Geometric figures 1.3.3 – uživatelská příručka

Lukáš Ondráček

Aplikace slouží k vykreslování mnohostěnů libovolné dimenze, generování jejich konvexního obalu a provádění dalších transformací prostřednictvím zásuvných modulů psaných v Pythonu.

Vykreslování je umožněno iterováním perspektivy až k dosažení projekce do 2D.

V adresáři `figures` jsou k dispozici připravené soubory s 3D a 4D platónskými tělesy a několika dalšími. Popis tělesa může být uložen v binárních souborech (.dat), otevíraných příkazem `open`, nebo jako skripty v Pythonu (.py), které lze načíst příkazem `source`.

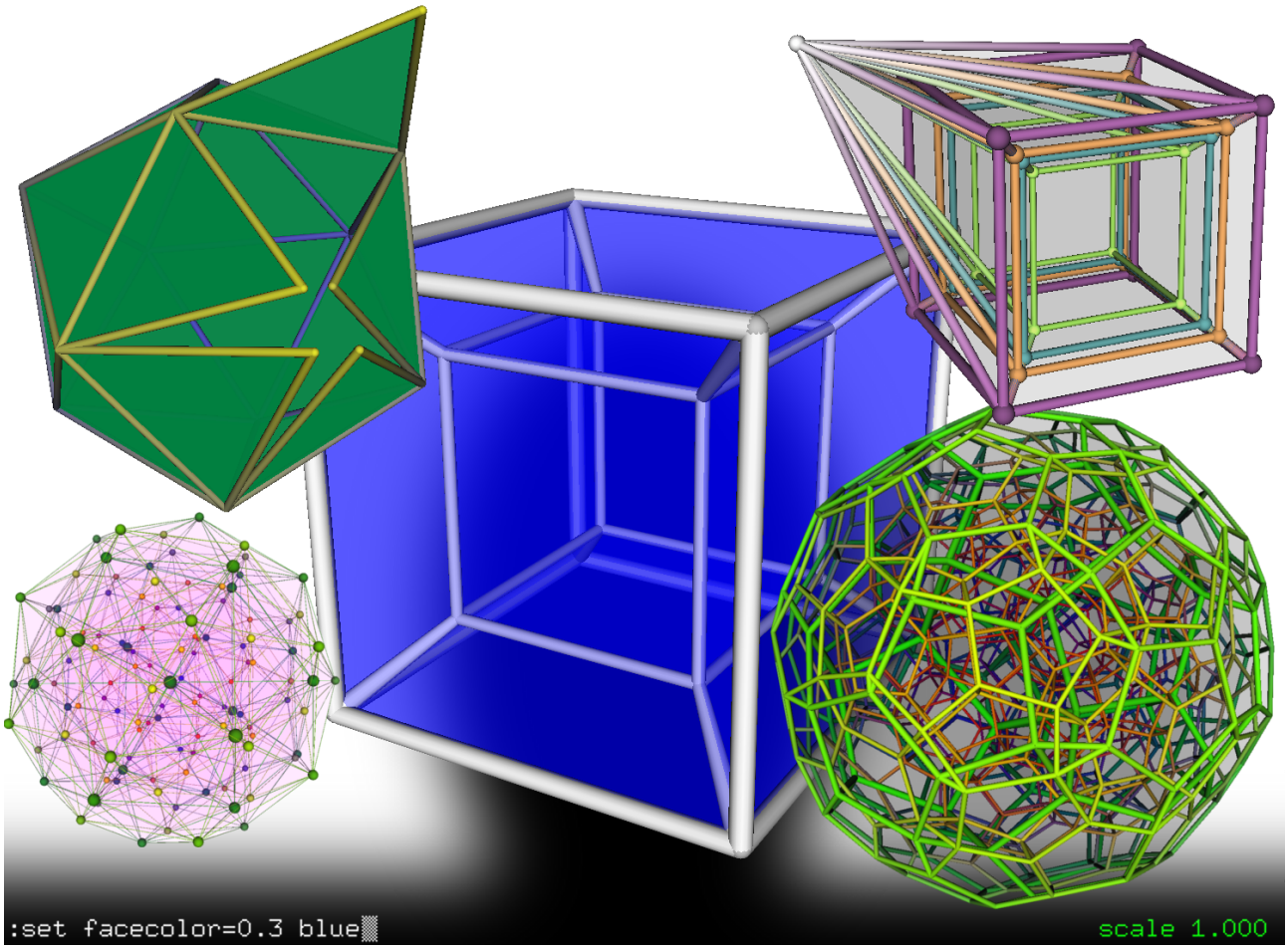
Z adresáře `modules` se načítají zásuvné moduly, Aktuálně jsou k dispozici:

- `duals` pro vytváření duálních těles,
- `cuts` pro odřezávání vrcholů (hran, stěn, ...),
- `figureInfo` pro vypisování informací o otevřených tělesech,
- `helpMod` zprostředkovávající nápovědu k ostatním modulům,
- `randomRot` pro náhodné rotace těles,
- `spaceNavigator` přidávající podporu 3d-myši
- a další, poskytující podpůrné funkce pro jiné moduly.

Konfigurace aplikace je uložena v souboru `config.py` ve složce s aplikací.

Jedná se o svobodný software poskytovaný pod licencí `GNU General Public Licence v3`, více informací viz soubor `README`, celý text licence je k dispozici v souboru `COPYING`.

Aplikace je vyvíjena primárně pro Linux a některé její funkce nemusí být na Windows k dispozici. Testováno na Debianu a Windows XP.



Instalace

Aplikaci je možné stáhnout z http://ondracek-lukas.hys.cz/projects ve verzi pro Linux (32-bit i 64-bit) a pro Windows (jen 32-bit). Stažený soubor je archiv (`zip` nebo `tar.gz`), který stačí rozbalit do libovolné složky.

(Ze stejné adresy vede i odkaz na github, odkud je možné stáhnout aktuální zdrojové kódy aplikace – často v novější verzi než předkompilovaná aplikace. Pro instalaci ze zdrojových souborů si přečtěte u nich přiložený soubor README.)

Pro spuštění aplikace jsou potřeba knihovny OpenGL, Freeglut 3.0 a Python 2.7.

Python je možné stáhnout z http://www.python.org/downloads, příp. na Linuxu i jako balík `python2.7`.

Freeglut pro Windows lze stáhnout z http://www.transmissionzero.co.uk/software/freeglut-devel/ jako `freeglut 3.0.0 MinGW Package`, stažený archiv obsahuje soubor `freeglut.dll`, který je potřeba zkopírovat do složky s aplikací. Pro Linux by měl být k dispozici balík `freeglut3`.

OpenGL pravděpodobně již máte nainstalován.

Po získání potřebných závislostí můžete spustit soubor `geometric_figures`.

Omezení

Maximální počet rozměrů je 100.

Maximální vzdálenost vrcholu od počátku je 1000. (souřadnice jsou reálná čísla)

Hranice všech těles (vč. stěn) musí být souvislá.

Ovládání

Při návrhu uživatelského rozhraní jsem vycházel z textového editoru Vi a přístup k většině prvků aplikace je tak realizován pomocí příkazového řádku umístěného v levém dolním rohu, pro jeho otevření stiskněte `:`. (Seznam příkazů níže.)

Kromě `:` se funkce všech ostatních kláves nastavuje v konfiguračním souboru. Ve verzi 1.2.1 byla přidána i podpora myši, která bez konfiguračního souboru také nemá žádnou funkci.

Výchozí konfigurace

Klávesám je většinou přiřazena funkce podle polohy na **anglické** klávesnici namísto jejich popisu.

Zobrazení nápovědy ke konfiguraci: `?` ... poskytuje modul `helpMod`

Otevírání těles: (z adresáře figures)

0D	**3D**	**4D**	**5D**
~ bod (0d)	2 čtyřstěn (3d-04)	7 5-nadstěn (4d-005)	
1D	3 krychle (3d-06)	8 teserakt (4d-008)	_ penterakt (5d-010)
‘ úsečka (1d-2)	4 osmistěn (3d-08)	9 16-nadstěn (4d-016)	+ 5-ortoplex (5d-032)
2D		0 24-nadstěn (4d-024)	
1 trojúhelník (2d-3)	5 dvanáctistěn (3d-12)	- 120-nastěn (4d-120)	
čtverec (2d-4)	6 dvacetistěn (3d-20)	= 600-nadstěn (4d-600)	
pětiúhelník (2d-5)			
šestiúhelník (2d-6)			

Otáčením kolečka myši se prochází všechna tělesa ve složce s posledně otevřeným souborem. (Není-li žádné těleso otevřeno, použije se adresář `figures`.)

Otáčení: (vždy kolem počátku v rovině dané dvojicí poloos)

2D: `q-w` xy

3D: `a-s` xz, `z-x` yz, pohyb myší se stisknutým levým tlačítkem (xz, yz)

4D: `d-f` x4, `c-v` y4, `e-r` z4, pohyb myší se stisknutým pravým tlačítkem (x4, y4)

(se stisknutým `Shift`em se klávesami otáčí skokově po 15°)

Náhodná rotace: `TAB` ... poskytuje modul `randomRot`

Perspektivní projekce `3D`→`2D` a `4D`→`3D` lze nastavit pohybem myši se stisknutým kolečkem.

Úpravy:
- Odebrání/přidání vrcholu: `P-N`
- Výběr předchozího/následujícího vrcholu: `p-n`
- Zrušení výběru: `o`/`Esc`
- Posun vybraného vrcholu: `h-l` x, `j-k` y, `u-i` z, `m-,` 4

Hledání konvexního obalu: `y-t` (start-stop)
Obnovit původní natočení tělesa: `g`
Vymazat hranici tělesa: `b`
Barvy:
- Výchozí barvy aplikace: `F1`
- Výchozí barvy konfiguračního souboru: `F2`
- Barvení os: `F3` 4. osa, `F4` 3. a 4. osa
- Barva stěn: `F5`, `F6`, `F7`

Velikosti hran a vrcholů: `F8`, `F9`, `F10`, `F11`, `F12`

Seznam příkazů

Argumenty příkazů musí být odděleny právě jednou mezerou.
Příkazy se překládají na výrazy Pythonu, které se poté vykonávají,
formát zápisu číselných literálů je tak stejný jako v Pythonu (používá se desetinná tečka namísto čárky),
barvy se zapisují ve formátu `#`*AARRGGBB*, `#`*RRGGBB* nebo jménem barvy
(konfigurační soubor zavádí `black`, `white`, `red`, `green`, `blue`, `yellow`, `cyan`, `purple`, `gray`, `transparent`).
AA, *RR*, *GG*, *BB* jsou hexadecimální hodnoty v rozsahu `00`–`FF`
udávající krytí a červenou, zelenou a modrou barevnou složku.

`help` [*příkaz*]
`help module` *název_modulu* … poskytuje modul `helpMod`
`help config` … poskytuje modul `helpMod`
Zobrazuje nápovědu,
v první variantě k příkazu (příp. obecnou nápovědu),
ve druhé variantě k modulu,
v poslední variantě k aktuální konfiguraci (obsahuje i názvy načtených modulů).
Pro zobrazení nápovědy musí mít okno dostatečné rozměry,
v opačném případě se zobrazí pouze upozornění.

`n`[`ew`] *počet_rozměrů*
Vytváří prázdný prostor daného počtu rozměrů.

`o`[`pen`] *cesta*
Otevírá binární soubor s uloženým popisem tělesa.
`~/` na začátku cesty se expanduje na domovský adresář, `%/` na umístění aplikace
(na Windows můžou být lomítka zpětná).

`w`[`rite`] *cesta*
Ukládá těleso do souboru, souřadnice vrcholů jsou přepočítány podle aktuálního natočení tělesa.
Opět dochází k expanzi `~/`, `%/`.

`close`
Ruší otevřené těleso.

`so`[`urce`] *cesta*
Provede skript Pythonu v hlavním jmenném prostoru (`__main__`).
V cestě dochází k expanzi `~/`, `%/`.

`q`[`uit`]
`exit`
Ukončí aplikaci.

`info` ... poskytuje modul `figureInfo`
Zobrazuje informace o otevřeném tělese.

`rot`[`ate`] *osa1 osa2 úhel*
Otočí těleso kolem středu souřadného systému v rovině dané osami souřadnic o daný úhel,
na pořadí os záleží.
Osy jsou označeny čísly, počínaje jedničkou.
Osa 1 (x) směřuje doprava, osa 2 (y) nahoru, osa 3 (z) a každá další ke kameře.
Úhel je ve stupních.

`rmap` *událost osa1 osa2*
`rmap` *událost*
První varianta přiřadí události rovinu rotace.
Pořadí os určuje směr rotace.
Druhá varianta ruší přiřazení.
Událost může být klávesová zkratka, tj. jeden znak ASCII nebo <[`c`][`a`][`s`]-*klávesa*>,
kde pomlčku předchází volitelné modifikátory Ctrl, Alt, Shift,
klávesa může být znak ASCII nebo `enter`, `esc`, `bs`, `delete`, `up`, `down`, `left`, `right`,
`home`, `end`, `pageup`, `pagedown`, `insert`, `f1`..`f12`, `mouse0`..`mouse9`;
rotace pak probíhá po celou dobu stisku klávesové zkratky.
Událost také může být osa myši s tlačítky a modifikátory, tj. <[`c`][`a`][`s`]-`mouse`[`0`][`1`][`2`][`3`][`4`][`5`][`6`][`7`][`8`][`9`](`x`|`y`)>,
rotace pak probíhá při pohybu myší v dané ose,
je-li stisknuta odpovídající kombinace tlačítek myši s modifikátory;
tlačítka myši jsou `0`-levé, `1`-kolečko, `2`-pravé, ...

`map` *událost příkaz*
`map` *událost*
První varianta přiřadí události příkaz, druhá varianta přiřazení ruší.
Událost může být klávesová zkratka, tj. znak ASCII nebo <[`c`][`a`][`s`][`r`]-*klávesa*>,
bez modifikátoru `r` stisk zkratky, s ním uvolnění zkratky (Release).
Událost také může být osa myši tj. <[`c`][`a`][`s`]-`mouse`[`0`][`1`][`2`][`3`][`4`][`5`][`6`][`7`][`8`][`9`](`x`|`y`)>.
Příkaz může být buď cokoliv, co lze napsat do konzole, nebo přímo výraz Pythonu.
V případě, že událostí je osa myši, musí příkaz obsahovat %,
které je následně nahrazeno změnou pozice myši v dané ose v pixelech od posledního volání.

`set`
`set` *proměnná*[`?`]
`set` [`no`]*proměnná*
`set` *proměnná*=*hodnota*
První varianta vypisuje všechna nastavení,
druhá jen konkrétní proměnnou (pro logické hodnoty je vyžadován otazník).
Třetí varianta nastavuje logické hodnoty,
čtvrtá nastavuje ostatní typy proměnných.
(Dostupné možnosti níže.)

`reset colors`
Obnoví výchozí nastavení barev.

`reset rot`[`ation`]
Obnovuje výchozí natočení tělesa.

`reset boundary`
Ruší těleso, ponechává pouze vrcholy.
Pokud je zapnuto hledání konvexního obalu, bude konvexní obal vytvořen znova.

`randomrot` ... poskytuje modul `randomRot`
Zapíná náhodnou rotaci tělesa.

`randomrot` [`no`]`auto` ... poskytuje modul `randomRot`
Zapíná/vypíná automatickou náhodnou rotaci po otevření tělesa.

`randomrot map` *klávesová_zkratka* ... poskytuje modul `randomRot`
Přiřazuje klávese funkci náhodné rotace.
Klávesová zkratka může být <[`c`][`a`][`s`][`r`]-*klávesa*>.

`vert`[`ex`] `sel`[`ect`] *číslo_vrcholu*
`vert`[`ex`] `next`
`vert`[`ex`] `prev`[`ious`]
`vert`[`ex`] `desel`[`ect`]
Označí konkrétní, následující, předchozí vrchol jako aktivní, zruší označení.
Vrcholy jsou číslovány od nuly.

`vert`[`ex`] `add` [*souřadnice1* [*souřadnice2* [...]]]
Přidává nový vrchol.
Souřadnice jsou relativní k aktuálnímu natočení tělesa, neuvedené se nastaví na `0`.
Pokud je aktivní hledání konvexního obalu, dojde k jeho přepočítání,
jinak se vytvoří volný vrchol.

`vert`[`ex`] `move` [*souřadnice1* [*souřadnice2* [...]]]
Posune aktivní vrchol o uvedené souřadnice.
Pokud je aktivní hledání konvexního obalu, přepočítá se,
jinak dojde pouze k rozlámání všech stěn incidentních s daným vrcholem,
jejichž dimenze je menší než dimenze prostoru, bez ohledu na novou polohu vrcholu.
Každá stěna se rozlomí tak,
aby žádným posunem aktivního vrcholu nemohlo dojít k rozšíření dimenze prostoru generovaného stěnou
(tedy na trojúhelník, čtyřstěn, ... a zbytek).

`vert`[`ex`] `r`[`e`]`m`[`ove`]
Odstraní aktivní vrchol.
Je-li aktivní hledání konvexního obalu, přepočítá se,
jinak dojde k odstranění všech stěn incidentních s daným vrcholem,
těleso se tak rozpadne na několik těles nižší dimenze.
Pokud aktivní zobrazení tvoří více těles, odstraní se tělesa (ne stěny) incidentní s vrcholem.

`create dual` ... poskytuje modul `duals`
Vytváří duální těleso k otevřenému,
to musí být konvexní a minimálně dvourozměrné
a žádná jeho stěna nesmí procházet počátkem.

`cut vertices` *poměr* ... poskytuje modul `cuts`
`cut edges` *poměr* ... poskytuje modul `cuts`
`cut faces` *dimenze poměr* ... poskytuje modul `cuts`
Odřeže z daného tělesa vrcholy, hrany, nebo obecně stěny dané dimenze.
Parametr udává poměr vzdáleností stěny od nadroviny řezu a stěny od počátku.

`spacenavigator` ... poskytuje modul `spaceNavigator`
Vypíše informace o stavu modulu spaceNavigator.

`spacenavigator sensitivity`[=*value*] ... poskytuje modul `spaceNavigator`
Vypisuje/nastavuje citlivost 3d-myši.

`spacenavigator device`[=*path*] ... poskytuje modul `spaceNavigator`
Vypisuje/nastavuje cestu k zařízení 3d-myši.
Po první úspěšné aktivaci zařízení již není možné měnit jeho cestu.

`spacenavigator` (`on`|`off`) ... poskytuje modul `spaceNavigator`
Aktivuje/deaktivuje 3d-myš.
Zařízení je automaticky aktivováno, je-li při načítání modulu nalezeno ve výchozím umístění,
v takovém případě se deaktivuje automatická náhodná rotace
(v průběhu náhodné rotace není možné 3d-myš používat).

`history`
Vypíše historii zadaných příkazů.

Nastavení (příkaz `set`)

Datovým typem proměnné může být reálné číslo, celé číslo, logická hodnota, nebo barva.

`background` – barva
Barva pozadí.

`campos`*osa* – číslo
Udává pozici kamery v dané ose (od třetí).
Vzdálenost je kompenzována zvětšením tak, aby vždy byla vidět koule (dané dimenze) o poloměru 1.
Před vykreslením je dále upraveno měřítko tělesa tak, aby se zevnitř dotýkalo jednotkové koule
(míru zvětšení udává koeficient `scale` zobrazený v pravém dolním rohu).
Měřítko tedy nemá vliv na vzdálenost kamery.

`camposl`*osa* – číslo
Stejný význam jako předchozí ale v logaritmické stupnici.
Lépe vystihuje přechod od perspektivy k rovnoběžnému promítání.

`convexhull` – logická hodnota
Určuje, zda se bude udržovat konvexní obal.
Po každém přepočítání se v příkazovém řádku objeví počet vrcholů, hran, stěn, buněk, ...

`dimen` – celé číslo, pouze pro čtení
Určuje počet rozměrů prostoru.

`edgesize` – číslo
Šířka hrany procházející počátkem v rovině xy v pixelech.
Zvětšením může dojít také ke zvětšení `vertsize` a `selvertsize`,
aby neměly menší hodnoty.

`facecolor` – barva s krytím
Nastavuje barvu a průhlednost stěn.

`grabmouse` – logická hodnota
Určuje, zda se má zachytit kurzor myši při použití v aplikaci.
Je-li zapnuto, kurzor myši při stisknutí tlačítek zmizí a po uvolnění se opět objeví na stejném místě,
tělesem je tak možné otáčet beze strachu z nárazu na hranici obrazovky.

`history` – celé číslo
Počet posledních příkazů, které se uchovávají v historii.
Po otevření příkazového řádku je možné se k nim vrátit šipkou nahoru.

`maxfps` – číslo
Určuje maximální snímkovou frekvenci – počet snímků za sekundu.

`mousesens` – číslo
Citlivost myši při použití pro otáčení těles ve stupních na pixel.

`pyexpr` – logická hodnota
Určuje, zda je možné místo příkazů zadávat také přímo výrazy Pythonu.
Je-li zapnuto a nepodaří-li se rozpoznat zadaný příkaz,
je interpretován jako výraz Pythonu.

`selvertcolor` – barva s krytím
Barva vybraného vrcholu.

`selvertsize` – číslo
Velikost vybraného vrcholu v počátku souřadného systému v pixelech.
Zmenšením může dojít také ke zmenšení `vertsize` a `edgesize`,
aby neměly větší hodnoty.
Nastavení příliš vysoké hodnoty ovlivní měřítko zobrazení, i když není žádný vrchol vybraný;
může se tak stát, že celé těleso splyne v jednom malém bodu uprostřed.

`spacecolor` – barva
`spacecolor+`*osa* – barva s krytím
`spacecolor-`*osa* – barva s krytím
Umožňuje přidělit každému bodu v prostoru barvu,
která se použije na vykreslení vrcholu nebo hrany v daném místě.
První varianta udává základní barvu (v počátku souřadného systému).
Druhá a třetí varianta určuje barvu v nadrovině kolmé na danou osu a protínající ji v +/- 1.
Jsou-li nastaveny barvy ve více osách, průměrují se (s ohledem na polohu bodu i nastavená krytí).

`speed` – číslo
Úhlová rychlost otáčení při držení klávesy namapované pomocí `rmap` ve stupních za sekundu.

`stdoutpyexpr` – číslo
Zapíná vypisování vykonávaných příkazů přeložených do Pythonu na standardní výstup.
Je možné použít k vygenerování části konfiguračního souboru.

`vertsize` – číslo
Velikost vrcholu v počátku souřadného systému v pixelech.
Zmenšením může dojít také ke zmenšení `edgesize`, aby neměla větší hodnotu.
Zvětšením může dojít ke zvětšení `selvertsize`, aby neměla menší hodnotu.

Vykreslování

Pro vykreslení vícerozměrného tělesa je potřeba použít zobrazení, které sníží počet jeho rozměrů. V této aplikaci je toho docíleno opakovaným použitím perspektivy.

Perspektivu si zjednodušeně můžeme představit tak, že vyjdeme ze souřadnic původního bodu bez poslední (tedy snížíme počet rozměrů) a zahozená souřadnice pak bude ovlivňovat vzdálenost bodu od středu souřadného systému nižší dimenze. Ona poslední souřadnice bude představovat vzdálenost od kamery: bude-li zobrazovaný bod daleko, bude jeho obraz blízko středu souřadného systému; bude-li blízko, obraz bude dále od středu (při zachování ostatních souřadnic).

Použijeme-li toto zobrazení na trojrozměrný prostor, dostaneme známý efekt – při posunu tělesa kolmo od projekční roviny (od kamery) se bude jeho obraz (na fotografii) přibližovat středu.

Stejné je to i v případě projekce ze 4D do 3D, akorát musíme připustit, že čtyřrozměrná kamera má trojrozměrný povrch, na nějž se bude promítat; ten leží v jedné nadrovině a na ní je kolmá osa určující vzdálenost. Budeme-li tělesa posouvat dále od kamery, budou se jejich obrazy přibližovat počátku i sobě navzájem.

Takto si například můžeme představit kostru čtyřrozměrné krychle. Ta je zepředu a zezadu ohraničena krychlemi, které jsou spojené, což v perspektivní projekci vytváří (3D) obraz krychle v krychli. Vnitřní krychle představuje tu vzdálenější, vnější tu bližší, pojem vzdálenosti tak má stále jasný význam, vnímáme-li vzniklý (3D) obraz jako celek.

Problém vznikne, použijeme-li perspektivu vícekrát, abychom snížili počet rozměrů o více než jeden. Začneme-li se například na onu „krychli v krychli“ dívat z některé konkrétní strany, objeví se druhá vzdálenost – jiná než ta předchozí –, což je naprosto nepřirozené, ale je potřeba si na to v této aplikaci zvyknout.

Zajímavou vlastností perspektivy je rozdíl v tom, jestli se kamera posune blíž, nebo se záběr pouze přiblíží. Jsou-li pozorované objekty daleko od kamery (což se kompenzuje zvětšením), nejsou až tak patrné rozdíly ve vzdálenostech mezi nimi. (Efekt velkého měsíce na fotografiích.)

V aplikaci je možné nastavit vzdálenost kamery `campos`, která se kompenzuje přiblížením, což při příliš vysokých hodnotách konverguje k rovnoběžnému promítání. Pro lepší vystižení přechodu mezi perspektivou a rovnoběžným promítáním je k dispozici i stejná proměnná v logaritmické stupnici `camposl`. U hodnot kolem 10 už je perspektiva nepozorovatelná, zatímco u hodnot pod 0.5 jsou její účinky nepřirozeně zveličeny.

A protože je perspektiva použita opakovaně, je toto nastavení k dispozici pro „všechny vzdálenosti“ od kamery (campos[l]3, campos[l]4, campos[l]5, ...).

Pro usnadnění orientace ve vícerozměrném prostoru je možné hrany a vrcholy obarvit podle jejich souřadnic.

Například u čtyřrozměrných těles jsou ve výchozím nastavení body s kladnou čtvrtou souřadnicí (blízko u kamery) zbarveny do červena a body se zápornou čtvrtou souřadnicí (daleko od kamery) do zelena, mezi nimi se pak tvoří lineární barevný přechod. S otáčením tělesa se barvy mění podle aktuálních souřadnic.

Barvit body je možné i podle více os současně, pak se jednotlivá obarvení začnou mísit. Stiskem F4 se kromě 4. osy obarví také 3. (blízko žlutá, daleko modrá). Příp. po otevření pětirozměrného tělesa se obarví 4. osa (stejně jako výše) a 5. osa (blízko modrá, daleko žlutá).

Barvení je možné nastavit libovolně pomocí proměnných `spacecolor` (viz výše).

Použití Pythonu

Po spuštění aplikace se automaticky importují moduly z adresáře `%/modules` (`%` označuje umístění spustitelného souboru aplikace) do hlavního jmenného prostoru `__main__`, poté se v tomto jmenném prostoru provede konfigurační soubor `%/config.py` (stejně se spouští také soubory načtené příkazem `source`).

V hlavním jmenném prostoru je tedy možné moduly používat přímo, v souborech modulů je potřeba je nejdříve importovat.

Pokud funkce volaná z aplikace vyhodí výjimku, je její text vypsán do konzole aplikace. Pokud vznikne výjimka při vykonávání skriptu (příp. importování modulu), vypíše se navíc stack trace na chybový výstup.

Přístup k aplikaci (modul gf)

Modul `gf` umožňuje přístup k aplikaci. Je možné jej používat pouze z hlavního vlákna Pythonu.
Funkce můžou vyhazovat výjimky, téměř vždy jde o `RuntimeError` blíže specifikovanou pouze textem výjimky (jejich odchytávání se nepředpokládá).

Pokud funkce přijímá nebo vrací barvu, vždy jde o textový řetězec (`"#`$[AA]RRGGBB$`"` nebo jméno barvy).
Pro převod barvy z/do použitelnějšího formátu lze použít funkce modulu `utils`.

Poloha vrcholu je n-tice souřadnic.
Těleso (resp. tělesa) je formátu:

```
[
    [ (vrchol1_x, vrchol1_y, ...), ...],
    [ [hrana1_vrchol1, hrana1_vrchol2], ...],
    [ [stěna1_hrana1, stěna1_hrana2, ...], ...],
    ...,
    [ [těleso1_faseta1, těleso1_faseta2, ...], ...]
]
```

Jde tedy o seznam s $(d+1)$ prvky (pro d počet dimenzí),
kde každý prvek obsahuje seznam těles dané dimenze.
Vrcholy jsou opět d-tice souřadnic,
vyšší tělesa jsou dány seznamem indexů stěn nižší dimenze, které je ohraničují.
Objektově orientovaný přístup k tělesu zajišťuje modul `objFigure`.

Funkce bez popisu fungují stejně jako výše popsané příkazy stejného jména, jejich argumenty jsou buď čísla nebo textové řetězce.

`addColorAlias(`*jméno*`,` *barva*`)`
Zavání novou pojmenovanou barvu.
Barva může být jakákoliv aktuálně přípustná, tedy i dříve vytvořený název.

`addCommand(`*příkaz*`,` *výraz_pythonu*`)`
`addCommand(`*příkaz*`,` *výraz_pythonu*`,` *počet_parametrů*`)`
`addCommand(`*příkaz*`,` *výraz_pythonu*`,` *počet_parametrů*`,` *typy_parametrů*`)`
Přidává nový příkaz, který se bude překládat na zadaný výraz Pythonu.
Je-li počet parametrů 0 nebo není zadán, za příkazem nesmí následovat žádné další znaky;
je-li kladný, jsou postupně nahrazovány znaky `%` ve výrazu Pythonu parametry;
je-li záporný, jsou čárkami oddělené parametry dosazeny za první %.
Počet parametrů v absolutní hodnotě udává, kolikátý parametr bude obsahovat zbytek příkazu vč. mezer.
Parametry jsou standardně dosazovány tak, jak jsou, (jako literály) a nefunguje pro ně AutoComplete;
to je možné změnit uvedením jejich typů.
Poslední parametr je textový řetězec obsahující jeden znak za každý parametr,
poslední znak určuje typ všech zbývajících; možné typy:

- `-` Literál (výchozí)
- `s` Textový řetězec
- `p` Textový řetězec doplňovaný jako cesta
- `c` Textový řetězec doplňovaný jako barva
- `C` Textový řetězec doplňovaný jako barva s průhledností

`gf.clear()`
Vymaže text vypsaný do konzole aplikace.

`gf.clearAfterCmd()`
`gf.clearAfterCmd(text)`
Vypíše jeden nový řádek.
Začne-li uživatel poté psát příkaz, je smazán pouze tento řádek.
(V první variantě jde o výchozí text *„Press any key or type command to continue.“*)

`gf.clearBeforePrinting()`
Před příštím vypisováním do konzole dojde k jejímu smazání,
namísto obvyklého připisování nových řádků.

`gf.close()`

`gf.echo(`*text*`)`
Vypíše text (i víceřádkový).
Při opakovaném volání se připisují nové řádky.

`gf.echoErr(`*text*`)`
Vypisuje text červeně.

`gf.expandPath(`*cesta*`)`
Expanduje v cestě `%/` a `~/`.

`gf.figureGet()`
Vrací aktuálně otevřené těleso.

`gf.figureOpen(`*těleso*`)`
`gf.figureOpen(`*těleso*`, `*zachovat_natočení*`)`
Otevírá dané těleso.
Pokud je zachování natočení nastaveno na `True`,
nedochází k události otevření tělesa, ale pouze k jeho změně;
těleso v takovém případě musí mít stejný počet rozměrů (souřadnic) jako to právě otevřené.

`gf.figureRead(`*cesta*`)`
Načte těleso ze souboru (binárního) a vrátí jej.

`gf.figureWrite(`*cesta*`, `*těleso*`)`
Uloží těleso do (binárního) souboru.

`gf.help(`*jméno*`)`

`gf.history()`

`gf.map(`*událost*`, `*příkaz*`)`
`gf.map(`*událost*`)`

`gf.new(`*počet_rozměrů*`)`

`gf.normalizeColor(barva)`
`gf.normalizeColorAlpha(barva)`
Normalizuje zadanou barvu (i pojmenovanou),
vrátí řetězec ve formátu `#`*RRGGBB*, resp. `#`*AARRGGBB* u druhé varianty, kde jsou všechna písmena velká.
První varianta připouští pouze barvy bez kanálu krytí.
Pokud barva není rozpoznána vyhodí funkce výjimku.

`gf.open(`*cesta*`)`

`gf.posRotate(`*pozice*`)`
`gf.posRotateBack(`*pozice*`)`
Transformuje souřadnice bodu, aby uvažovaly aktuální natočení tělesa,
resp. zpět do absolutních souřadnic ve druhé variantě.

`gf.printCentered(text)`
Vypíše blok textu na střed obrazovky, stejně jako nápověda.

`gf.quit()`

`gf.registerCallback(`*název_události*`, `*funkce*`)`
Registruje callback pro některou z událostí.
Na rozdíl od příkazu `map`, k těmto událostem je možné přiřadit postupně i více funkcí.
Druhý parametr je přímo funkce (tedy typ `Callable` namísto textu), její parametry jsou určeny událostí.
Seznam událostí:

`"idle"()`
: Nastává jednou v každém snímku, neprobíhá-li žádná jiná akce
(vykonávání skriptu, otevřená konzole, otáčení tělesa, ...).
Může být tedy voláno podle `maxfps` i mnohokrát za sekundu.
Vykonávání skriptu probíhá i tehdy, pokud je v něm právě zpracováván příkaz `gf.sleep`;
`idle` může být v některých případech použito jako neblokující alternativa.

`"new"()`
: Nastává při změně počtu dimenzí prostoru,
přesněji vždy při volání `new`, `open` a `figureOpen` bez zachování rotace.

`"open"(`*cesta*`)`
: Nastává při otevření nového tělesa,
tedy při volání `open` a `figureOpen` bez zachování rotace.
Pokud se těleso načítalo ze souboru, dostane funkce jako parametr jeho cestu, jinak `None`.

`"modified"()`
: Nastává při úpravě tělesa,
tedy při volání `figureOpen` se zachováním rotace a při práci s vrcholy.

`gf.removeCommand(`*prefix_příkazu*`)`
`gf.removeCommand()`
Odstraní všechny uživatelsky přidané příkazy s daným prefixem (příp. úplně všechny).

`gf.resetBoundary()`

`gf.resetColors()`

`gf.resetColorAliases()`
Vymaže všechny pojmenované barvy.

`gf.resetRotation()`

`gf.rmap(`*událost*`, `*osa1*`, `*osa2*`)`
`gf.rmap(`*událost*`)`

`gf.rotate(`*osa1*`, `*osa2*`, `*úhel*`)`

`gf.set_`*proměnná*`(`*hodnota*`)`
`gf.set_`*proměnná*`(`*index*`, `*hodnota*`)`
`gf.get_`*proměnná*`()`
`gf.get_`*proměnná*`(`*index*`)`

`gf.sleep(ms)`
Zpracovává události okna minimálně po zadanou dobu v milisekundách (celé číslo);
je-li potřeba překreslení, dojde k němu (i při ms=0).
Dojde-li k přerušení (například stiskem klávesy), vrátí funkce `False`;
v tom případě by měl skript skončit co nejdříve.
(Událost, která přerušení vyvolala je poté zpracována.)

Příklad použití pro vytvoření animace:

```
minDelay=1000/gf.get_maxfps()
time=gf.time()
while gf.sleep(minDelay):
    newTime=gf.time()
    elapsed=newTime-time
    time=newTime
    Vykonat pohyb za posledních elapsed milisekund.
```

Pro činnosti na pozadí by měl být raději použit callback `idle` (viz `gf.registerCallback`), aby se střídavě mohl vykonávat i jiný kód (během zpracování volání funkce `time idle` nenastane).

`gf.source(`*cesta*`)`

`gf.time()`
Vrací čas v milisekundách od spuštění aplikace.

`gf.unregisterCallback(`*název_události*`, `*funkce*`)`
Ruší registraci funkce jako callbacku.
Musí být zavoláno se stejnými parametry jako předcházející registrace.

`gf.vertexAdd()`
`gf.vertexAdd(`*pozice*`)`
Přidává nový vrchol umístěný v počátku, nebo na danou pozici.
Na rozdíl od příkazu `vertex add`, zde není zohledněno natočení tělesa, souřadnice jsou absolutní.
(Může vyvolat přepočítání konvexního obalu.)

`gf.vertexRemove(index)`
Odstraní zadaný vrchol.
(Může vyvolat přepočítání konvexního obalu.)

`gf.vertexDeselect()`
`gf.vertexNext()`
`gf.vertexPrevious()`
`gf.vertexSelect(`*index*`)`
Určuje vybraný vrchol.

`gf.vertexSelected()`
Vrací index vybraného vrcholu.

`gf.vertexSetPos(`*index*`, `*nová_pozice*`)`
`gf.vertexGetPos(`*index*`)`
Nastavuje/vrací absolutní polohu vrcholu.

`gf.write(`*cesta*`)`

Modul `algebra`

Modul `algebra` zpřístupňuje některé postupy lineární algebry jako funkce.

Vektory jsou vždy n-tice čísel, přijímá-li jich funkce více, většinou musí mít stejný počet složek.
Báze je seznam vektorů.

`algebra.dotProduct(`*vektor1*`, `*vektor2*`)`
Skalární součin vektorů.

`algebra.vectLen(`*vektor*`)`
Euklidovská délka vektoru.

`algebra.vectDiff(`*vektor1*`, `*vektor2*`)`
Rozdíl vektorů.

`algebra.vectSum(`*vektor1*`, `*vektor2*`)`
Součet vektorů.

`algebra.vectMult(`*koeficient*`, `*vektor*`)`
Násobek vektoru.

`algebra.orthogonalizeVect(`*vektor*`, `*ortonormální_báze*`)`
Ortogonalizace vektoru vzhledem k dané ortonormální bázi.

`algebra.orthonormalizeBasis(`*seznam_vektorů*`)`
Ortonormalizace báze.
Vstupem může být libovolná množina vektorů,
báze bude obsahovat tolik vektorů, jakou dimenzi vstup generoval.

`algebra.orthonormalBasisFromPoints(`*seznam_vrcholů*`)`
Vytvoří ortonormální bázi ze seznamu vrcholů.
Poloha afinního prostoru je zanedbána.

`algebra.Hyperplane(normála, pozice_v_normále)`
Třída představující nadrovinu.
Normála je vektor a pozice v ní číslo, po jejich roznásobení bychom měli dostat bod v nadrovině.
Atributy: Normála `.normal`, pozice v ní `.normalPos`.
Metoda `orientedDistance(`*pozice*`)` vrací vzdálenost bodu ve směru normály.

Modul `duals`

Modul `duals` vytváří duální tělesa.

`duals.createDual(`*těleso*`)`
Vrací duální těleso k zadanému.
Parametr i návratová hodnota je typu `objFigure.Figure`.
Není-li vstupní těleso konvexní, je relevantní pouze topologie výstupního tělesa.

Modul `cuts`

Modul `cuts` umožňuje řezání těles a odřezávání jejich částí.

Tělesa jsou vždy typu `objFigure.Figure`, nadroviny `algebra.Hyperplane`.

`cuts.cutFigure(`*těleso*`, `*nadrovina*`)`
Rozřeže těleso nadrovinou na dvě části.
Vrací trojici: záporná část, řez, kladná část; každý prvek trojice je seznamem těles.

`cuts.cutOff(`*těleso*`, `*seznam_nadrovin*`, `*zobrazit_průběh*`)`
`cuts.cutOffFaces(`*těleso*`, `*poměr*`, `*seznam_stěn*`, `*zobrazit_průběh*`)`
`cuts.cutOffFacesDim(`*těleso*`, `*poměr*`, `*dimenze_stěn*`, `*zobrazit_průběh*`)`
Odřeže od tělesa některé části.
První varianta odřezává nadrovinami části s kladnou orientovanou vzdáleností od nich.
Druhá varianta odřezává konkrétní stěny (různých dimenzí) a třetí všechny stěny dané dimenze,
poměr je vzdálenost nadroviny od stěny ku vzdálenosti stěny od počátku.
Je-li zobrazení průběhu nastaveno na `True`, vypisuje se aktuální stav.

Modul `figureInfo`

Modul `figureInfo` zjišťuje a vypisuje informace o tělesech.

Při otevírání tělesa ze souboru *jméno*`.dat` načítá metainformace ze souboru `.`*jméno*`.txt`,
kde jako první řádek očekává název tělesa a jako zbytek souboru jeho popis.
Pokud je těleso generováno jiným způsobem, je potřeba tyto informace po vygenerování zaktualizovat.
Informace se po aktualizaci vypíší.

`figureInfo.setNameDescPath(`*název*`,` *popis*`,` *cesta*`)`
Nastavuje název, popis a cestu k souboru aktuálně otevřeného tělesa,
poté vypíše všechny dostupné informace.

`figureInfo.getNameDesc()`
Vrací jméno a popis tělesa.
(Popis může být `None`.)

`figureInfo.getPath()`
Vrací cestu k souboru aktuálně otevřeného tělesa (je-li známa).

`figureInfo.counts(`*figure*`)`
Spočítá stěny po dimenzích a vrátí textovou reprezentaci výsledku,
argumentem je objekt `objFigure.Figure`.

`figureInfo.printAll()`
Vypíše všechny dostupné informace (jako příkaz `info`).

Modul `gfUtils`

Modul `gfUtils` poskytuje funkce pro pohodlnější přístup k aplikaci.

`gfUtils.openFileRelative(`*relativní_pořadové_číslo*`)`
Umožňuje procházení těles v jednom adresáři.
Z abecedně řazeného seznamu souborů otevře soubor s daným pořadovým číslem relativním k otevřenému.
S parametrem 1 tedy otevře následující soubor, s -1 předchozí.
Pokud není otevřeno žádné těleso nebo k němu není přidružený žádný soubor, použije se složka `%/figures`.
Otevírají se binární soubory s příponou `.dat` i skripty s příponou `.py`.

`gfUtils.openConvexFromVertList(`*seznam_vrcholů*`,` *název*`,` *popis*`,` *cesta*`)`
Vytvoří a otevře těleso, které je konvexním obalem dané množiny vrcholů,
zároveň mu přiřadí název, popis a cestu v modulu `figureInfo`.

Modul `helpMod`

Modul `helpMod` poskytuje nápovědu pro ostatní moduly a aktuální konfiguraci.

`helpMod.addPage(`*název*`,` *obsah*`)`
Přidá stránku nápovědy, kterou je pak možné zobrazit příkazem `help` *název*.

`helpMod.addModule(`*název*`,` *obsah*`)`
Přidá stránku nápovědy modulu, kterou je poté možné zobrazit příkazem `help module` *název*.
Zároveň připíše název modulu k nápovědě k aktuální konfiguraci.

`helpMod.printPage(`*název*`)`
Zobrazí stránku nápovědy, dostupné také pod příkazem `help` *název*.

`helpMod.configAdd(`*text*`)`
Přidá řádky do nápovědy ke konfiguraci.

`helpMod.configPrint()`
Zobrazí nápovědu ke konfiguraci, také dostupné pod příkazem `help config` nebo stiskem `?`.

Modul `objFigure`

Modul `objFigure` poskytuje objektově orientovaný přístup k tělesům.

`objFigure.Figure(`*seznam_těles_v_hranici*`, `*gfIndex*`)`
- `.addToBoundary(`*těleso*`)`
  Přidá těleso do hranice.
- `.rmFromBoundary(`*těleso*`)`
  Odebere těleso z hranice.
- `.boundary`
  Množina obsahující tělesa v hranici.
- `.dim`
  Počet rozměrů tělesa.
- `.spaceDim`
  Počet souřadnic vrcholů.
- `.gfIndex`
  Index v datové struktuře tělesa z modulu `gf` nebo `None`.

Třída představující těleso.
Z objektu je možné získat iterátor procházející všechna tělesa (všech rozměrů) v hranici,
aby bylo každé těleso vráceno jen jednou, může být vždy aktivní globálně maximálně jeden iterátor.

`objFigure.Vertex(`*pozice*`, `*gfIndex*`)`
- `.position`
  Pozice vrcholu.

Třída představující vrchol, potomek předchozí.

`objFigure.fromGfFigure(`*těleso_z_modulu_gf*`)`
Převádí těleso z formátu modulu `gf` na objekt `Figure`.
Nastavuje atributy `gfIndex`.

`objFigure.toGfFigure(`*objekt_Figure*`)`
Převádí objekt Figure do formátu modulu `gf`.
Nastavuje atributy `gfIndex`.

`objFigure.figuresIterator(`*seznam_těles*`)`
Vrací iterátor přes hranice (všech dimenzí) daných těles.
Opět může být aktivní maximálně jeden.

Modul `randomRot`

Modul `randomRot` umožňuje náhodné rotace těles.

`randomRot.randomRot(`*povolit_pohyb_kamery*`)`
Rotace tělesa,
volitelně může měnit i nastavení perspektivy `campos`.

`randomRot.set_auto(`*logická_hodnota*`)`
Zapíná/vypíná automatické náhodné rotace po otevření souboru.

`randomRot.map(`*klávesová_zkratka*`)`
Nastaví mapování klávesové zkratky na zapínání náhodné rotace a přepínání režimů.

Modul spaceNavigator

Modul `spaceNavigator` přidává ve verzi pro Linux podporu 3D myši 3DConnexion SpaceNavigator.
To nejspíš nebude pro většinu uživatelů příliš užitečné,
modul muže být ale také použit jako kostra pro přidání podpory jiného polohovacího zařízení.
Jak zpřístupnit zařízení aplikaci je popsáno přímo v souboru `spaceNavigator.py`.

`spacenavigator.set_device(`*cesta*`)`
`spacenavigator.get_device()`
Nastavuje/vrací cestu k zařízení.

`spacenavigator.set_sens(`*hodnota*`)`
`spacenavigator.get_sens()`
Nastavuje/vrací citlivost zařízení.

`spacenavigator.on()`
`spacenavigator.off()`
Zapíná/vypíná zařízení.

`spacenavigator.isActive()`
Zjišťuje stav zařízení.

`spacenavigator.info()`
Vypisuje informace o stavu modulu.

`spacenavigator.leftBtnFunc`
`spacenavigator.rightBtnFunc`
Funkce volané při stisku jednoho z tlačítek 3D myši.
Je možné je libovolně měnit.

Modul utils

Modul `utils` obsahuje různé užitečné funkce.

`utils.colorToTuple(`*barva*`)`
Převádí textovou reprezentaci barvy na čtveřici (*krytí*, *červená*, *zelená*, *modrá*),
kde každá složka nabývá hodnot 0-255.

`utils.tupleToColor(`*čtveřice*`)`
Převádí čtveřici na textovou reprezentaci barvy.

`utils.figuresMaxRadius(`*seznam_těles*`)`
Vrátí maximální vzdálenost vrcholu od počátku souřadného systému.
Tělesa jsou objekty `objFigure.Figure`.

`utils.figuresScale(`*seznam_těles*, *koeficient*`)`
Vynásobí souřadnice vrcholů daným koeficientem.
Tělesa jsou objekty `objFigure.Figure`.
Funkce pracuje na místě.