

PyDeflektor

Vývojová dokumentace

Marek Balvín

2021

CO JE PYDEFLEKTOR?	3
HLAVNÍ KOSTRA PROGRAMU	3
TŘÍDA PLAY	3
Play.check_events	3
Play.draw_frame	4
Laser	4
Play.draw_laser	4
Časomíra	4
Play.blit_time_left	5
Kolize herních objektů	5
Play.tracer_mirror_events	5
Update herních objektů	5
Play.mirror_update	5
Play.mirror_reset	6
Play.target_update	6
Update herní smyčky	6
Play.update	6
Play.run	6
TŘÍDA LEVEL	6
Level.load_tracer(<i>position, vector</i>)	7
HERNÍ OBJEKTY	7
Třída Wall	7
Třída Source	7
Třída Mirror	8
Mirror.update	8
Mirror.rotate_cw	8
Mirror.rotate_ccw	8
Mirror.rotate_out_cw	8

Mirror.rotate_out_ccw	8
Třída Tracer	9
Tracer.update	9
Třída Target	9
Třída Template	9
 SETTINGS.PY	 9
get_texture	9
get_level	9
 SOUBOR LEVELS.JSON	 10
Data	10
GridWidth, GridHeight	10
TileSize	10
SourceId	10
Countdown	10

Co je PyDeflektor?

PyDeflektor je jednoduchá tile-based hra, jejímž principem je dostat laser ze zdroje do cíle pomocí otáčení zrcadel.

Hlavní kostra programu

Program je implementován v jazyce Python. Je využit modul `Pygame`. Zapíná se přes soubor *main.py*. Hlavní smyčku tvoří třída `Play`, která v podstatě ovládá průběh hraní.

Třída `Play`

Třída `Play` vyžaduje parametr `level`, který určuje, jaký level se má otevřít. Pokud daný level není nalezen ve složce *levels*, vyskočí vyjímka. Level a všechny proměnné s ním spojené jsou uloženy v `Play.level`, jako třída `Level`.

Rozměry herního okna se importují ze souboru *settings.py*.

Běh hry je ovládán především proměnnými `Play.state` a `Play.playing`.

`Play.state` je při spuštění nastavena na „`init`“ během hry je přepínána mezi „`ready`“, „`started`“ a „`level_fin`“.

- „`ready`“ znamená, že level je načten, a hra je pozastavená.
- „`started`“ znamená, že hra běží.
- „`level_fin`“ znamená, že level je dohrán.

Po dohrání posledního levelu je `Play.state` nastavena na „`end`“.

`Play.playing` se využívá při hraní daného levelu, konkrétně říká, zda se hraje, nebo zda je pozastaven, respektive zda je laser v pohybu, nebo ne. Ze začátku je nastavena na `False`, ale během hraní se změní na `True`. Pokud vyprší čas, je opět nastavena na `False`.

`Play.check_events`

Klasický pygame event handling. Hlídá a ovládá se především `Play.state`, `Play.playing` a stisk kláves.

Play.draw frame

Funkce ovládá, co má být zobrazeno v herním okně. Vykresluje herní objekty přes jejich skupiny uložené v daném levelu. Zároveň vykresluje kurzor na pozici myši v herním okně. Podle hodnoty v `Play.state` zobrazuje do herního okna daná informační okna (`Play.init_screen`, `Play.start_screen`, `Play.game_over_screen`, `Play.end_screen`).

Laser

Laser je ve hře implementován pomocí `pygame.draw.lines`, využívající list `Play.laser_points`. Na začátku je funkcí `Play.get_laser` vyslán pygame sprite Tracer, jehož pozice je uložena na konci `Play.laser_points`, s každým snímkem je list aktualizován funkcí `Play.update_laser` a vykreslená čára je tak neustále spojena s pohybujícím se Tracerem.

V případě odrazu Traceru od zrcadla jsou souřadnice středu zrcadla uloženy do listu.

V případě nárazu do stěny je Tracer smazán funkcí `Play.del_laser` a souřadnice místa, kde Tracer narazil jsou uloženy do listu.

Play.draw laser

Funkce, která vykresluje laser pomocí `pygame.draw.lines`. Volá se jen pokud je v `self.laser_points` více než jeden prvek a když je `Play.playing` `True`.

Je-li během hry laser někde přerušen (tedy je-li otočeno zrcadlo, které už bylo Tracerem „navštíveno“), je využit `Mirror.out_vector` pro určení nového vektoru Traceru. Tracer je v takovémto případě vymazán a znovu vykreslen na pozici daného zrcadla.

Časomíra

Čas, který je potřeba pro dokončení levelu je nahrán z `Play.level.countdown` do proměnné `Play.time_left` (v sekundách). Je-li `Play.time_left` roven 0, je `Play.playing` nastaven na `False`.

Play.blit time left

Funkce ovládající časomíru. Je-li `Play.time_left` větší než 0 a `Play.playing` `True`, je od něj v každém snímku odečtena konstantní hodnota, dokud není 0.

Je-li `Play.playing` `False`, pak je `Play.time_left` nastaven na původní hodnotu, tedy na `Play.level.countdown`.

Zároveň vykresluje čas do herního okna.

Kolize herních objektů

Všechny herní objekty dědí od třídy `pygame.sprite`, kolize jsou tedy implementovány buď pomocí `pygame.sprite.spritecollide`, `pygame.Rect.colliderect` nebo `pygame.Rect.collidepoint`. Zároveň jsou všechny herní objekty uloženy ve svých skupinách `sprite.group`, pro jednodušší ovládání.

Play.tracer mirror events

Funkce hlídá kolize mezi Tracerem a zrcadly (objekty třídy `Mirror`). Iteruje přes všechny objekty ve skupině `Play.level.mirror_group` a v případě, že Tracer narazí do zrcadla, je aktualizován vektor Traceru (`Play.level.tracer.vector`) pomocí `pygame.Vector2.reflect_ip` podle normálového vektoru daného zrcadla. Zároveň je aktualizován `Mirror.out_vector`, který se používá při změně zrcadla, které již bylo Tracerem „navštíveno“. Navíc je do `Play.laser_points` přidána pozice zrcadla (přidává se na předposlední index, protože na posledním indexu musí vždy být aktuální pozice Traceru).

V případě, že Tracer narazí do zdi (objekt třídy `Wall` ve skupině

`Play.level.wall_group`), je do `Play.laser_points` na předposlední index přidána pozice, kde Tracer narazil, a Tracer je smazán.

Update herních objektů

Play.mirror_update

Funkce iterující přes všechny objekty v `Play.level.mirror_group`. Ovládá proměnou `Mirror.is_selected` a `Mirror.visited`.

V případě, že je pozice myši v oblasti `Mirror.rect` a je stisknuto levé tlačítko myši, jsou nejprve u všech zrcadel nastaveny proměnné `Mirror.is_selected` na `False` a následně je u daného zrcadla nastavena proměnná `Mirror.is_selected` na `True`. Zároveň aktualizuje proměnou `Mirror.visited` u každého zrcadla, jehož souřadnice nejsou v `Play.laser_points` na `False`.

V případě, že je `Play.playing` `False`, jsou u všech zrcadel proměnné `Mirror.is_selected` nastaveny na `False`.

Play.mirror_reset

Funkce iterující přes všechny objekty v `Play.level.mirror_group`. U každého zrcadla nastaví `Mirror.normal_vector` na počáteční hodnotu, tedy `(0,1)`.

Play.target_update

Funkce iterující přes objekty v `Play.level.target_group`. Hlídá kolize Traceru s Targetem (objekt třídy `Target`) pomocí `pygame.rect.colliderect`.

V případě kolize se do `Play.laser_points` vloží na předposlední index souřadnice Targetu a je zavolána funkce `Play.del_laser`.

Pokud jsou souřadnice Targetu v `Play.laser_points`, je `Play.playing` nastaven na `False` a `Play.state` je nastaven na „level_fin“.

Update herní smyčky

Play.update

Funkce sjednocuje ostatní update funkce.

Pokud v dané chvíli existuje Tracer, je updatován i Tracer a Target.

Play.run

Funkce sjednocuje vykreslující funkce a event handling.

Zároveň načítá pozici myši a kontroluje stisk tlačítka myši.

Třída Level

Levely se načítají ze souboru [levels.json](#). V případě, že budete doplňovat vlastní levely, dodržte prosím formát, ve kterém jsou zapsány ostatní levely, jinak hrozí, že nastane chyba.

Levely jsou tile-based, přičemž jeden tile má pevnou velikost 32x32 pixelů.

Třída vytváří `pygame.sprite` skupiny pro herní objekty a načítá textury levelu, přičemž každému hernímu objektu alokuje jeho pozici v herním okně podle jeho id v listu „data“ v daném JSON souboru.

Proměnná `Level.source_id` určuje směr, kterým je objekt `Source` otočen a směr, kterým je na začátku posílán `Tracer`.

Proměnná `Level.countdown` určuje čas k dokončení daného levelu.

Proměnná `Level.max_level` obsahuje informaci o počtu levelů v souboru `levels.json`.

Důležité je, že objekty `Wall` jsou rozděleny do dvou skupin, přičemž skupina

`Level.texture_wall_group` obsahuje objekty, u kterých není potřeba kontrolovat kolize.

Level.load_tracer(*position*, *vector*)

Funkce vytváří objekt `Tracer` na pozici dané parametrem `position` a s vektorem daným parametrem `vector`.

Herní objekty

Všechny herní objekty dědí od `pygame.sprite.Sprite`. Využívají funkci `get_texture` ze souboru `settings.py`, která načítá texturu objektu ze složky `textures`.

Třída `Wall`

Parametr `position` určuje pozici objektu v rámci levelu.

Parametr `wall_img` určuje texturu, která se má načíst.

Třída `Source`

Parametr `position` určuje pozici objektu v rámci levelu.

Parametr `direction` udává orientaci textury v levelu.

Proměnná `Source.id` přebírá parametr `direction` a tedy určuje orientaci textury.

Proměnná `Source.laser_direction` určuje počáteční vektor `Traceru`.

Objekt `Source` je zdroj laseru, resp. `Traceru` při startu levelu.

Třída **Mirror**

Parametr `position` určuje pozici objektu v rámci levelu.

List `Mirror.vector_set` obsahuje všechny možné vektory, podle kterých zrcadlo odráží Tracer.

Proměnná `Mirror.normal_vector` určuje aktuální vektor, podle kterého se Tracer odráží.

Proměnná `Mirror.out_vector` ukládá vektor pod kterým do zrcadla dorazil Tracer a používá se k vypočítání vektoru nového Traceru při otočení už navštíveného zrcadla.

Proměnná `Mirror.is_selected` určuje, jestli je zrcadlo označené nebo ne.

Mirror.update

Parametr `where` udává, na které vrstvě (`pygame.Surface`) se zrcadlo aktualizuje.

Funkce kontroluje, jestli je zrcadlo označeno. Pokud `Mirror.is_selected` je `True`, je kolem zrcadla vykresleno ohraničení.

Zároveň aktualizuje texturu zrcadla podle aktuální hodnoty `Mirror.normal_vector`.

Mirror.rotate cw

Funkce otáčí označené zrcadlo po směru hodinových ručiček a mění tak

`Mirror.normal_vector`.

Mirror.rotate ccw

Funkce otáčí označené zrcadlo proti směru hodinových ručiček a mění tak

`Mirror.normal_vector`.

Mirror.rotate out cw

Funkce otáčí `Mirror.out_vector` podobně jako funkce `Mirror.rotate cw`.

Mirror.rotate out ccw

Funkce otáčí `Mirror.out_vector` podobně jako funkce `Mirror.rotate ccw`.

Třída **Tracer**

Parametr `position` určuje pozici objektu v rámci levelu.

Parametr `vector` určuje počáteční vektor Traceru.

Objekt Tracer slouží jako „průzkumník“ – je vyslán ze Source a je za ním vykreslován laser.

Tracer.update

Funkce se stará o pohyb Traceru.

Třída **Target**

Parametr `position` určuje pozici objektu v rámci levelu.

Objekt Target je „cíl“ do kterého když dorazí Tracer, je level ukončen.

Třída **Template**

Parametry `width` a `height` slouží k výpočtu polohy objektu v herním okně. Měly by to být aktuální hodnoty herního okna.

Objekt Template je využíván jen pro svoji pozici v herním okně, na jeho místo je vykreslováno informační okno.

Settings.py

Soubor *settings.py* obsahuje pomocné funkce:

get_texture

Funkce nahrává textury ze složky *textures*.

get_level

Funkce nahrává JSON soubory levelů ze složky *levels*.

Kromě nich obsahuje ještě označení některých barev, které jsou použity v ostatních souborech.

Soubor levels.json

Klíče a jejich účel:

Data

Listový zápis rozložení levelu. Prvky listu jsou id příslušných textur.

GridWidth, GridHeight

Udávají šířku a výšku levelu, musí se shodovat s rozložením v `data`, jinak se level může načíst špatně nebo může vyskočit vyjímka.

TileSize

Určuje velikost tiles. Hra je nastavena na tiles o velikosti 32x32 pixelů. Jiná hodnota nebude fungovat.

SourceId

Určuje prostorovou orientaci objektu Source v levelu a kam bude na počátku levelu směřovat Tracer.

1 znamená, že Tracer vyletí doprava.

2 znamená, že Tracer vyletí nahoru.

3 znamená, že Tracer vyletí doleva.

4 znamená, že Tracer vyletí dolů.

Countdown

Určuje čas (v sekundách) potřebný k dokončení levelu.