Dnes bude

dokončení semestru

- Událostmi řízené programování (kdy jen čekáme, až přijde událost, na kterou nastavíme ovladač),
- Tkinter (jakožto knihovna umožňující tvorbu formulářových aplikací),
- demonstrace knihovny Tkinter k implementaci hry (o štrůdlu),
- ukázka, co všechno lze v Pythonu programovat.

Událostmi řízené programování

funguje úplně jinak, než jsme zvyklí

- Důkladněji bude v létě v C#, sledujte podobnosti a rozdíly v implementaci.
- My jsme zvyklí program organizovat od začátku do konce (co přesně kdy se má stát), jenže
- řada aplikací vyžaduje vyplnění údajů, což by se takto dělalo těžko,
- naopak samo vyplňování údajů nás nezajímá, my chceme začít pracovat, až jsou data pohromadě.
- O běh programu se tedy chová prostředí, nás vzbudí, až nastane nějaká událost.
- Události: kliknutí myši, příjezd myši, odjezd, kliknutí pravým tlačítkem, dvojklik,...

Událostmi řízené programování

z technického hlediska

- Obstaráme si formulář,
- tomu (a jeho elementům) nastavíme funkce jako ovladače událostí,
- řekneme "plav"!

Tkinter

Je knihovna implementující událostmi řízené programování

- `import tkinter`
- nebo lépe `from tkinter import *`
- Obsahuje formulář, tlačítka, textová políčka editovatelná, - a needitovatelná,...
- Použijeme ji tak, že postavíme formulář objekt třídy Tk, pomocí atributů nastavíme jeho vzhled, nastavíme ovladače,
- postavíme ovládací prvky, atributy jim nastavíme vzhled, nastavíme ovladače, naplácáme prvky na formulář,
- zavoláme formuláři metodu `mainloop`

Prvky

které se mezi verzemi nepochybně mění

- Label (text k výpisu)
- Button (tlačítko)
- Entry (textové okénko)
- Checkbutton, Scale, Text
- LabelFrame, Canvas, Menu,...

Tkinter

příklad

```
from tkinter import *
formular=Tk() # udelej novy formular
formular.geometry("800x600+50+10")# velikost/umístění
hlaseni=Label(formular,text="Nazdarek!")
hlaseni.pack()# Prilep hlaseni na formular
formular.mainloop() # a jedeme!
#Překreslení: formular.update()
```

Rozmisťování: pack

a související atributy

- `pack` dá prvek na formulář dle defaultního zarovnání (na střed) co nejvíce k defaultní hraně formuláře (horní).
- Implicitní atributy:
  - `fill` vyplnit `X`, `Y`, `BOTH`, potažmo `tkinter.X`, `Y`, resp. `BOTH`.
  - `expand` natáhnout 0 – ne, jinak – ano (na celou dostupnou část okna)
  - `side` kam přitáhnout `TOP`, `LEFT`, `BOTTOM`, `RIGHT`.

Rozmisťování

to skutečné

- `grid` vytvoří tabulku a prvky strká do buněk.
- Implicitní argumenty:
  - `row, column` berou celá čísla (kam prvek dát),
  - `rowspan, columnspan` kolik buněk pokrýt,
  - ...
- `place` dej na určené místo.
  - `x, y` – kam,
  - `width, height` velikost prvku,
  - `relx, rely, relheight, relwidth` – relativní jednotky (vzhledem k velikosti okna, argumenty jsou necelé).

Společné metody

prvků, se kterými budeme pracovat

- `config(parametr=hodnota)` – nastaví daný atribut na dotyčnou hodnotu.
- Příklad: `l.config(text="Popiska")`.
- Bez parametru vrátí slovník (asociativní pole) všech parametrů (s nastavením).
- `cget("parametr")` – zjistí hodnotu atributu jako string.
- Zbytek nám prozradí `config()`.

Společné atributy

prvků

- zpracování událostí: `mainloop, quit, wait_variable(var), wait_visibility(window), wait_window(window), update, update_idletasks,...` mainloop lze opustit metodou quit, likvidaci okna provede destroy.
- funkce s událostmi: `bind (udalost, funkce), unbind(udalost),...` nabinduje funkci na událost (resp. odbinduje).
- časovače: `after(cas, funkce), after_idle(funkce), after_cancel(id)`
- správa oken: `lift, lower` – posunou okno nahoru/dolů.

Udalosti

a názory

- Události se `bind/unbind` předávají jako string.
- Možné události související s myší: `<Button-1>`, `<B1-Motion>`, `<ButtonRelease-1>`, `<Double-Button-1>`, `<Enter>`, `<Leave>`,
- s klávesnicí: `<Key>`, `<FocusIn>`, `<FocusOut>`, konkrétní klávesy `<Return>`, `<Control_L>`, `<Alt_L>`, `<Escape>`, `<L>`, ...
- Událost související s myší dostává objekt (1. argument) popisující souřadnice (x, y), události s klávesnicí dostávají objekt s atributem char (který událost způsobil).

Možné prvky

jak se do nich dostat už víme

- `Button, Canvas, Checkbutton, Entry, Frame, Label, LabelFrame, Listbox, Menu, Menubutton, Message, OptionMenu, PanedWindow, Radiobutton, Scale, Scrollbar, Spinbox, Text, Toplevel`
- Dialogové okno:
  `tkinter.messagebox.showwarning('titulek','hlaseni')`

Význačné prvky – Button

a ukázka práce s nimi

```
from tkinter import *
import tkinter.messagebox
def prikliku():
    print("Klikli jsme!")
def prikliku2(eventa):
    tkinter.messagebox.showwarning('Ha!')
f=Tk()
b=Button(f,text="Popiska",command=prikliku)
b.bind('<Button-1>',prikliku2)
b.pack()
f.mainloop()
```

Checkbutton

a. k. a. Checkbox

```
from Tkinter import *
f = Tk()
pchbx = IntVar()
chbx = Checkbutton(f, text="Vyber", variable=pchbx,
onvalue=1, offvalue=0)
chbx.pack()
mainloop()
```

Při kliknutí nastavuje proměnnou `pchbx`, implicitní argumenty `onvalue`, `offvalue` – hodnoty je-li vybráno resp. ne (přiřadí 1 resp. 0).

Textové pole

je pořád stejné – ale chová se zase jinak...

```
t=Text(f)
t.pack()
```

Obsah textového pole najdeme v:

```
t.get(zacatek,konec)
```

Jenže pozor, lze použít konstanty `END`, `CURRENT`,.. – CURRENT je znak nejblíže k myši, řádek/sloupec: `'radek.sloupec'` , např. `1.0`),...

Radiobutton

aneb výběry z možností jsou podobné jako Checkbuttony

```
...
pohlavi=IntVar()
r1=Radiobutton(f, text="Muz", variable=pohlavi,
value=1)
r1.pack()
r2=Radiobutton(f,text="Zena", variable=pohlavi,
value=0)
r2.pack()
#krom IntVar je i StringVar (na stringy)
#hodnotu zjistime pohlavi.get()
```

Canvas

slouží k malování obrázků

```
c=Canvas(f,width=100,height=100)
c.pack()
c.create_line(0,0,100,100,fill="red", dash=(4,4))
c.create_rectangle(25,25,75,75,fill="blue")
```

Balíčky

jsou dalším prostředkem dělení programu

- Kód umíme dělit do funkcí, souborů, modulů.
- Moduly umíme importovat (import jmeno).
- Balíček může sestávat z několika souborů (může být obsažnější).
- V adresáři vytvoříme soubor `__init__.py` (klidně prázdný). Ten je inicializátorem balíčku.
- V něm můžeme definovat některé položky a také co se nahraje, řekneme-li `from balicek import *`:
- `__all__=[ "pokus", "druhy_pokus"]` – v tomto případě importujeme tyto dvě položky. Ostatní ne.

Příklad

balíčku p

__init__.py:

```
__all__=[ "pokus", "druhy_pokus"]
```

pokus.py:

```
def f():
    print("Nazdarek")
```

druhy_pokus.py:

```
def g():
    print("Funkce g")
```

treti_pokus.py:

```
def skrytejsi_funkce():
    print("Sem se dostat da trochu praci.")
```

Příklad

balíčku p

program.py

```
from p import *
pokus.f()
druhy_pokus.g()
#treti_pokus.skrytejsi_funkce() by spadlo
import p.treti_pokus
p.treti_pokus.skrytejsi_funkce()
# Ted uz to projde.
```

Hry s ohodnocením

byly

Definition

Hra s ohodnocením je taková hra, kdy cílové stavy jsou ohodnoceny číslem. Jeden hráč se pokouší výsledek maximalizovat, druhý minimalizovat.

Definition

Hra s nulovým součtem je taková hra, ve které zisk jednoho hráče je roven ztrátě druhého hráče.

Některé hry

- **Výlet s přítelkyní do New Yorku:** Chceme navštívit co nejvíce hostinců a technických pamětihodností, přítelkyně chce vidět co nejvíce muzeí a kadeřnictví. Dohodnete se tudíž, že se budete střídat v rozhodování kam jít na jednotlivých křižovatkách.
- **Traverzování po matici:** První hráč mění sloupce, druhý hráč mění sloupce. Začínáme v prvním řádku, první hráč vybere sloupec v prvním řádku a hodnotu na jeho pozici získává. Druhý hráč vybere řádek a získává hodnotu z vybraného řádku ve sloupci vybraném prvním hráčem. Takto se střídají (předem známou dobu).
- **Společná otázka:** Jak hrát?

Algoritmus MINIMAX

- Algoritmus lze použít pro hry s ohodnocením.
- Postavíme strom hry.
- Začneme od koncových vrcholů.
- Hodnota podstromu je minimum resp. maximum z hodnot synů (podle toho, zda hraje minimalizující nebo maximalizující hráč).

Algoritmus NEGAMAX

- Varianta algoritmu MINIMAX pro hry s nulovým součtem:

$$\text{ind} \max_{i \in S} -f(i) = \text{ind} \min_{i \in S} f(i).$$

- Jde vlastně o totéž, je ovšem jednodušší na naprogramování.

Demonstrace Negamaxu

na hře o štrudlu

- Štrůdl bereme z kraje (levého nebo pravého), chceme ho sníst co nejvíce.
- Vyzkoušíme všechny možnosti a podíváme se, co vyjde lépe.
- Předvedeno přímo na místě.

Heuristiky

- Obvykle se pokoušíme neprohledávat zbytečně všechno, pokud najdeme jednu možnost výhry, nemusíme hledat i všechny ostatní.
- α-β-prořezávání: Umíme-li v nějakém synu S vyhrát aspoň α a najdeme v některém následujícím synu T, že protihráč nás umí dotlačit na méně, nemá smysl vrchol T dále zkoumat.
- Pro opačný případ se používá β: Pokud nás nepřítel umí zatlačit na nejvýš β a v jiném synu mu utečeme přes, nemá smysl ten druhý syn zkoumat dále.

Reálné hry

Šachy, dáma, halma, mlýn...

- Můžeme postavit strom hry, ten je ale příliš velký.
- Nasadíme proto všelijaké heuristiky. Ty dosavadní ale stejně daleko nevedou.
- Statická ohodnocovací funkce: Funkce, která se pokouší odhadnout, zda je pozice perspektivní (dobrá) nebo ne.
- Prohledáváme strom hry jen po nějakou dobu (do nějaké hloubky). Na nalezené (neterminální) pozice nasadíme statickou ohodnocovací funkci.
- U šachů například můžeme počítat materiální převahu a body za ohrožené figurky (Colossus na Atari kolem roku 1985).

Horizont, statická ohodnocovací funkce

- Horizont stanoví, do jaké hloubky graf hry (zpravidla realizovaný stromem) prohledáváme.
- Statická ohodnocovací funkce nastoupí, pokud se dostaneme na horizont.
- Dalšího zrychlení lze (zkusit) dosáhnout tak, že napřed prohledáváme perspektivní vrcholy (kde statická ohodnocovací funkce dává lepší výsledky).
- Jde ovšem jen o heuristiku, která někdy funguje, jindy se dostane do problémů!

Komentář k reálným hrám

- Typicky stavíme strom hry. Graf stavíme až v závěrečné fázi hry, do té doby budujeme strom (a ignorujeme možnost, že některé stavy se už vyskytly).
- Heuristické algoritmy lze pojmout dvěma způsoby:
- Metoda, která se pokouší najít optimum co nejrychleji,
- metoda, jak najít aspoň nějaké (suboptimální) řešení.
- Zatím bylo to první. Jak použít heuristiku k nalezení suboptimálního řešení?

$\alpha - \beta$ prořezávání jako heuristika

- Jsou dva možné způsoby:
- Metoda okénka: Stanovíme krajní hodnoty α a β a výsledky ležící mimo tento interval ořežeme.
- Kaskádní varianta – strom rozšiřujeme po hladinách (protože pokud bychom ho stavěli prohledáváním do hloubky, prozkoumáme typicky nezajímavé větve a zajímavé tahy nám uniknou).

Minimaxové věty

- Vraťme se k maticovým hrám (stylu Al-Capone a Babinský na sebe udávají).
- Má smysl uvažovat nejen o deterministické variantě (tzv. čistá strategie – obzvlášť pokud hráči táhnou nezávisle, tedy na sebe nevidí), ale má smysl definovat pravděpodobnostní distribuci (a podle té hrát). Tomu říkáme mixovaná strategie.

Theorem

Pro každou kombinatorickou hru s nulovým součtem s konečnými strategiemi existuje hodnota V a mixovaná strategie pro každého hráče taková, že:

- *Pokud podle své strategie hraje druhý hráč, první hráč nemůže vyhrát více než V.*
- *Pokud podle své strategie hraje první hráč, druhý hráč nemůže vyhrát více než $-V$.*

Nash equilibrium

Definition

Nashovou rovhováhou nazveme sadu mixovaných strategií (pro každého hráče jednu) v konečných hrách aspoň dvou nespolupracujících hráčů, kde žádný z hráčů si nemůže pomoci tím, že strategii změní.

Theorem (J. Nash)

Pro každou hra n hráčů, kde každý hráč má konečně možných strategií, existují strategie určující Nashovu rovhováhu.

Programování robota

ze kterého tu drze promítám slidy a na kterém hrajeme hru o štrůdlu

- Na Raspberry pi lze instalovat Linux (nejlépe Raspbian).
- Jak se pohybovat v UNIXu by mělo být předneseno na speciálním kurzu,
- Linuxy bývají vybavené interpretem Pythonu.
- Raspberry pi má navíc GPIO, k ovládání periferií.
- Do Pythonu je v Raspbianu vyveden balík `RPi` vybavený balíkem `GPIO`, tedy jen řekneme `import RPi.GPIO as GPIO` a máme další knihovnu (ve které jsou funkce tentokrát pro vypínání, pouštění a měření proudu).
- Předvést a okomentovat `robot_tk.py`,
- a následně další skripty (`stop.py`, `robot_manual.py`, `robot_ir.py`, `robot_sonar.py`)

Konec

Děkuji za pozornost...