10. Udalosti v grafickej ploche

V programoch môžeme využívat’ tzv. udalosti, ktoré vznikajú v bežiacej grafickej aplikácii bud’ aktivitami používatel’a (klikanie myšou, stláčanie klávesov) alebo operačného systému (tikanieˇ časovača). Na úvod si pripo meňme, čo už vieme o grafickej ploche. Pomocou metód grafickej plochyˇ Canvas (definovanej v module tkinter) kreslíme grafické objekty:

* **canvas.create\_line() - kreslí úsecku alebo krivku z nadväzujúcich úseciekˇ**
* **canvas.create\_oval() - kreslí elipsu**
* **canvas.create\_rectangle() - kreslí obd´lžnik**
* **canvas.create\_text() - vypíše text**
* **canvas.create\_polygon() - kreslí vyfarbený útvar zadaný bodmi na obvode**
* **canvas.create\_image() - kreslí obrázok (precítaný zo súboru .gif alebo .png)ˇ**

**Dalšie pomocné metódy manipulujú s už nakreslenými objektami:ˇ**

* **canvas.delete() - zruší objekt**
* **canvas.move() - posunie objekt**
* **canvas.coords() - zmení súradnice objektu**
* **canvas.itemconfig() - zmení d’alšie parametre objektu (napr. farba, hrúbka, text, obrázok, ...)**

**Dalšie metódy umožˇ nujú postupne zobrazovat’ vytváranú kresbu:ˇ**

* **canvas.update() - zobrazí nové zmeny v grafickej ploche**
* **canvas.after() - pozdrží beh programu o zadaný pocet milisekúndˇ**

***Udalost’***

***Udalost’ou voláme akciu, ktorá vznikne mimo behu programu a program môže na túto situáciu reagovat’. Najcastejšie sú to udalosti od pohybu a klikania myši, od stlácania klávesov, odˇ casovaca (vnútorných hodín OS), od rôznychˇ zariadení, ... V našom programe potom môžeme nastavit’, co sa má udiat’ pri ktorej udalosti. Tomuto sa zvykne hovorit’ udalost’ami riadené programovanie (event-driven programming).***

Naučíme sa, ako v našich grafických programoch reagovat’ na udalosti od myši a klávesnice.ˇ

Aby grafická plocha reagovala na klikania myšou, musíme ju zviazat’ (bind) s príslušnou udalost’ou (event).

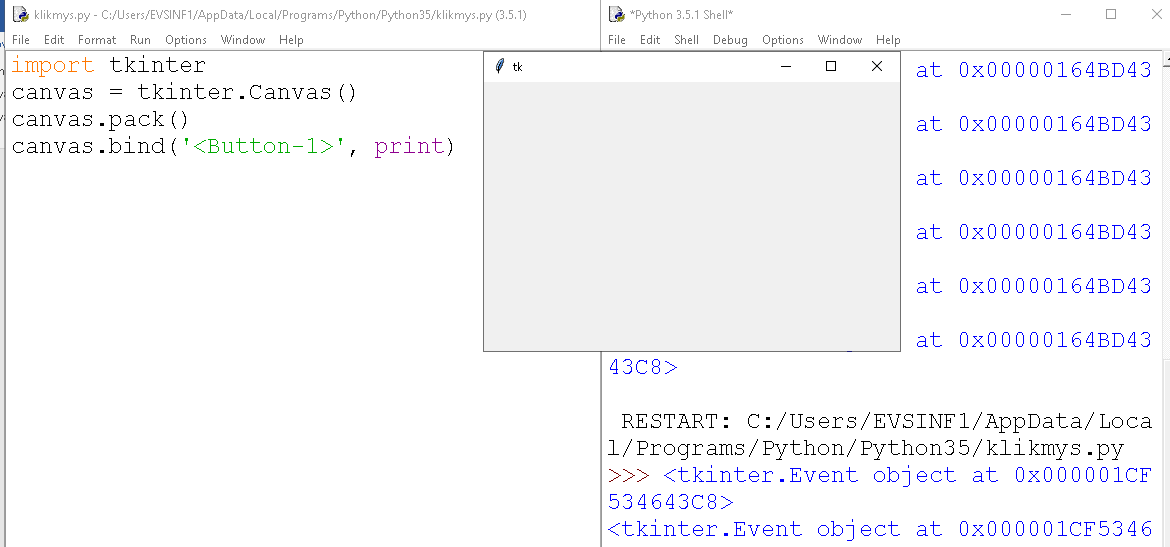
|  |
| --- |
| ***metóda bind()***  ***Táto metóda grafickej plochy slúži na zviazanie niektorej konkrétnej udalosti s nejakou funkciou, ktorá sa bude v programe starat’ o spracovanie tejto udalosti. Jej formát je:*** |
| *canvas.bind(meno\_udalosti, funkcia)* |
| ***kde meno\_udalosti je znakový ret’azec s popisom udalosti (napr. pre kliknutie tlačidlom myši) aˇ funkcia je referencia na funkciu, ktorá by sa mala spustit’ pri vzniku tejto udalosti. Táto funkcia, musí byt’ definovaná s práve jedným parametrom, v ktorom nám systém prezradí detaily vzniknutej udalosti.*** |

**10.1 Klikanie a t’ahanie myšou**

Ukážeme tieto tri „myšacie“ udalosti:

* kliknutie (zatlačenie tlačidla myši) - ret’azecˇ '<Button-1>' - 1 označuje l’avé tlacidlo myši,2 by tu znamenala stredné tlačidlo, 3 pravé tlačidlo
* t’ahanie (posúvanie myšou so zatlačeným tlačidlom) - ret’azecˇ '<B1-Motion>'
* pustenie myši - ret’azec '<ButtonRelease-1>'

**10.1.1 Klikanie myšou**

Kliknutie myšou do grafickej plochy vyvolá udalost’ s menom '<Button-1>'. Ukážme ako vyzerá samotné zviazanie funkcie:

**import tkinter**

canvas = tkinter.Canvas() canvas.pack()

canvas.bind('<Button-1>', print)

Druhý parameter metódy bind() musí byt’ referencia na funkciu, ale nie hocijakú, na funkciu, ktorá má jeden parameter. Tu sme použili štandardnú funkciu print a ked’že do bind() treba poslat’ referenciu na túto funkciu, nesmieme za identifikátor print písat’ zátvorky ( ).

Ked’ teraz tento malý testovací program spustíte, objaví sa prázdne grafické okno a program čaká, čo sa bude diat’ (Ak budete tento program spúšt’at’ mimo IDLE, nezabudnite na záver pripísat’ volanie mainloop(), napr. canvas. mainloop() ). Ked’že sme grafickej ploche udalost’ kliknutie myšou zviazali s funkciou print(), každé kliknutie do plochy automaticky vyvolá túto funkciu. Pri klikaní dostávame nejaký takýto výpis (pri každom kliknutí do plochy sa vypíše jeden riadok):

<ButtonPress event state=Mod1 num=1 x=194 y=117>

<ButtonPress event state=Mod1 num=1 x=194 y=173>

<ButtonPress event state=Mod1 num=1 x=328 y=31>

<ButtonPress event state=Mod1 num=1 x=17 y=9>

Zviazanie udalosti s nejakou funkciou teda znamená, že každé vyvolanie udalosti (kliknutie l’avým tlačidlom myši do grafickej plochy) automaticky zavolá zviazanú funkciu. To, co nám pritom vypisujeˇ print(), je ten jeden parameter, ktorý tkinter posiela pri každom jeho zavolaní.

Vytvorme si teraz vlastnú funkciu, ktorú zviažeme s udalost’ou kliknutia:

|  |
| --- |
| **import tkinter**  canvas = tkinter.Canvas() canvas.pack()  **def** klik(parameter):  print('klik')  canvas.bind('<Button-1>', klik) |

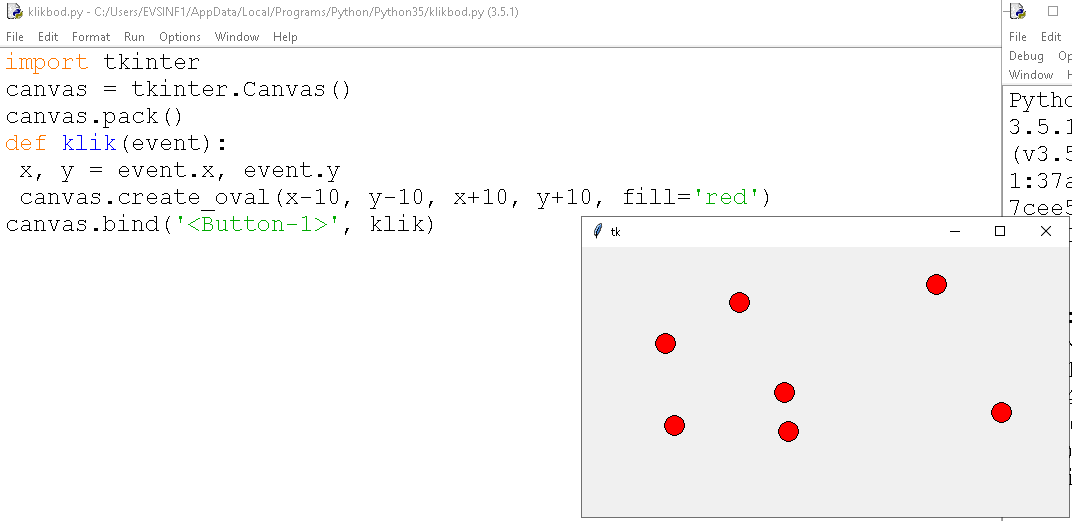
Vytvorili sme funkciu klik(), ktorá sa bude automaticky volat’ pri každom kliknutí do plochy. Nezabudli sme do hlavicky funkcie pridat’ jeden formálny parameter, inak by Python pri vzniku udalosti protestoval, že našu funkciuˇ klik() chcel zavolat’ s jedným parametrom a my sme ho nezadeklarovali. Ak by sme tento program spustili, pri každom kliknutí do plochy by sa do textovej plochy shellu mal vypísat’ text 'klik'.

Teraz k samotnému parametru v našej funkcii klik(): tento parameter slúži na to, aby nám tkinter mohol nejakým spôsobom posielat’ informácie o udalosti. My vieme, že funkcia klik() sa zavolá vždy, ked’ sa niekam klikne, ale nevieme, kde presne do plochy sa kliklo. Práve na toto slúži tento parameter: z neho vieme vytiahnut’, napr. x-ovú a y-ovú súradnicu kliknutého miesta. V nasledovnom príklade vidíme, ako sa to robí. Ešte sme tento parameter premenovali na event (udalost’ po anglicky), aby sme lepšie rozlíšili to, že s týmto parametrom prišla udalost’. Preto tieto súradnice získame ako event.x a event.y:

|  |
| --- |
| **import tkinter**  canvas = tkinter.Canvas() canvas.pack()  **def** klik(event):  print('klik', event.x, event.y)  canvas.bind('<Button-1>', klik) |

V tomto programe sa pri každom kliknutí vypíšu do shellu aj súradnice kliknutého miesta.

V d’alšom príklade ukážeme, ako využijeme súradnice kliknutého bodu v ploche:

****

|  |
| --- |
| **import tkinter**  canvas = tkinter.Canvas() canvas.pack()  **def** klik(event):  x, y = event.x, event.y canvas.create\_oval(x-10, y-10, x+10, y+10, fill='red')  canvas.bind('<Button-1>', klik) |

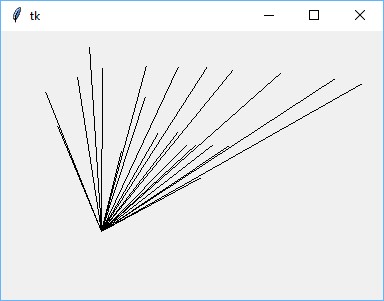
**10.1. Klikanie a t’ahanie myšou**

Teraz sa pri kliknutí nakreslí cervený kruh a využijú sa pritom súradnice kliknutého miesta: stred kruhu je kliknutéˇ miesto.

Akcia, ktorá sa vykoná pri kliknutí môže byt’ vel’mi jednoduchá, napr. spájanie kliknutého bodu s nejakým bodom grafickej plochy:

|  |
| --- |
| **import tkinter**  canvas = tkinter.Canvas() canvas.pack()  **def** klik(event):  canvas.create\_line(100, 200, event.x, event.y)  canvas.bind('<Button-1>', klik) |

Napr.



Ale môžu sa nakreslit’ aj komplexnejšie kresby, napr. 10 sústredných farebných kruhov:

|  |
| --- |
| **import tkinter import random**  canvas = tkinter.Canvas() canvas.pack()  **def** klik(event):  x, y = event.x, event.y **for** r **in** range(50, 0, -5):  farba = f'#{random.randrange(256\*\*3):06x}' |
| canvas.create\_oval(x-r, y-r, x+r, y+r, fill=farba)  canvas.bind('<Button-1>', klik) |

Napr.



Vrát’me sa k príkladu, v ktorom sme kreslili malé krúžky:

|  |
| --- |
| **import tkinter**  canvas = tkinter.Canvas() canvas.pack()  **def** klik(event):  x, y = event.x, event.y canvas.create\_oval(x-5, y-5, x+5, y+5, fill='red')  canvas.bind('<Button-1>', klik) |

Do tohto programu chceme pridat’ takéto správanie: tieto kliknuté body (cervené krúžky) sa budú postupne spájat’ˇ úseckami (zrejme sa bude úseˇ cka kreslit’ až od druhého kliknutia). Pridáme dve globálne premennéˇ xx a yy, v ktorých si budeme pamätat’ predchádzajúci kliknutý bod. Pred prvým kliknutím sme do xx priradili None, co bude oznaˇ covat’,ˇ že predchádzajúci vrchol ešte nebol:

**import tkinter**

canvas = tkinter.Canvas() canvas.pack()

|  |
| --- |
| xx = yy = **None**  **def** klik(event):  x, y = event.x, event.y canvas.create\_oval(x-5, y-5, x+5, y+5, fill='red') **if** xx != **None**: canvas.create\_line(xx, yy, x, y) xx, yy = x, y  canvas.bind('<Button-1>', klik) |

Žial’ to nefunguje: po spustení a kliknutí sa dozvieme:

|  |
| --- |
| ...  File ..., line 11, **in** klik **if** xx != **None**:  UnboundLocalError: local variable 'xx' referenced before assignment |

Problémom sú tu globálne premenné. Používat’ globálne premenné vo vnútri funkcii môžeme, len dovtedy, kým ich nemeníme. Prirad’ovací príkaz vo funkcii totiž znamená, že vytvárame novú lokálnu premennú (v mennom priestore funkcie klik()). Takže Python to v tejto funkcii pochopil takto: do premenných xx a yy sa vo vnútri funkcie prirad’uje nejaká hodnota, takže obe sú lokálne premenné. Ked’ ale príde vykonávanie funkcie na podmienený príkaz if xx != None:, Python už vie, že xx je lokálna premenná, ktorá nemá zatial’ priradenú žiadnu hodnotu. A preto nám oznámil túto chybovú správu: UnboundLocalError: local variable 'xx' referenced before assignment.

Takže s globálnymi premennými vo funkcii sa bude musiet’ pracovat’ nejako inak. Zrejme, kým do takejto premennej nepotrebujeme vo funkcii nic priradzovat’, iba ju používat’, problémy nie sú. Problém nastáva vtedy, ked’ chcemeˇ (pomocou prirad’ovacieho príkazu) menit’ obsah globálnej premennej.

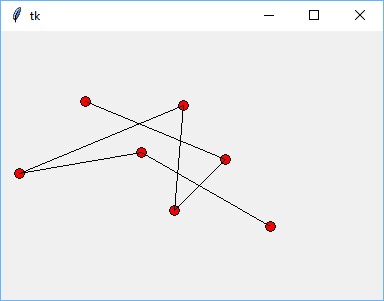
Toto nám pomôže vyriešit’ nový príkaz global:

|  |
| --- |
| príkaz **global** príkaz má tvar: |
| **global** premenná **global** premenná, premenná, premenná, ... |
| Príkaz sa používa vo funkcii vtedy, ked’ v nej chceme pracovat’ s globálnou premennou (alebo aj s viac premennými), ale nechceme, aby ju Python vytvoril aj v lokálnom mennom priestore, ale ponechal len v globálnom. |

Po doplnení tohto príkazu do predchádzajúceho príkladu všetko funguje tak, ako má:

|  |
| --- |
| **import tkinter**  canvas = tkinter.Canvas() canvas.pack()  xx = yy = -1  **def** klik(event):  **global** xx, yy x, y = event.x, event.y |
| canvas.create\_oval(x-5, y-5, x+5, y+5, fill='red') **if** xx >= 0:  canvas.create\_line(xx, yy, x, y) xx, yy = x, y  canvas.bind('<Button-1>', klik) |

Po spustení dostávame takýto obrázok:



Nebezpečé:ˇ Príkaz global umožnuje modifikovat’ globálne premenné vo funkciách, teda vlastne robit’ vedl’ajší účinokˇ (*side effect*) na globálnych premenných. Toto je ale vel’mi nesprávny spôsob programovania (*bad programming practice*) a väcšinou svedčí o programátorovi , amatérovi.ˇ

Kým sa nenaučíme, ako to obíst’, budeme to používat’, vel’mi opatrne. Neskôr to využijeme vel’mi výnimočne,ˇ najmä pri ladení. Správne sa takéto problémy riešia najčastejšie definovaním vlastných tried a použitím atribútov tried.ˇ

**10.1.2 Tahanie myšou**

Obsluha udalosti t’ahanie myšou (pohyb myši so zatlačeným tlačidlom) je vel’mi podobná klikaniu. Udalost’ má menoˇ '<B1-Motion>'. Pozrime, čo sa zmení, ked’ kliknutieˇ '<Button-1>' nahradíme t’ahaním '<B1-Motion>':

**import tkinter**

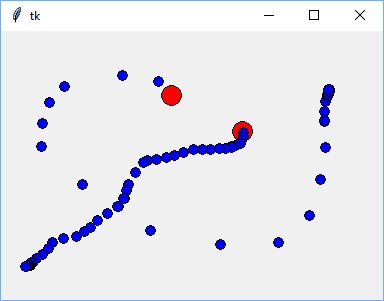
canvas = tkinter.Canvas() canvas.pack()

|  |
| --- |
| **def** klik(event):  x, y = event.x, event.y canvas.create\_oval(x-10, y-10, x+10, y+10, fill='red')  canvas.bind('<B1-Motion>', klik) *# '<B1-Motion>' namiesto '<Button-1>'* |

Funguje to vel’mi dobre: pri t’ahaní sa na pozícii myši kreslia červené kruhy. Pri pomalom t’ahaní sú kruhy nakreslenéˇ vel’mi nahusto. Často budeme v našich programoch spracovávat’ obe udalosti: kliknutie aj t’ahanie. Niekedy je to tá istá funkcia, inokedy sú rôzne a preto je dobre ich pomenovat’ zodpovedajúcimi názvami, napr.

|  |
| --- |
| **import tkinter**  canvas = tkinter.Canvas() canvas.pack()  **def** klik(event):  x, y = event.x, event.y canvas.create\_oval(x-10, y-10, x+10, y+10, fill='red')  **def** tahanie(event):  x, y = event.x, event.y canvas.create\_oval(x-5, y-5, x+5, y+5, fill='blue')  canvas.bind('<Button-1>', klik) canvas.bind('<B1-Motion>', tahanie) |

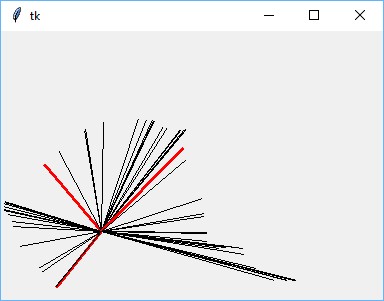
Pri kliknutí (ešte bez t’ahania) sa nakreslí červený kruh, pri t’ahaní sa kreslia už len malé modré kruhy.ˇ



Na podobnom princípe môžeme upravit’ aj kreslenie lúčov z bodu (100, 200) do pozície myši:ˇ

|  |
| --- |
| **import tkinter**  canvas = tkinter.Canvas() canvas.pack()  **def** klik(event): canvas.create\_line(100, 200, event.x, event.y, fill='red', width=3)  **def** tahanie(event):  canvas.create\_line(100, 200, event.x, event.y)  canvas.bind('<Button-1>', klik) canvas.bind('<B1-Motion>', tahanie) |

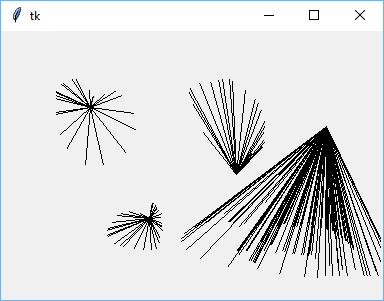
Pri kliknutí sa nakreslí červená úsečcka, pri t’ahaní sa kreslia už len čierne.ˇ



Alebo malou zmenou kliknutím definujeme pozíciu (globálny bod (xx, yy)), z ktorého sa budú kreslit’ lúče:ˇ

|  |
| --- |
| **import tkinter**  canvas = tkinter.Canvas() canvas.pack()  **def** klik(event):  **global** xx, yy xx, yy = event.x, event.y  **def** tahanie(event): |
| canvas.create\_line(xx, yy, event.x, event.y)  canvas.bind('<Button-1>', klik) canvas.bind('<B1-Motion>', tahanie) |

Potom dostávame takúto kresbu:



Ďalej nadviažeme na program, v ktorom sme postupne spájali kliknuté body. Pri tomto programe sme využili nový príkaz global, aby sme sa dostali ku globálnym premenným. Tento nie najvhodnejší príkaz môžeme obíst’, ked’ využijeme menitel’ný (mutable) typ zoznam.

Budeme t’ahat’ (t’ahaním myši) jednu dlhú lomenú čiaru, pričom si budeme ukladat’ súradnice prijaté z udalosti doˇ zoznamu čísel:ˇ

|  |
| --- |
| **import tkinter**  canvas = tkinter.Canvas() canvas.pack()  zoznam = [] ciara = canvas.create\_line(0, 0, 0, 0)  **def** klik(event): zoznam[:] = [event.x, event.y]  **def** tahanie(event):  zoznam.extend([event.x, event.y]) canvas.coords(ciara, zoznam) |

canvas.bind('<Button-1>', klik) canvas.bind('<B1-Motion>', tahanie)

Všimnite si, že vo funkciách používame 3 globálne premenné:

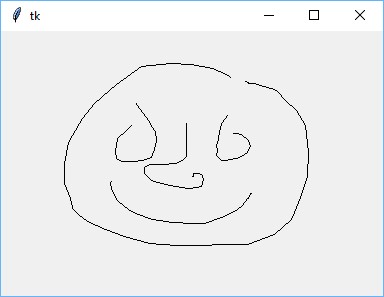
* canvas - referencia na grafickú plochu
* ciara - identifikátor objektu čiara, potrebujeme ho pre neskoršie menenie postupnosti súradníc príkazomˇ coords()
* zoznam - zoznam súradníc je menitel’ný objekt, teda môžeme menit’ obsah zoznamu bez toho, aby sme do premennej zoznam prirad’ovali (prirad’ujeme bud’ do rezu alebo voláme metódu extend(), t. j. prilep nejakú postupnost’ na koniec zoznamu)

– modifikovanie zoznamu vo funkcii, pricom zoznam nie je parametrom funkcie, tiež nie je najvhodnejším spôsobom programovania, tiež je to nevhodný vedl’ajší účinokˇ podobne ako príkaz global, zatial’ to inak robit’ nevieme, tak je to dočasne akceptovatel’néˇ

Ťahanie čiary v predchádzajúcom príklade žial’ kreslí jedinú čiaru: každé d’alšie kliknutie a t’ahanie začne kreslit’ˇ novú čiaru, pričom stará čiara zmizne. Vyriešime to tak, že pri kliknutí začneme kreslit’ novúčiaru a tú starú necháme tak:

|  |
| --- |
| **import tkinter**  canvas = tkinter.Canvas() canvas.pack()  zoznam = []  **def** klik(event):  **global** ciara zoznam[:] = [event.x, event.y] ciara = canvas.create\_line(0, 0, 0, 0)  **def** tahanie(event):  zoznam.extend([event.x, event.y]) canvas.coords(ciara, zoznam)  canvas.bind('<Button-1>', klik) canvas.bind('<B1-Motion>', tahanie) |

Poexperimentujeme:



Malou zmenou dosiahneme vel’mi zaujímavý efekt. Vyskúšajte a poštudujte:

|  |
| --- |
| **import tkinter import random**  canvas = tkinter.Canvas() canvas.pack()  zoznam = []  **def** klik(event):  **global** poly zoznam[:] = [event.x, event.y] farba = f'#{random.randrange(256\*\*3):06x}' poly = canvas.create\_polygon(0, 0, 0, 0, fill=farba)  **def** tahanie(event):  zoznam.extend([event.x, event.y]) canvas.coords(poly, zoznam)  canvas.bind('<Button-1>', klik) canvas.bind('<B1-Motion>', tahanie) |

