INFORMATIKA

Stavíme se Stavitelkou

ŠÁRKA GERGELITSOVÁ – TOMÁŠ HOLAN

Matematicko-fyzikální fakulta UK, Praha

Stavitelka je aplikace pro vytváření 3D konstrukcí pomocí jazyka Python. Tento text ukazuje její použití, myslíme si, že by mohl zajímat ty, kdo učí programování, i ty, kteří se učí programovat a chtějí programem vytvořit něco hezkého.

1. Aplikace

Stavitelka je webová aplikace na `https://stavitelka.geometry.cz/`, obsahuje *textové pole* pro vkládání programu, *tlačítko* `Postav` a *„obrázek“* – okno s odpovídající scénou. Používá se tak, že do textového pole napíšeme nebo vložíme program, stiskneme tlačítko a můžeme si prohlížet výsledek. Obrázek zobrazuje trojrozměrnou scénu, jejíž prohlížení můžeme ovládat pomocí myši.

Program

Scéna se popisuje pomocí programu v jazyku Python, ale k vytvoření jednoduché scény nemusíme umět programovat, ani nepotřebujeme žádné velké znalosti. Projdeme si je postupně.

Objekt

Scéna se skládá z objektů. Stavitelka umí pracovat se třemi typy objektů – kvádr, koule, a válec. Objekt vytvoříme tak, že zavoláme patřičnou funkci, takže když program bude obsahovat jediný příkaz

```
st.Koule()
```

uvidíme kouli, a podobně příkazem

```
st.Kvadr()
```

nebo

```
st.Valec()
```

vytvoříme kvádr nebo válec (obr. 1).

Obr. 1 Základní tvary s výchozími hodnotami parametrů

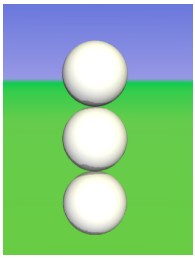
Všechny příkazy pro Stavitelku začínají písmeny „st.“. To zelené a modré na obrázku je něco jako tráva a obloha.

Vlastnosti a parametry

Každý vytvořený objekt má nějakou `polohu`, `zvětšení` a též `pootočení` neboli `rotaci`. Každý z těchto údajů je určen třemi čísly – trojrozměrná poloha (x, y, z), zvětšení v každém ze tří směrů (s_x, s_y, s_z) a rotace podle každé ze tří os (r_x, r_y, r_z). Když je neuvedeme, použijí se výchozí hodnoty. K tomu je dobré říci, že rozměry všech těchto základních objektů jsou **1**, s výjimkou výšky válce, která je **2**. Výchozí poloha každého objektu i výchozí pootočení jsou $(0, 0, 0)$ a výchozí zvětšení je $(1, 1, 1)$. Poloha objektu je poloha jeho středu, proto koule sahá od $-0{,}5$ do $+0{,}5$ ve všech směrech a válec v ose y sahá od -1 do $+1$. Takže třeba program

```
st.Koule()
st.Koule(y=1)
st.Koule(y=2)
```

vyrobí jednoduchého sněhuláka (obr. 2),

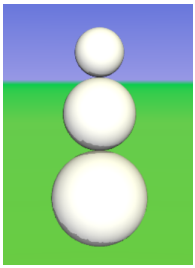


Obr. 2 Umístění objektu

ale pokud přidáme velikosti a náležitě upravíme polohu (souřadnice y),

```
st.Koule(sx=2, sy=2, sz=2)
st.Koule(sx=1.5, sy=1.5, sz=1.5, y=1.75)
st.Koule(y=3)
```

dostaneme lepšího sněhuláka (obr. 3).



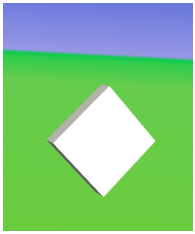
Obr. 3 . . . a změna velikosti

Parametry můžeme uvádět v jakémkoliv pořadí a když nějaký vynecháme, použije se jeho výchozí hodnota.

Ještě ukázka parametrů `zvětšení` a `rotace`: Třeba kvádr zmenšený ve směru osy z (to je osa zepředu dozadu) na čtvrtinu a otočený podle stejné osy o 45 stupňů

```
st.Kvadr(sz=1/4, rz=45)
```

vypadá takhle:



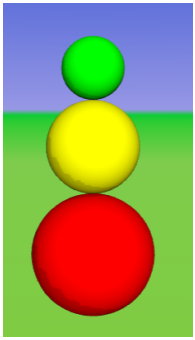
Obr. 4 Rotace

Barva

Mezi parametry je ještě jeden (ale už poslední!) – a to je `barva`. `Barva` se uvádí jako trojice čísel mezi 0 a 1 uvádějících intenzitu červené, zelené a modré, neboli RGB. Výchozí barva, jak jsme už viděli, je bílá.

```
st.Koule(sx=2, sy=2, sz=2, barva=(1, 0, 0))
st.Koule(sx=1.5, sy=1.5, sz=1.5, y=1.75, barva=(1, 1, 0))
st.Koule(y=3, barva=(0, 1, 0))
```

vypadá takhle:



Obr. 5 Změna barvy

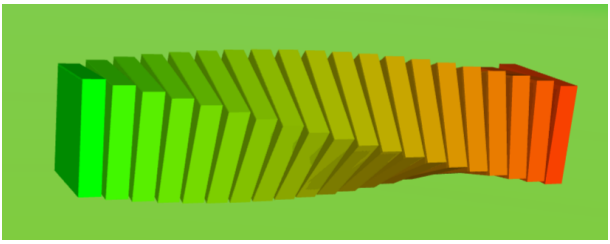
Tip: Pokud se vám dlouhé řádky nevejdou do textového pole, chyťte ho za pravý dolní roh a roztáhněte ho na potřebnou velikost.

Proč je to program

Víme tedy, že Stavitelka je aplikace, která z příkazů vyrobí trojrozměrnou scénu, kterou si můžeme prohlížet. Ale proč se ty příkazy zapisují jako program a proč v jazyku Python?

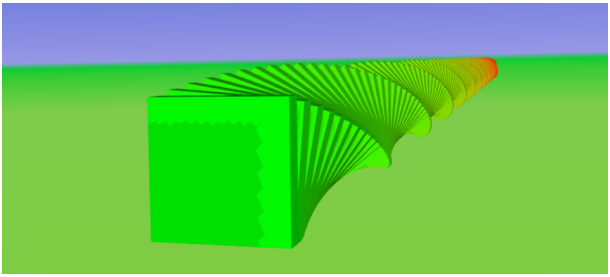
Odpověď zní: protože v programovacím jazyku dokážeme snadno zapsat opakování, podmínky, výpočty. Třeba když chceme vyrobit dvacet čtvercových desek – kvádrů, různě posunutých, pootočených a obarvených, nemusíme psát dvacet příkazů, ale použijeme cyklus:

```
N = 20
for i in range(N):
    st.Kvadr(sx=0.2, x=0.25*i, rx=5*i, barva=(i/N,1-i/N,0))
```



Obr. 6 Cyklem generovaná řada čtvercových desek

A to, že jsme číslo 20 schovali do proměnné N, nám dovolí přidáním jedné nuly vytvořit místo dvaceti desek dvě stovky desek.



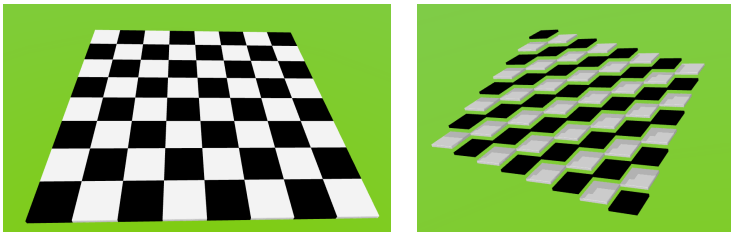
Obr. 7 . . . delší cyklus

Stejně snadno vytvoříme třeba nejen šachovnici:

```
for i in range(8):
    for j in range(8):
        st.Hranol(sy=0.1, x=i, z=j, barva=((i+j)%2,)*3)
```

ale i šachovnicové schodiště

```
...st.Hranol(sy=0.1, x=i, z=j, y=(i+j)/4, barva=((i+j)%2,)*3)
```



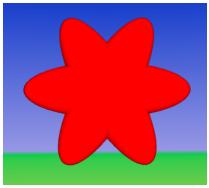
Obr. 8 Šachovnice (v kódu $y = 0$) a schodiště ($y = (i + j)/4$)

Díky tomu, že scénu zapisujeme jako program, a díky tomu, že Stavitelka umí vytvářet prostorové objekty, se vyhneme nutnosti studovat nějaký jazyk pro tvorbu 3D scény.

2. Tvary

Ale není všechno tak krásně jednoduché. Kdybychom chtěli vytvořit třeba něco, co by připomínalo květ, můžeme vzít kouli, v jedné ose, třeba z, ji zmenšit, aby byla skoro placatá, ve druhé ose, třeba (ve svislé ose) y ji zmenšit a pak ji vytvořit několikrát, vždy trochu pootočenou:

```
N = 3
for i in range(N):
    st.Koule(sz=0.2, sy=1/3, rz = i*360/N, barva=(1, 0, 0))
```



Obr. 9 Tři škálované a překrývající se koule

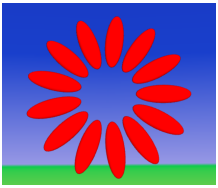
Jenže kdybychom chtěli třeba lichý počet okvětních lístků, tak to takhle nedokážeme a problém je v tom, že rotace otáčí každý objekt kolem jeho středu a to, co vidíme na obrázku jako 6 dílků, jsou ve skutečnosti jenom 3 dílky, které se překrývají.

Jak tedy rozmístit okvětní lístky kolem středu? Řešení spočívá v tom, co se ve Stavitelce jmenuje „tvar“, v možnosti vytvořit nový druh objektu, který bude obsahovat jiné objekty. A tyto jiné objekty v něm můžeme vhodně rozmístit, takže když se nově vytvořený tvar bude otáčet kolem svého středu souřadnic, tedy kolem své souřadnice $(0, 0, 0)$, budou se kolem tohoto bodu otáčet všechny objekty ve tvaru obsažené.

Ukažme si příklad, definice tvaru začíná `st.TVAR_zacatek_definice(jmeno)`, končí `st.TVAR_konec_definice()` a nový exemplář vytvoříme pomocí `st.TVAR(jmeno)`, kde můžeme zase uvést parametry stejně jako u kvádru, koule nebo válce:

```
st.TVAR_zacatek_definice("listek")
st.Koule(x=1, sz=0.2, sy=1/3, barva=(1, 0, 0))
st.TVAR_konec_definice()

N = 13
for i in range(N):
    st.TVAR("listek", rz=i*360/N)
```



Obr. 10 V novém tvaru je koule posunutá mimo střed, kolem něhož se otáčí

Definice tvaru může obsahovat další tvary, takže z lístků můžeme definovat tvar pro květinu a z květin třeba celý záhon, a protože máme k dispozici všechny nástroje jazyka Python, mohou mít jednotlivé květiny

třeba náhodně vybranou velikost nebo pootočení:

```
st.TVAR_zacatek_definice("listek")
st.Koule(x=1, sz=0.2, sy=1/3, barva=(1, 0, 0))
st.TVAR_konec_definice()

st.TVAR_zacatek_definice("kytka")
N = 13
for i in range(N):
    st.TVAR("listek", rz=i*360/N)
st.Koule(sz=0.2, barva=(1, 1, 0))
st.Valec(sx=0.2, sz=0.2, sy=2, y=-1.8, z=0.2, barva=(0, 1, 0))
st.TVAR_konec_definice()

import random

SIRKA = 10
HLOUBKA = 10
KROK = 3
for ix in range(SIRKA):
    for iz in range(HLOUBKA):
        st.TVAR("kytka", x=ix*KROK, z=iz*KROK, ry=random.
            random()*360)

xrozmer = KROK*SIRKA
zrozmer = KROK*HLOUBKA
yrozmer = 0.3
st.Kvadr(sx=xrozmer, sz=zrozmer, sy=yrozmer, x=xrozmer/2-2, z=
    zrozmer/2-2, y=-4, barva=(0.5,0.3,0))
```

Obr. 11 Pole téměř slunečnicové

Protože popis scény je program, můžeme do něj psát komentáře.

Tato scéna zobrazuje 100 květin, každá z nich se skládá ze 13+2 objektů, to máme dohromady 1 501 trojrozměrných objektů. Stavitelka zvládne

zobrazovat menší tisíce objektů, takže u takovéhle scény už může zobrazování začít drhnout. Ale na hraní a pokusy by to mělo stačit.

Dodatečná změna barvy

Už jsme viděli, že mezi parametry objektů Stavitelky je také parametr `barva`. U základních objektů (Koule, Valec, Kvadr) je její význam jasný. Parametr `barva` ale můžeme zadat, i když vytváříme `tvar`. A tento parametr může tvar předat svému objektu, který o to „požádá“: Když v definici tvaru použijeme barvu `BAREVNY_PARAMETR`, bude nahrazena barvou, kterou zadáme při vytváření tvaru ve scéně, například:

```
st.TVAR_zacatek_definice("listek")
st.Koule(x=1, sz=0.2, sy=1/3, barva=st.BAREVNY_PARAMETR)
st.TVAR_konec_definice()

st.TVAR_zacatek_definice("kytka")
...
    st.TVAR("listek", rz=i*360/N, barva=st.BAREVNY_PARAMETR)
...
st.TVAR_konec_definice()

import random

barvy = [(1,0,0),
(1,1,0),(1,0.5,0),(0,1,1),(1,0,1),(1,0,0.5),(0,0,1)]

...
st.TVAR("kytka", x=ix*KROK, z=iz*KROK,
    ry=random.random()*360, barva=barvy[random.randint(0,6)])
```

Obr. 12 Objekt předává svou barvu vnořenému tvaru – květina svému okvětnímu lístku

Zde dostane každá vytvořená květina parametr `barva`, kterým se nahradí barevný parametr `(BAREVNY_PARAMETR)` v její definici i v jí podřízené definici okvětního lístku.

3. Funkce definující tvar

Když naučíme Stavitelku, jak má vypadat nějaký tvar, zapamatuje si to a pak můžeme vytvářet jeho instance, které budou mít vlastní `polohu`, `zvětšení` a `rotaci`. Kdybychom potřebovali dodatečně měnit některé parametry, nejde to, až na výše uvedenou výjimku jednoho parametru pro barvu.

Ale můžeme si pomoci funkcí, která pro každou hodnotu parametrů nadefinuje nový tvar; možná to vypadá složitě, ale v jazyku Python je to vcelku jednoduché, třeba takto:

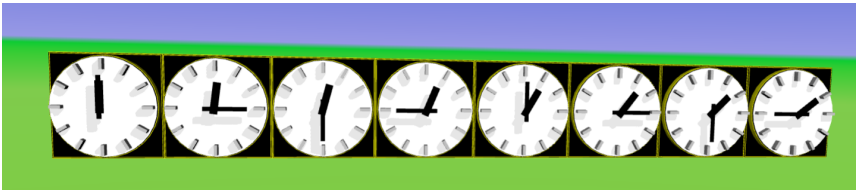
```
def Hodiny(hodiny, minuty):
    jmeno = f"hodiny_{hodiny}_{minuty}"

    st.TVAR_zacatek_definice(jmeno)
    st.TVAR("cifernik")
    uhel_mala = -360*(hodiny+minuty/60)/12
    st.TVAR("mala_rucicka", z=0.3, rz=uhel_mala)
    uhel_velka = -360*minuty/60
    st.TVAR("velka_rucicka",  z=0.3, rz=uhel_velka)
    st.TVAR_konec_definice()

    return jmeno

for hodiny in range(2):
    for minuty in range(0,60,15):
        st.TVAR(Hodiny(hodiny, minuty),
                x=4*hodiny + minuty/15)
```

V této funkci používáme tři tvary: `"cifernik"`, `"mala_rucicka"` a `"velka_rucicka"`, jejichž definice zde neuvádíme.



Obr. 13 Cyklem můžeme nejen generovat řadu objektů, ale díky funkci i definovat nové tvary

Každé zavolání funkce Hodiny, třeba Hodiny(12, 45) nadefinuje nový tvar s odpovídajícím jménem, třeba `hodiny_12_45`* a zavolání funkce st.TVAR potom vytvoří instanci tohoto nového tvaru.

4. Off-line Stavitelka

Pokud nám nevyhovuje online prostředí, anebo pokud chceme na psaní programů používat nějaké nástroje k tomu určené, můžeme použít modul stavitelka.py (odkaz je dole na stránce aplikace). Jen musíme na začátek programu doplnit příkaz

```
import stavitelka as st
```

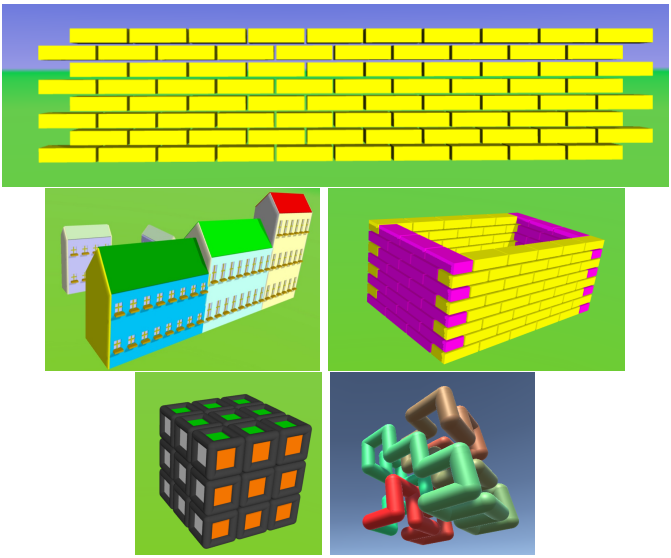
a na konec volání funkce

```
st.ZobrazScenu()
```

Příkazy mezi tím budou stejné jako ve výše popsané online verzi a poslední příkaz vytvoří HTML soubor, který se automaticky otevře v prohlížeči.

5. Úlohy

Pokud vás nenapadá, co byste mohli pomocí Stavitelky stavět, zkuste třeba šachovnici, zeď z cihel daných rozměrů a s danými mezerami, čtyři zdi navazující kolem dokola, hrad, stůl s počítačem, počítačovou učebnu (spousta stolů s počítači), budovu vaší školy, Rubikovu kostku nebo potrubí.



Obr. 14 Díky Stavitelce vygenerujeme i složitější scénu