**Co je pro nás nového?**

Určitě jste si všimli, že pár věcí jsme se ještě neučili. Jedná se převážně o tato témata.

**Kreslení textu**

Pokud si chceme na obrazovku vykreslit text, musíme si nejdříve vytvořit písmo.

**Systémové písmo**

Pokud chceme použít **systémové písmo**, použijeme:



, kde font\_name je jméno nainstalovaného písma a size je poté velikost v pixelech.

Seznam všech dostupných systémových písem získáme pomocí pygame.font.get\_fonts().

**Vlastní písmo**

Pokud nechceme použít nějaké již nainstalované písmo, ale místo toho raději nějaké **vlastní písmo**, které sami přibalíme k našemu programu, nahrajeme jej pomocí:



Parametr size zůstává stejný, ale místo jména systémového písma nyní vkládáme cestu k .ttf souboru s naším písmem. Pokud jako path vložíme None, použije se výchozí písmo.

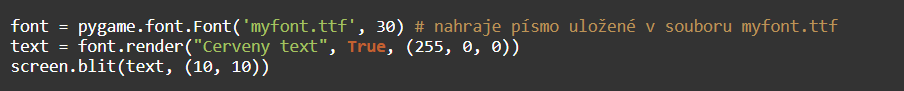
**Vykreslení písma**

Jakmile máme takto vytvořené písmo, vyrenderujeme jej pomocí:



, kde txt je string, který chceme vyrenderovat; True je parametr pro antialiasing, který zaoblí rohy písma *(což je obecně žádoucí)* a color je barva písma.

Nyní bychom měli v proměnné text uložený povrch, který nám ještě zbývá vykreslit na obrazovku. Takže pokud bychom chtěli vykreslit červený text na pozici (10, 10), mohl by náš kód vypadat nějak takto:

**Zachování velikosti při libovolné velikosti obrazovky**

Doteď jsme si vždy vytvořili proměnnou screen = pygame.display.set\_mode(...). Takto jsme museli nastavit pevné rozměry, nebo zjišťovat rozměry při vykreslení na celou obrazovku. Mnohem snadnější přístup by ale byl, pokud bychom mohli vykreslovat pouze pro jedno rozlišení a pak celou obrazovku automaticky převést na jiné. Tohoto efektu můžeme dosáhnout následovně:

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.**Pohybování obdélníky**

Pro kreslení obdélníků jsme zadali souřadnice x a y, čímž jsme v každém cyklu vytvořili nový obdélník na nové pozici. Samotný [obdélník](https://www.pygame.org/docs/ref/rect.html) v pygame je ale mnohem mocnější. Pro začátek nám bude stačit znát funkci rect.move\_ip(dx, dy), která posune obdélníkem o dx pixelů doprava a o dy pixelů dolů. Existuje také funkce rect.move(dx, dy), která ale místo posunutí původního obdélníku vrátí nový obdélník s posunutím.

**Kolize obdélníků**

Obdélníky v pygame se za nás také postarají o zjištění kolizí, tedy zda se střetávají s nějakým jiným objektem. Hlavní funkce, které by nás mohly zajímat, jsou:

* rect.collidepoint((x, y)) - vrací True, pokud se bod (x, y) nachází uvnitř obdélníku
* rect.colliderect(rect2) - vrací True, pokud mají dva obdélníky průnik
* rect.collidelist(lst) - vrací True, pokud alespoň jeden z obdélníků uvnitř seznamu má průnik s hlavním
* rect.collidelistall(lst) - vrací True, pokud všechny z obdélníků uvnitř seznamu mají průnik s hlavním

**Příprava pygame**

Řekněme, že hru budeme připravovat na rozlišení 1920x1080, ale bude spustitelná s rozlišením libovolným. Proto si vytvoříme oddělené proměnné display a screen. Zároveň by se nám asi líbilo, aby okno mělo nějaký titulek, který by reprezentoval naši hru. K tomu slouží pro nás nová metoda pygame.display.set\_caption(name), která nastaví titulek okna na hodnotu name.

Poslední věc, která by se nám mohla hodit, je proměnná round\_tick. Tu si vytvoříme a budeme ji inkrementovat každým krokem během kola, bude nám fungovat jako časovač. Toho využijeme například, když budeme chtít zrychlit míč každých pár sekund.

**Třída Game**

Naše hry by měly mít objektovou architekturu a ideálně by všechny objekty ve hře měly být instance samostatných tříd. Pygame nám k tomu i nabízí několik tříd, ze kterých můžeme své objekty oddědit.

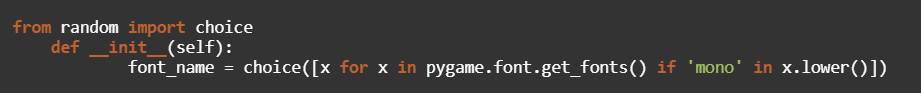
Jelikož je to naše první pygame hra, strukturu souboru si trochu zjednodušíme, nicméně se bude stále jednat o objekt. Celou hru implementujeme jako třídu Game, ve které budeme nastavovat všechny její atributy. To hlavně abychom se vyhnuli zbytečnému předávání proměnných funkcím hry přes parametry nebo dokonce přes nebezpečné globální proměnné. Takto budou ze všech metod třídy dostupné. Třída Game bude nyní vypadat takto:

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.**Příprava písem**

Vždy je dobrý nápad nemíchat mnoho druhů písem v jedné aplikaci. Proto si vybereme jedno písmo, které budeme používat. Jelikož budeme kreslit skóre, u kterého je žádoucí, aby pozice jednotlivých znaků zůstala vždy stejná, mohli bychom dojít k tom, že se nám nejvíce hodí písmo s pevnou šířkou znaků (*monospace*).

Pokud bychom ale byli nerozhodní, které z těch všech nainstalovaných písem si máme vybrat, mohli bychom za nás nechat vybrat takové písmo náš program podle toho, zda název písma obsahuje text 'mono':

*Známe-li přímo jméno písma, které chceme použít, upravíme tento kód podobně, jako jsme si ukázali v minulé lekci.*

Pak nám již zbývá pouze vytvořit jednotlivá písma. Budeme určitě potřebovat jedno menší, na stálé zobrazování skóre a druhé větší, na vypisování výzev a informací:

Obsah obrázku Písmo, text, rukopis, kaligrafie

Obsah vygenerovaný umělou inteligencí může být nesprávný.

**Příprava herních objektů**

Základní pygame prostředí máme nachystané, přejděme k herním objektům.

**Hráči**

V naší hře budou dva hráči. První hráč bude vlevo a druhý vpravo. Oba se budou moci pohybovat nějakou rychlostí. Zároveň, jelikož jsou oba hráči obdélníky, byla by škoda je rovnou jako pygame obdélníky nevytvořit, když k tomu máme připravenou třídu. Každý hráč také bude mít skóre a u hráče číslo 2 budeme chtít vědět, zda-li místo něj hraje počítač:

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.**Míč**

Podobně jako hráče si vytvoříme také míč. Ten se ale, na rozdíl od hráčů, kde směr pohybu určuje vstup od uživatele, může pohybovat libovolným směrem. Budeme si tedy muset uložit jeho současný směr. Míč se také postupem času může zrychlovat až do určité maximální rychlosti:

Obsah obrázku text, Písmo, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.Abychom si usnadnili výpočty a zaručili, že míč nebude mít nikdy hodnotu směru mimo rozsah, napíšeme si rovnou metodu, pomocí které budeme jeho směr nastavovat:

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.**Hranice obrazovky**

Poslední objekty, které budeme vytvářet, sice nebudou viditelné, ale přesto jsou vysoce důležité. Jedná se o hranice obrazovky, které si vytvoříme jako pygame obdélníky. Díky tomu pak nebudeme muset porovnávat souřadnice každého dalšího objektu, jestli náhodou neopustil obrazovku, ale místo toho nám bude stačit porovnat kolizi dvou obdélníků. Tyto obdélníky budou vždy pokrývat celou jednu hranu herní plochy, to nás asi nepřekvapí. Co by nás ale mohlo překvapit, je jejich tloušťka. Ta musí být minimálně tak velká jako maximální rychlost míče, jinak by se totiž mohlo stát, že by míč při vyšších rychlostech naše okraje prostě přeskočil. Tento jejich přesah promítneme za hranice viditelné části herní plochy:

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.A máme hotovo! Tedy alespoň máme hotovy naše objekty.

**Herní cyklus**

Pokračovat budeme tím, že si vytvoříme klasický herní cyklus, který jsme v předchozích lekcích dělali již několikrát:

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.Tento náš základní herní cyklus běží rychlostí maximálně FPS snímků za sekundu. V každé své iteraci zkontroluje, jestli někdo nepoužil klávesu Q, aby se hra vypnula, popř. její vypnutí nebylo vyžádáno jinak. Pokud ne, tak vykreslí plochu screen na obrazovku.

https://www.itnetwork.cz/python/pygame/pygame-pong-priprava