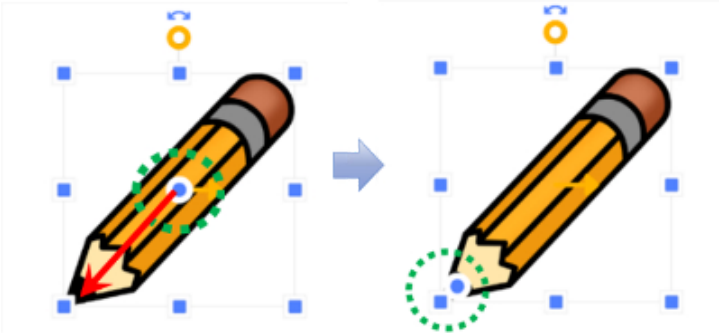
Endi yettita misoldan 4-sini ko'rib chiqish vaqti keldi. Kitobning yakuniga oz qoldi. Kod miqdori, albatta, avvalgidan ko'proq bo'ladi. Biroq, tashvishlanishning hojati yo'q, chunki qiyinchiliklar bilimni oshirishda ijobiy xizmat qilishini tushunsak, biz uchun foydali bo'ladi.

{% code lineNumbers="true" %}

```
from pgzhelper import *

WIDTH = 960
HEIGHT = 540

drawing = False

pencil = Actor('pencil', (WIDTH / 2, HEIGHT / 2), anchor=('left', 'bottom'))
pencil.scale = 0.3
pencil.brush_init((WIDTH, HEIGHT), 5, 'blue')
ochirgich = Actor('eraser', (900, 50))
ochirgich.scale = 0.5

def draw():
    screen.fill('white')
    pencil.brush_draw()
    pencil.draw()
    ochirgich.draw()

    if drawing is False:
        pencil.brush_stop()

def on_mouse_move(pos):
    pencil.left, pencil.bottom = pos

def on_mouse_down(pos):
    global drawing
    drawing = True

    if ochirgich.collidepoint_pixel(pos):
        pencil.brush_clear()

def on_mouse_up():
    global drawing
    drawing = False
```

{% endcode %}

Keling birinchi 8-qatorda qalam nomli aktyor obyektini yaratuvchi kodni ko'rib chiqamiz. Obyektning langar qiymati (anchor qiymati) "chap", "pastki" ('left', 'bottom')ligi sababi Entry blokli kodlashdagi bilan bir xil. Buning sababi shundaki, ekranda qalam tanasidan chizish noodatiy, qalam uchidan chizishi tabiiydir.



6-qatorda, avvalgi ichki kutubxona o‘zgaruvchilaridan farqli bo‘laroq, bizning ehtiyojlarimizga mos ravishda biz tomonidan aniqlangan o‘zgaruvchi birinchi marta paydo bo‘ladi. **drawing** deb nomlangan o‘zgaruvchining maqsadini nomidan ma’lum darajada taxmin qilish mumkin (o‘zgaruvchi yaratishda mazmunli nom berish muhimligini avvalgi darslarda bir necha marta eslatganmiz). Bu o‘zgaruvchi sichqoncha tugmachasi bosilgan holda chizish holatida ekanini yoki sichqoncha tugmachasidan barmoqni ko‘tarib, chizishni to‘xtatgan holatni aniqlash uchun ishlatiladi. Oxir-oqibat, faqat ikkita holatni bilish kifoya bo‘lgani uchun ushbu o‘zgaruvchi qiymati sifatida faqat Rost/Yolg'on(True/False) qiymatlaridan foydalanish yetarli.

Endi drawing o‘zgaruvchisining maqsadi va vazifasini tushunib oldik. Navbatda *ushbu o‘zgaruvchining ta’rifi (definition) kod ichida qayerda joylashgani va nima uchun bu o‘zgaruvchi funksiyaning ichki sohasida (scope) emas, balki tashqi sohasida joylashganini o‘rganish zarur.* **Bu masala o‘zgaruvchini qayerda yaratish kerakligi haqidagi kodlash qoidalariga bog‘liq. Agar o‘zgaruvchi faqat bitta funksiyada ishlatilishi kerak bo‘lsa, u holda u funksiyaning ichida aniqlanadi. Ammo agar bir nechta funksiyalarda yoki kodning umumiy qismlarida birgalikda ishlatilishi kerak bo‘lsa, bu kabi funksiyalardan tashqarida aniqlanadi. Shuni ham yodda tutingki, kodning umumiy qismlarida birgalikda ishlatiladigan o‘zgaruvchilar global (umumiy) o‘zgaruvchilar deb ataladi.**

Ko'rishingiz mumkin, **drawing o'zgaruvchisi ikkita joyda ishlatilgan**: 26-qatorda **on_mouse_up** qayta chaqirish funksiyasi (funksiya nomi **on**_ bilan boshlanganligi sababli, u PygameZero kutubxonasi tomonidan hodisaga (masalan, sichqoncha tugmasi bosilishi) ko‘ra avtomatik ravishda chaqirilishini bildiradi)) va 33-qatorda **on_mouse_down** qayta chaqirish funksiyasi. U ikkala funktsiya tomonidan umumiy bo'lishi kerak bo'lgan o'zgaruvchi bo'lgani sababli funktsiyalardan tashqarida yaratilgan.

Keling, ushbu o'zgaruvchidan funktsiyada qanday foydalanishni batafsil ko'rib chiqaylik. **U funktsiyadan tashqaridagi o'zgaruvchi bo'lsa ham, avvalgiday siz shunchaki o'zgaruvchining qiymatini o'qib, uni funksiya ichida ishlatishingiz mumkin. Ammo, agar siz ushbu o'zgaruvchini funktsiya ichida o'zgartirmoqchi bo'lsangiz, Pythonning yangi grammatikasini bilishingiz kerak. 27 va 34 qatorlarda ko'rib turganingizdek, o'zgaruvchi nomining oldiga global kalit so'zini(keyword) qo'shish orqali o'zgaruvchini e'lon qilishingiz kerak, bu o'zgaruvchining qiymatini o'zgartirishdan oldin uning funksiyadan tashqarida mavjudligini aniq ko'rsatadi.**

{% hint style="info" %} *Aytgancha, bu erda ehtiyot bo'lish kerak.* ***Grammatikaga qat'iy rioya qilmaydigan Python-ning moslashuvchanligi afzallik, lekin shu bilan birga, bu kamchilik.*** *Agar siz global kalit so'zni aniq ishlatmasangiz va uni ishga tushirmasangiz, u oldindan xatosiz ishlaydi. Biroq, o'zgaruvchi qiymatini o'zgartirishga urinayotganda xatolik yuz beradi. Nima uchun siz xohlaganingizcha ishlamasligi haqida oldindan mantiqiy xatolarni topish oson bo'lmagan vaziyatlarga duch kelishingiz mumkin, shuning uchun ehtiyot bo'ling.* {% endhint %}

15~18-qatorlarda **draw** qayta chaqirish funksiyasida fon rangi o'rnatiladi va ekranda paydo bo'ladigan obyektlar chiziladi. Yuqorida aytib o'tilganidek, har bir obyekt uchun **draw** funktsiyalarini chaqirish tartibini,

ya'ni ularni ekranda ko'rsatish tartibini hisobga olish muhimdir. Bizning chizganimiz qalam ustida paydo bo'ladigan xatoli vaziyatlarga duch kelmaslik uchun ushbu tartibni saqlang.

20-21-qatorlarda **drawing** o'zgaruvchisining qiymatidan foydalanib, cho'tka bilan chizishni to'xtatish vaqtini aniqlash uchun shart ishlatiladi. Qiymat **yolg'on** (False) bo'lganda, **brush_stop** funktsiyasi (usuli) cho'tkadan foydalanishni to'xtatadi.

23-24 qatorlarda qalam obyektining markazini sichqoncha koordinatalari bilan real vaqtda sinxronlashtirish (joylashtirish) vazifasini bajariladi. Bu qalam obyekti kursor bilan birga harakatlanishi uchun kerak. 24-qatordagi sintaksis biroz g'alati tuyulishi mumkin, chunki **"pos" qiymati bittalik qiymatga o'xshaydi, lekin aslida (x, y) koordinatalarini o'z ichiga olgan *o'zgarmas ro'yxat* (Tuple) ni ifodalaydi.** O'zgarmas ro'yxat (Tuple) — bu biz hali duch kelmagan yangi ma'lumotlar turi, shuning uchun uni batafsilroq ko'rib chiqamiz.

O'zgarmas ro'yxat (keyinchalik "Tupl" deb ataladi)

Tupllar biz allaqachon tanish bo'lgan ro'yxat (list) ma'lumotlari turiga o'xshaydi, shuning uchun avval ulardan foydalanish sintaksisini ajratib olaylik. Tashqi farq shundaki, ro'yxatning to'rtburchak qavslar "[ ]" odatiy qavslar "( )" bilan almashtiriladi.

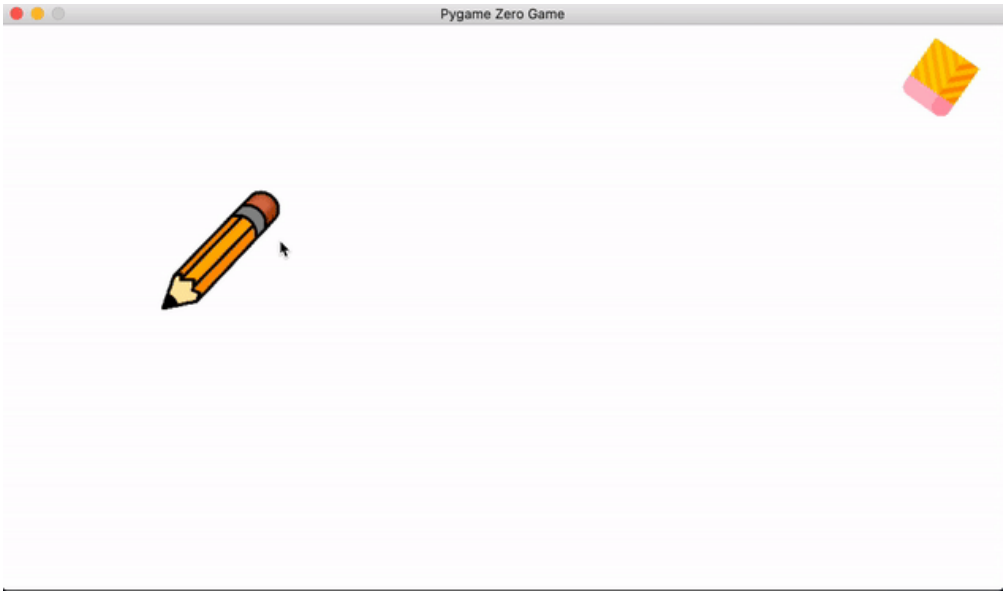
> **tupl_nomi = (bir nechta qiymatlar, masalan, harflar, raqamlar va boshqalar, lekin qiymatlar vergul (,) bilan ajratiladi)**

Biroq, ro'yxat va tupl o'rtasidagi asosiy farq keyinchalik qiymatlarni o'zgarish imkonidir. Bu o'zgaruvchan va faqat o'qish mumkin bo'lgan (read only) ma'lumotlar turlari o'rtasidagi farq. Tupllar ikkinchisiga qarashli — yaratilgandan keyin ularning qiymatlarini o'zgartirib bo'lmaydi. Bu yaratuvchi va dasturning boshqa foydalanuvchilari tomonidan dasturlash jarayonida tasodifiy o'zgarishlarning oldini olish orqali dastlabki qiymatlarni himoya qiladi. Funktsiya muallifi tomonidan belgilangan cheklovlarga rioya qilishimiz kerak. Boshqa dasturchilar tomonidan yozilgan funktsiyalar yoki kutubxonalardan foydalanganda, biz shikoyat qilishimiz qiyin (?) va kodlashni funktsiya muallifi tomonidan taklif qilingan yoki cheklangan doirada ishlash deb hisoblash osonroq. Barcha cheklovlar to'g'risidagi ma'lumotlarni tegishli funktsiya uchun kutubxona qo'llanmasida topish mumkin.

Shunday qilib, yana bir bor xulosa qilish uchun 24-qator sintaksisi pos **tupl**idan x-koordinataning qiymatini oladi va uni qalam.left o'zgaruvchisiga saqlaydi, va y-koordinataning qiymati olinadi va qalam.bottom da saqlanadi.

26-qatordagi **on_mouse_down** qayta chaqirish qilish funksiyasi, uning nomidan taxmin qilganingizdek, sichqoncha tugmasi bosilganda kutubxona orqali avtomatik ravishda chaqiriladi. Bunday holda, bosish paytida sichqoncha kursori joylashgan nuqtaning koordinatalari avtomatik ravishda pos funktsiya parametri sifatida uzatiladi. Ushbu qiymatdan foydalanib, biz qiziqarli dasturlarni yaratishimiz mumkin. Endi, nihoyat, ushbu bob kodining oxirgi 30-31 qatorlarini ko'rib chiqamiz. Bizning kodimizda birinchi marta obyektlar o'rtasida to'qnashuvni tekshirish paydo bo'ladi, bu ko'pincha o'yin kodlashda ishlatiladi. **collidepoint_pixel** funktsiyasi (usuli) sichqonchani bosgan nuqta (bu yana pos tupl qiymati) o'chirgich obyektining ichida ekanligini tekshiradi. Agar shunday bo'lsa (natija rost bo'lsa), biz o'chirgich obyektini bosganimizni tekshiramiz va agar shunday bo'lsa, chizganimizni butunlay o'chirib tashlaymiz.

Bobning boshida juda ko'p kod paydo bo'lgan bo'lsa-da, uni tahlil qilgandan so'ng, dasturlash ko'rinadigan darajada murakkab emasligi aniq bo'ladi. Shunday qilib, kod hajmidan qo'rqishning ma'nosi yo'q. Biz ushbu rasm chizish taxtasini natijasini ko'rib chiqib, bobni yakunlaymiz.


Pygame Zero Game