GYMNÁZIUM JÍROVCOVA 8

MATURITNÍ PRÁCE

Tetris

Kateřina Finková
vedoucí práce: Dr.rer.nat. Michal Kočer

V Českých Budějovicích 14. února 2023 **2022/2023**

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně s vyznačením všech použitých pramenů.

V Českých Budějovicích dne ..... podpis .....

Kateřina Finková

Abstrakt

Tato práce se zabývá počítačovou hrou Tetris, jednou z nejznámějších počítačových her minulého století. V teoretické části se zaměřuji na historii Tetrisu, konkrétně na jeho rozšíření po světě a boj o jeho vlastnictví. V praktické části pak programuji Tetris v programovacím jazyce Python a popisuji vytvořený program.

Klíčová slova

Tetris, Počítačové hry, Programování počítačových her, Python, Pygame

Poděkování

dopsat

Obsah

Úvod

dopsat

Část I

Historie Tetrisu

1 Počátky Tetrisu

Tetris je počítačová hra vytvořená Alexejem Pažitnovem v roce 1984. Pažitnov se inspiroval hlavolamem Pentonimo. Hlavolam je složen z dřevěných dílků a krabičky, do které hráč umisťuje dílky tak, aby se tam všechny vešly. [2]

Alexej pojmenoval dílky tetramino podle slova tetra znamenající čtyři, protože každý dílek se skládá ze čtyř bloků. Svou hru pojmenoval Tetris spojením slov tetra a tenis.[2]

1.1 Alexej Pažitnov

Když mu bylo 17, poprvé se dotkl počítače. I když měl talent pro vědu a matematiku, programování mu napřed moc nešlo. I přesto ale viděl počítač jako svou budoucnost.[1]

Alexej původně pracoval v Moskevském institutu letectví, kde se s ostatními vědci střídal o počítač. Byla to velmi slibná a stabilní kariéra, ale rozhodl se odejít na Ruskou akademii věd do Dorodnitsynova výpočetního centra, kde později dostal svůj vlastní počítač Electronica 60. [2]

1.2 Výroba Tetrisu

Pažitnov spolupracoval s Dmitrijem Pavlovským, kolegou a přítelem z centra. Pavlovsky k nim přizval Vadima Gerasimova, teenagera, který byl ale velmi zkušený v programování. Přemýšleli nad tím, že by mohli prodávat balíček videoher, Pažitnovem pojmenovaným Computer FunFair, ale v té době to v Sovětském svazu nebylo možné. Místo toho se rozhodli zlepšovat své schopnosti. Snažili se spolu vytvářet klony her ze západu a zkoušeli přidávat zvuk, text a grafiku. Pod vedením Pavlovského vytvořili Antix, klon hry Xonix[1]. Hra se

[1]Xonix - hra, ve které se pohybováním kosočtverce vytváří pole, do kterého už kosočtverec znovu nesmí.

rychle rozšířila mezi ostatní pracovníky centra a i na některé domácí počítače v Moskvě.[1]

Alexej se rozhodl Gerasimovovi ukázat svůj nejnovější projekt - Tetris. Byl to ale jen prototyp, Gerasimov mohl pohybovat tetraminy a vytvářet nové obrazce, ale cíl hry nebyl úplně jasný. Moc ho to nezaujalo, ale Pajitnov věděl, že to ještě není dokonalé. O pár měsíců později navrhl svým spolupracovníkům Tetris znovu, ale tentokrát už měla tetramina padat do úzké hranaté zasklené plochy a hromadit se dole. [1]

Pažitnov sám dopracoval Tetris a dál přemýšlel nad tím, že by hru prodával. Avšak Tetris nebyl ještě dokonalý. Hra neměla skórování, příběh, postavy, dobrou grafiku ani zvuk. Hra byla moc abstraktní a zaujala by jen pracovníky výpočetního centra. Dalším problémem bylo, že hra byla určena pro počítač Electronica 60, který byl napodobeninou v té době velmi zastaralého západního počítače, tudíž by ani moc lidí nemělo možnost si Tetris zahrát. [1]

Naštěstí Vadim Gerasimov uměl dobře s Microsoft DOS, operačním systémem, který běžel na v té době populárním IBM PC. Tento počítač uměl ukázat jednoduchou grafiku a barvy. Gerasimovovi se za pár dní podařilo program přepsat a Tetris se dál zlepšoval. Mezitím se hra rozšířila po výpočetním centru. [1]

Po dvou měsících byl Tetris už v takovém stavu, že byl podle Pažitnova čas ho rozšířit mezi širší publikum. Ale kdyby ho chtěli prodávat, Tetris by patřil Akademii věd, sami by software distribuovat nemohli. Proto nahráli Tetris na diskety a hra se rychle stala v Moskvě velmi populární.[1]

Vladimír Pokhilko, vědecký pracovník klinické psychologie v Moskevském zdravotním středisku, byl jedním z prvních, co hru dostali mimo výpočetní centrum. On a jeho spolupracovníci si hru tak zamilovali, že byl nucen všechny diskety vyhodit. Hru pak používal pro testování pacientů, později s Pažitnovem a Gerasimovem vytvořili Tetris pro dva hráče. Chtěl totiž pozorovat, jak budou hráči interagovat mezi sebou a se hrou.[1]

2 Cesta do světa

Hra zaujala spoustu lidí včetně Pažitnova nadřízeného, Viktora Brijabrina. Také si myslel, že by se Tetris mohl rozšířit dál, a tak poslal hru kolegům v Maďarsku. Na Institutu výpočetní techniky v Budapešti se Tetris rozšířil stejně rychle jako v Moskvě. [1]

V Maďarsku byl představen na exhibici a padl do oka Robertu Steinovi, vlastníkovi britské firmy Andromeda software. Stein se rozhodl kontaktovat technický institut, kde Alexej pracoval, aby mohl získat práva pro distribuci hry. Lidé z institutu ho pozvali do Moskvy, kde se s ním dohodli, ale smlouva ještě nebyla uzavřena. [2]

Licenci od Steina získalo americké Spectrum Holobyte. Jelikož zájem americké společnosti o Sovětský svaz díky jednání Reagana a Gorbačova rostl, řediteleSpectrum Holobyte, Gilmana Louieho, napadlo hru propagovat hlavně tím, že pochází ze Sovětského svazu. Tato reklama se objevila na jedné celé straně novin USA Today. [2]

Robert Stein ale měl problém - licenci ke hře ještě oficiálně nevlastnil, avšak ji už prodal Spectrum Holobyte a britské firmě Mirrorsoft, které patřili druhému nejmocnějšímu vlastníkovi novin, Robertovi Maxwellovi. Potom, co se na hru objevila reklama v novinách, byl velmi motivován licenci oficiálně získat. [2]

2.1 Jednání o licenci

Mezitím vznikla v Sovětském svazu agentura Elektronorgtechnica (zkráceně ELORG), která měla za úkol kontrolovat import a export softwaru a hardwaru. Ta nově začala vyjednávat se Steinem. Robert letěl opět do Moskvy, jednání trvala několik dní. Smlouva byla uzavřena ale až později, v květnu roku 1988. Tetris se mezitím stal nejprodávanější hrou v USA a Velké Británii a získal dvě ocenění od The Software Publishers Association. [2]

Mirrorsoft a Spectrum Holobyte přidali do hry stereotypní sovětské obrázky. Také přidali do hry letadlo Matthiase Rusta, který přistál na Rudém náměstí v Moskvě, který ale byl

Sovětském svazu považován za radikála. Stein tedy slíbil, že letadlo bude odstraněno. Nechtěl totiž o Tetris přijít, protože věděl, že mu může ještě vydělat spoustu peněz. [2]

Stein sdělil Mirrorsoftu, že má pro ně licenci na videoherní automaty. Mirrorsoft ji hned poté slíbil v Americe firmě Atari a v Japonsku firmě Sega, ale licenci ve skutečnosti nikdo z nich neměl. Stein se sice potkal v Paříži s ředitelem Elektronorgtechnica, ale zde byl pouze nucen zaplatit penále za nezaplacené licenční poplatky, další licenci nedostal. [2]

3 Henk Rogers

Henk Rogers z Bulletproof Software se o Tetrisu dozvěděl v roce 1988 na akci v Las Vegas. Jelikož měl dobré kontakty v Japonsku, začal ihned jednat o licenci pro diskety a domácí konzole s Gilmanem Louiem. Louie mu licence slíbil, ale pak zjistil, že licenci k domácím konzolím v Japonsku už prodal Mirrorsoft firmě Sega.[2]

Nakonec vše vyřešil syn Roberta Maxwella, Kevin. Bulletproof Software získali práva na diskety a Atari získalo práva na vše ostatní. Všechno bylo dohodnuto bez jakéhokoli vědomí Elektronorgtechnica, která nedala tato práva nikomu, Stein si pouze myslel, že je vlastní.[2]

Henk byl ale rozhodnutý získat víc, a tak navštívil Atari. Po večeři s jejich ředitelem získal licenci pro Famicom - domácí herní konzoli od japonské firmy Nintendo, v Americe známé jako Nintendo Entertainment System. Stejně jako v Severní Americe a v Evropě měl Tetris v Japonsku obrovský úspěch. [2]

Elorg mělo ale smlouvu pouze s Andromeda Software a jen o licenci pro domácí počítače. Vůbec ještě netušili o úspěších Tetrisu v zahraničí a ani o tom, že se prodává i v jiném formátu než pro domácí počítač. Pažitnov o tom také nic nevěděl a už vůbec za to nebyl nijak finančně kompenzován. [2]

4 Tetris pro Game Boy

Jedním z nejmocnějších fanoušků Tetrisu byl Minoru Arakawa, ředitel Nintenda. Nintendo zrovna vytvořilo novou přenosnou herní konzoli - Game Boy. Arakawa viděl obrovský potenciál ve spárování Gameboye s tak návykovou hrou jako je Tetris. Kontaktoval Henka Rogerse, aby mu zajistil licenci pro přenosné konzole. Tu už se snažil tou dobou získat Stein.[2]

Rogers si se Steinem vyměnili několik faxů, Stein neustále sliboval, že poletí vše stvrdit do Moskvy, ale k odletu se pořád neměl. Henk se rozhodl, že do Moskvy poletí sám, i když tam vůbec nikoho neznal. Kromě něj se tam vydal Kevin Maxwell, v Mirrorsoftu totiž přišli na to, že tahle práva zatím nikomu nepatří. Samozřejmě se do Moskvy vydal i Robert Stein.[2]

Henk byl v Sovětském svazu poprvé, nikoho tam neznal a ani nevěděl, kde se ředitelství Elektronorgtechnika v Moskvě nachází. Město bylo celé šedé, v hotelu nefungovala televize a do restaurace musel mít rezervaci. Když se zeptal na recepci, kde může najít ELORG, odpovědí mu byl jen vyděšený obličej. Bylo podezřelé se ptát, kde se nachází státní instituce. [1]

Ale to ho neodradilo. Když začínal v Japonsku, podařilo se mu navázat kontakt s prezidentem Nintenda tak, že mu napsal, že dokáže pro Famicom vytvořit deskovou hru Go - strategickou hru, kde je úkolem hráče zajmout kamínky protihráče - s dostatečně komplexní AI. [1]

V SSSR byla tato desková hra populární, a proto se rozhodl najít v Moskvě asociaci hráčů Go. Sice anglicky hráči neuměli, ale udělalo na ně dojem, že si chtěl zahrát s jedním z nejlepších sovětských hráčů. Díky tomu byl schopen se spojit s průvodkyní, která ho dovedla k jeho cíli. Varovala ho, že bez pozvání se tam nedostane.[1]
Nikolajevič Belikov byl v Elorgu postaven do čela vyjednávání ohledně Tetrisu. Byl připraven se potkat se Steinem a Kevinem Maxwellem v ten samý den. Vůbec nevěděl, že Henk Rogers je také v Moskvě. [2]

První schůzku měl Belikov s Henkem. Henk zde zjistil, že práva k Tetrisu mu vůbec

nepatří, přesto se snažil Belikova přesvědčit, aby mu Elorg práva pro Nintendo prodalo. Belikov věděl, co nintendo znamená (!) a rozhodl se tajně upravit smlouvu se Steinem, kde doplnil, že Steinova práva se netýkají herních konzolí Nintenda. Také zvedl pokuty za nedodržování smlouvy, aby od jeho ostatních úprav odlákal pozornost. [2]

Další schůze byla s Maxwellem. Belikov před něj postavil Henkovu kopii hry (o které do toho dne nevěděl) a zeptal se ho, proč je na ní jméno Mirrorsoftu, když k Tetrisu Mirrorsoft práva nemá. Maxwella to rozhodilo natolik, že zapomněl, že práva ke hře skrz smlouvu se Steinem má a byl nucen odletět pryč, aby vymyslel novou nabídku se svým týmem.[2]

Belikov si nebyl jistý, jak Stein na novou smlouvu zareaguje, ale Stein si ničeho nevšiml a smlouvu podepsal. Díky tomu byly všechny verze Tetrisu, které nebyly pro domácí počítač, nelegální.[2]

Rogers se rychle vrátil do Moskvy s novou nabídkou. Rozhodl se ale ještě před tím najít Pažitnova a rychle se z nich stali přátelé. Henk opustil Moskvu s právy pro Nintendo a novou možností získat globální práva k Tetrisu. Arakawa v tom viděl možnost, jak vyhrát nad Atari.[2]

5 Boj o globální licenci

Henk byl poslán zpět do Moskvy s novou smlouvou a právníkem. Mirrorsoft mělo pouze den na to, aby přišli s jejich nabídkou. Vložil se do toho Robert Maxwell a kontaktoval všechny své politické známosti. Belikov kvůli tomu začal být tlačen do smlouvy s Mirrorsoftem.[2]

Atari se rozhodlo zažalovat Nintendo. V Atari si mysleli, že vyhrají a začali svou verzi Tetrisu propagovat. Nakonec ale vyhrálo Nintendo a Henk mohl uzavřít smlouvu s Elorgem ohledně globálních práv k Tetrisu.[2]

doplnit

6 Pažitnov v Americe

Tetris se stal ještě populárnějším. Jeho prodeje bořily rekordy. Alexej Pažitnov ale pořád za svou hru žádné peníze nedostal. Henk se ale stal velmi bohatým. Pomohl Pažitnovi a jeho příteli Vladimírovi s jejich rodinami se odstěhovat v roce 1991 do Ameriky. Alexej si zde s Vladem založili herní firmu Animatek.[2]

V roce 1996 začal pracovat jako herní vývojář pro Microsoft. V tomto roce se také rozpadla Elektronorgtechnica. Pažitnov měl konečně šanci získat zpět Tetris, ihned s Henkem odletěli zpět do Moskvy. Vytvořili spolu The Tetris Company a práva k Tetrisu se konečně vrátila do správných rukou.[2]

Část II

Klon tetrisu

7 Začátek programu

Program začínám importováním modulů pygame a random. Modul pygame potřebuji ke kreslení hry a k jejímu ovládání. Modul random v programu využívám k náhodnému generování tetrimin.

Po importování modulů obsahuje program seznam barev. Tyto konkrétní barvy program používá k vybarvení tetrimin. Každá barva je vždy RGB kód, tady seznam tří čísel.

```
1  import pygame
2  import random
3
4  barvy = [
5      (0, 0, 0),
6      (251, 248, 204),
7      (255, 207, 210),
8      (241, 192, 232),
9      (207, 186, 240),
10     (163, 196, 243),
11     (180, 243, 248),
12     (185, 251, 192)
13 ]
```

Zdrojový kód 7.1: Importování modulů a tvorba barev tetrimin

8 Třída Tetrimino

Třída je skupina různých objektů s podobnými vlastnostmi. Třída Tetrimino vytváří herní dílek. Jako první definuji tvary jednotlivých dílků a jejich možné rotace. Dílky vlastně zakresluji do pomyslené tabulky se čtyřmi sloupci a čtyřmi řádkami. Každá buňka této tabulky je označena číslem od nuly do patnácti. Jednotlivé dílky jsou tudíž seznamy čísel buněk, které překrývají. Každou rotaci dílku vložím do jednoho seznamu, tyto seznamy vložím do seznamu tvary.

```
class Tetrimino(object):
    tvary = [
        [[1,5,9,13],[4,5,6,7]], #I_tetrimino
        [[4,5,9,10],[2,6,5,9]], #Z1_tetrimino
        [[6,7,9,10],[1,5,6,10]], #Z2_tetrimino
        [[1,2,5,9],[0,4,5,6],[1,5,8,9],[4,5,6,10]],
        #L1_tetrimino
        [[1,2,6,10],[5,6,7,9],[2,6,10,11],[3,5,6,7]],
        #L2_tetrimino
        [[1,4,5,6],[1,4,5,9],[4,5,6,9],[1,5,6,9]], #T_tetrimino
        [[1,2,5,6]] #O_tetrimino
```

Zdrojový kód 8.1: Tvary tetrimin

0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
4	5	6	7	4	5	6	7	4	5	6	7	4	5	6	7
8	9	10	11	8	9	10	11	8	9	10	11	8	9	10	11
12	13	14	15	12	13	14	15	12	13	14	15	12	13	14	15

Tabulka 8.1: Všechny možné rotace tetrimina L1

8.1 Konstruktor

Konstruktor třídy je část kódu třídy, kde jsou všechny údaje o objektu patřícím do této třídy. V konstruktoru třídy Tetrimino jsou souřadnice x a y, které udávají místo na herní ploše, odkud se načítá tetrimino (buňka nula v naší tabulce). Self.typ udává tvar dílku. Je to číslo, pod kterým je schovaný seznam rotací v seznamu tvary. Jelikož je každý tvar tetrimina náhodný, tak použiji příkaz random.randint a do závorky napíši, z jakého rozpětí čísel má číslo tvaru být, tedy on nuly až do délky seznamu tvary. Od délky seznamu musí být odečtena jednička, protože položky seznamu jsou číslovány od nuly, jinak by se mohlo vygenerovat číslo tvaru, které už v seznamu není.

Self.barva je barva tetrimina. Protože chci, aby každý dílek měl svou příslušnou barvu, tak se barva musí rovnat hodnotě self.typ. Jelikož ale políčka v herní ploše, která neobsahují žádné tetrimino, mají hodnotu nula, tak se self.barva nemůže rovnat nule, neboť by pro ostatní dílky dílek neexistoval a mohly by jím volně procházet. Proto je hodnota self.barva zvětšeno o 1, aby k tomuto problému nedocházelo. Self.r značí otočení dílku. Self.r základního tvaru dílku je vždy rovno nule.

```
1     def __init__(self, x, y):
2         self.x = x
3         self.y = y
4         self.typ = random.randint(0, len(self.tvary) - 1)
5         self.barva = self.typ + 1
6         self.r = 0
```

Zdrojový kód 8.2: Konstruktor třídy Tetrimino

8.2 Funkce třídy Tetrimino

Funkce okoli zjišťuje, která část tabulky je zaplněna tetriminem. Když použiji tuto funkci, tak vypíše, které buňky v tabulce dílek překrývá. Funkce to udělá tak, že vybere ze seznamu tvary seznam rotací podle toho, kolik je self.typ. Poté vybere ze seznamu rotací tetrimina tvar podle toho, jakou má tetrimino self.r.

```
    def okoli(self):
        return self.tvary[self.typ][self.r]
```

Zdrojový kód 8.3: Funkce okoli

Dále musím nadefinovat otáčení tetrimina. Když chci otočit dílek doprava, tak přičtu k self.r jedničku, to pak vydělím počtem tvarů otočení dílku. Podle toho, jaké vznikne z tohoto dělení modulo pak zjistím, jak se má tetrimino otočit. To samé udělám při otočení doleva, ale místo přičtení od self.r odečtu.

```
    def otoc_R(self):
        self.r = (self.r + 1) % len(self.tvary[self.typ])

    def otoc_L(self):
        self.r = (self.r - 1) % len(self.tvary[self.typ])
```

Zdrojový kód 8.4: Otáčení tetrimina

9 Třída Tetris

Druhou třídou mého programu je třída Tetris. Tato třída obsahuje všechny potřebné funkce pro běh hry. Napřed si předdefinuji některé proměnné. Proměnná highscore si pamatuje nejlepší výsledek hráče. Pamatuje si jej ale jen během jednoho zapnutí, když hráč hru vypne a znovu zapne, nastaví se highscore opět na nulu. X a y jsou souřadnice bodu, kde začíná herní plocha. Zoom je zvětšení, které používám v programu jako velikost strany jednotlivých čtverců herní plochy. Proměnná tetrimino označuje momentální padající dílek, který zatím není vygenerován.

```
1 class Tetris(object):
2     highscore = 0
3     x = 100
4     y = 60
5     zoom = 20
6     tetrimino = None
```

Zdrojový kód 9.1: Proměnné třídy Tetris

9.1 Konstruktor

Self.height a self.width jsou výška a šířka hrací plochy. Self.pole je výpisem všech poloh na hrací ploše, jedničky představují zaplněná místa a nuly místa prázdná. Self.score je momentální skóre hráče, které je na začátku hry nula. Self.status je momentální stav hry. Stav "game" je počáteční stav.

```
1     def __init__(self, height, width):
2         self.height = height
3         self.width = width
4         self.pole = []
5         self.score = 0
```

```
6           self.status = "game"
```

Zdrojový kód 9.2: Konstruktor třídy Tetris

V konstruktoru také vytváříme herní plochu. Podle toho, jak je herní plocha vysoká, program vygeneruje jednotlivé řádky new_ line. Podle toho, jak je herní plocha široká, se vloží do každé new_ line nuly. Každá tato řádka se pak uloží do self.pole.

```
1           for i in range(height):
2               new_line = []
3               for j in range(width):
4                   new_line.append(0)
5               self.pole.append(new_line)
```

Zdrojový kód 9.3: Tvorba herní plochy

9.2 Funkce třídy Tetris

Funkce new_tetrimino má na starosti generování nového tetrimina. Nový dílek se vždy objeví na souřadnicích [3,0].

```
1       def new_tetrimino(self):
2           self.tetrimino = Tetrimino(3, 0)
```

Zdrojový kód 9.4: Vytvoření nového tetrimina

Funkce prekryti zjišťuje, zda tetrimino, která padá dolů, překrývá jiný dílek nebo jestli je mimo herní plochu. Pokud dílek nic nepřekrývá, pak má proměnná kryt hodnotu False. Souřadnice x a y jsou levý horní roh mřížky, kterou zaplňuje tetrimino. Abychom tedy zjistili překrytí v rámci celého dílku, musíme postupně kontrolovat všechny čtverečky tetrimina. Proto postupně generujeme čísla, která by mohla patřit do seznamu zjištěného pomocí funkce okoli. Za podmínky, že je vygenerované číslo v seznamu, tak aby se proměnná kryt změnila na hodnotu true, musí být splněna jedna z těchto podmínek:

- tetrimino je níž než je dolní hranice herní plochy
- tetrimino je za pravou hranicí herní plochy
- tetrimino je za levou hranicí herní plochy

- dané místo herní plochy je už zaplněno jiným dílkem

```
1     def prekryti(self):
2         kryt = False
3         for i in range(4):
4             for j in range(4):
5                 if i * 4 + j in self.tetrimino.okoli():
6                     if i + self.tetrimino.y > self.height - 1
                          or j + self.tetrimino.x > self.width - 1
                           or j + self.tetrimino.x < 0 or \
7                           self.pole[i + self.tetrimino.y][j +
                               self.tetrimino.x] > 0:
8                               kryt = True
9         return kryt
```

Zdrojový kód 9.5: Funkce překrytí

i	j	výsledek	i	j	výsledek
0	0	0	1	0	4
0	1	1	1	1	5
0	2	2	1	2	6
0	3	3	1	3	7

i	j	výsledek	i	j	výsledek
2	0	8	3	0	12
2	1	9	3	1	13
2	2	10	3	2	14
2	3	11	3	3	15

Tabulka 9.1: Generování čísel pro kontrolu překrytí.

Funkce plna_ line zjišťuje, jestli je řádek už zcela zaplněný tetriminy. doplnit

```
1     def plna_line(self):
2         lines = 0
3         for i in range(1, self.height):
4             nuly = 0
5             for j in range(self.width):
6                 if self.pole[i][j] == 0:
```

```
                    nuly += 1
            if nuly == 0:
                lines += 1
                for k in range(i, 1, -1):
                    for l in range(self.width):
                        self.pole[k][l] = self.pole[k - 1][l]
        self.score += lines
        if self.score > self.highscore:
            self.highscore = self.score
```

Zdrojový kód 9.6: Funkce plna_ line

Funkce slam posune dílek na nejnižší možné místo v hrací ploše. Dílek bude padat tak dlouho, dokud nebude něco překrývat. Pak se posune o jedna zpět, protože by jinak byl mimo herní plochu nebo překrýval jiné tetrimino. Nakonec se dílek zastaví, aby hráč mohl začít pohybovat s dalším.

```
    def slam(self):
        while not self.prekryti():
            self.tetrimino.y += 1
        self.tetrimino.y -= 1
        self.stop()
```

Zdrojový kód 9.7: Funkce slam

Funkce go_ down posouvá tetrimina dolů. Vlastně je tato funkce velmi podobná funkci slam ale s tím rozdílem, že funkce slam nechá spadnout dílek rovnou, kdyžto funkce go_ down posouvá dílek dolů postupně. Pokud dílek něco překryje tak ho také posune zpět nahoru a zastaví.

```
    def go_down(self):
        self.tetrimino.y += 1
        if self.prekryti():
            self.tetrimino.y -= 1
            self.stop()
```

Zdrojový kód 9.8: Funkce go_ down

Funkce stop zastaví pohyb dílku, když dopadne. Vybarví příslušné čtverečky tetrimina, pak zkontroluje, zda se hráči povedlo vyplnit celý řádek. Následně vygeneruje nové tetrimino, pokud zjistí, že nový dílek už překrývá jiný, změní status hry na "gameover". doplnit

```
    def stop(self):
```

```
        for i in range(4):
            for j in range(4):
                if i * 4 + j in self.tetrimino.okoli():
                    self.pole[i + self.tetrimino.y][j + self.
                       tetrimino.x] = self.tetrimino.barva
        self.plna_line()
        self.new_tetrimino()
        if self.prekryti():
            self.status = "gameover"
```

Zdrojový kód 9.9: Funkce stop

Funkce go_ side_ R a go_ side_ L hýbou dílkem doprava respektive doleva. Funkce si zapamatuje momentální souřadnici x, poté k ní přičte jedna pro posunutí doprava nebo odečte jedna pro posunutí doleva. Pokud ale dílek bude překrývat jiný nebo bude mimo herní plochu, tak vrátí dílek na svou původní souřadnici.

```
    def go_side_R(self):
        x = self.tetrimino.x
        self.tetrimino.x += 1
        if self.prekryti():
            self.tetrimino.x = x

    def go_side_L(self):
        x = self.tetrimino.x
        self.tetrimino.x -= 1
        if self.prekryti():
            self.tetrimino.x = x
```

Zdrojový kód 9.10: Funkce go_ side_ R a go_ side_ L

Funkce otoc_ R a otoc_ L otáčí dílek doprava respektive doleva. Funkce si uloží momentální rotaci dílku, pak ho otočí příslušným směrem. Když zjistí, že tímto pohybem překryje jiné tetrimino nebo dílek bude mimo herní plochu, tak nastaví r na původní rotaci.

```
    def otoc_R(self):
        old_r = self.tetrimino.r
        self.tetrimino.otoc_R()
        if self.prekryti():
            self.tetrimino.r = old_r

    def otoc_L(self):
        old_r = self.tetrimino.r
        self.tetrimino.otoc_L()
        if self.prekryti():
```

```
11                self.tetrimino.r = old_r
```

Zdrojový kód 9.11: Funkce otoc_ R a otoc_ L

10 Pygame

Pygame je modul, který se používá pro programování her v Pythonu. Abych ho mohla použít, musím tento modul po importování ještě načíst. K tomu slouží příkaz pygame.init(). Poté si vypíšu barvy, které budu dále potřebovat, každou barvu zapíši jako rgb kód. BARVA1 je šedivá, kterou použiji jako barvu pozadí. BARVA2 je černá, tou je vybarvena mřížka hrací plochy. Pro tlačítka ovládání jsou barvy 3 a 4. BARVA3 je zároveň použita i jako barva většiny textů. Text highscore a Game over je nakreslen pomocí BARVA5. Proměnná screen označuje okno, ve kterém se bude zobrazovat hra. Pro velikost okna použiji proměnnou size. Pak nastavím název okna hry na Tetris.

```
1  pygame.init()
2
3  BARVA1 = (50, 50, 50)
4  BARVA2 = (0, 0, 0)
5  BARVA3 = (161, 121, 226)
6  BARVA4 = (207, 186, 240)
7  BARVA5 = (255, 173, 179)
8
9  size = (400, 500)
10 screen = pygame.display.set_mode(size)
11 pygame.display.set_caption("Tetris")
```

Zdrojový kód 10.1: Počátek Pygame

Proměnná konec označuje, jestli hra stále běží. Pokud hráč hru vypne, tak se konec změní na True. Proměnná hodiny vytvoří hodiny pro počítání času. Pro výpočet aktualizování obrazu použiji proměnnou fps. Spuštění a upravování Tetrisu budu dělat za pomoci proměnné hry, kam zároveň napíšu i výšku a šířku potřebnou pro vytvoření herní plochy. Do proměnné pocitadlo ukládám jaký čas už uběhl. Proměnná zmacknuti je v programu použita při zaznamenávání, zda je zmáčknutá klávesa šipka dolů.

```
konec = False
hodiny = pygame.time.Clock()
fps = 25
hra = Tetris(20, 10)
pocitadlo = 0
zmacknuti = False
```

Zdrojový kód 10.2: Další proměnné pro Pygame

10.1 Počátek hlavní programové smyčky

Hlavní programová smyčka je místo v programu, kde se odehrává celý děj programu. Všechen následující kód patří do této smyčky. Pokud nebylo vygenerováno žádné tetrimino a stav hry je nastaven na ”start”, tak hra vytvoří nové tetrimino.

```
while not konec:
    if (hra.tetrimino is None) and (hra.status == "start"):
        hra.new_tetrimino()
```

Zdrojový kód 10.3: Počátek hlavní programové smyčky

10.2 Pohyb dolů

Jak už bylo výše řečeno, proměnná pocitadlo zaznamenává uběhlý čas. Kdykoliv se rozběhne smyčka, tak se k této proměnné přičte jedna. Následně řeším, kdy se má pohybot tetrimino dolů. Pokud je zbytek po dělení pocitadlo fps roven nule nebo hráč zmáčkl tlačítko šipka dolů a k tomu je stav hry ”start”, pak hra posune momentálně pohybující se dílek dolů.

```
    pocitadlo += 1
    if (pocitadlo % fps == 0 or zmacknuti) and hra.status == "
       start":
        hra.go_down()
```

Zdrojový kód 10.4: Pohyb dolů

10.3 Události

V této části smyčky program zjišťuje jestli hráč např. nezmáčkl určitou klávesu nebo jestli někam neklikl. Když hráč hru vypne neboli zavře okno s hrou, tak se konec nastaví na True

a hra se zavře. Dále ošetřuji, co program udělá, když hráč zmáčkne některé klávesy. Pokud status hry není "gameover", tak program zjišťuje, zda jsou zmáčknuty tyto klávesy:

- Klávesa D - dílek se otočí doprava
- Klávesa A - dílek se otočí doleva
- Klávesa šipka dolů - zmáčknutí se nastaví na True
- Klávesa šipka doleva - dílek se posune doleva
- Klávesa šipka doprava - dílek se posune doprava
- Klávesa S - dílek dopadne na nejnižší možné místo na herní ploše.

Pokud je status hry "gameover" nebo "menu2" a zároveň přestane být stiknuta klávesa šipka dolů, tak se proměnná zmacknuti změní na False. Program nechá padat tetrimino dál, ale pomaleji, než když hráč mačká příslušnou klávesu.

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            konec = True
        if event.type == pygame.KEYDOWN:
            if hra.status != "gameover":
                if event.key == pygame.K_d:
                    hra.otoc_R()
                if event.key == pygame.K_a:
                    hra.otoc_L()
                if event.key == pygame.K_DOWN:
                    zmacknuti = True
                if event.key == pygame.K_LEFT:
                    hra.go_side_L()
                if event.key == pygame.K_RIGHT:
                    hra.go_side_R()
                if event.key == pygame.K_s:
                    hra.slam()
            if hra.status == "gameover" or hra.status == "menu2
               ":
                if event.key == pygame.K_ESCAPE:
                    hra.__init__(20, 10)
                    hra.status = "start"
        if event.type == pygame.KEYUP:
                if event.key == pygame.K_DOWN:
                    zmacknuti = False
```

Zdrojový kód 10.5: Ošetření událostí

Když hráč někam v okně klikne, tak program zjistí, na jaké konkrétní místo hráč klikl. Za podmínky, že je stav hry roven ”menu” nebo ”game” a zároveň hráč klikl na místo, kde se v těchto dvou stavech nachází tlačítko Start, tak se stav hry změní na ”start”, což znamená, že začne padat první tetrimino. Když hráč klikne na místo, kde se nachází tlačítko Menu a stav hry je ”game”, tak se stav změní na ”menu”. Pokud je stav hry roven ”gameover”, tak se stav změní na ”menu2”.

```
        if event.type == pygame.MOUSEBUTTONDOWN:
            mys = pygame.mouse.get_pos()
            if hra.status == "menu" or hra.status == "game":
                if 5 <= mys[0] <= 70 and 30 <= mys[1] <= 50:
                    hra.status = "start"
            if 5 <= mys[0] <= 70 and 60 <= mys[1] <= 80:
                if hra.status == "game":
                    hra.status = "menu"
                if hra.status == "gameover":
                    hra.status = "menu2"
```

Zdrojový kód 10.6: Ošetření události kliknutí myší

10.4 Kreslení herní plochy

dopsat

```
    screen.fill(BARVA1)
    for i in range(hra.height):
        for j in range(hra.width):
            pygame.draw.rect(screen, BARVA2, [hra.x + hra.zoom
               * j, hra.y + hra.zoom * i, hra.zoom, hra.zoom],
               1)
            if hra.pole[i][j] > 0:
                pygame.draw.rect(screen, barvy[hra.pole[i][j]],
                    [hra.x + hra.zoom * j + 1, hra.y + hra.zoom
                    * i + 1, hra.zoom - 2, hra.zoom - 2])

    if hra.tetrimino is not None:
        for i in range(4):
            for j in range(4):
                p = i * 4 + j
                if p in hra.tetrimino.okoli():
                    pygame.draw.rect(screen, barvy[hra.
                       tetrimino.barva], [hra.x + hra.zoom * (j
```

```
                            + hra.tetrimino.x) + 1,
14                          hra.y + hra.zoom * (i + hra.tetrimino.y
                               ) + 1, hra.zoom - 2, hra.zoom - 2])
```

Zdrojový kód 10.7: Kreslení herní plochy

10.5 Nápisy

Zde vytvářím fonty pro texty, které se objevují ve hře. Font vytvářím tak, že za pomoci pygame.font.SysFont program nechám najít v počítači font Rockwell, poté zadám velikost písma a jestli bude text tučně nebo kurzívou.

```
1     font = pygame.font.SysFont('rockwell', 25, True, False)
2     font1 = pygame.font.SysFont('rockwell', 12, True, False)
```

Zdrojový kód 10.8: Fonty

Pro vytvoření textu použiji font.render. Samozřejmě napíši, co chci aby v textu stálo. Další v závorce je anti-alias. Když bude mít hodnotu False, bude mít v zaoblených částech písmen viditelné pixely, pokud bude mít hodnotu True, pixely viditelné nebudou. Protože je ale text docela přiblížený, bude při hodnotě True vypadat spíše rozmazaně, proto jsem zvolila hodnotu False. Poslední, co v závorce uvádím, je barva textu.

Pro texty, které vypysují skóre a nejlepší skóre musím zapsat text jako f-string, abych do nich mohla vypisovat čísla z k nim korespondujících proměnných.

```
1     font = pygame.font.SysFont('rockwell', 25, True, False)
2     font1 = pygame.font.SysFont('rockwell', 12, True, False)
3     text = font.render(f"Score: {str(hra.score)}", False,
         BARVA3)
4     text_start = font.render("Start", False, BARVA3)
5     text_menu = font.render("Menu", False, BARVA3)
6
7     text_controls_A = font1.render("A - turn left", False,
         BARVA3)
8     text_controls_D = font1.render("D - turn right", False,
         BARVA3)
9     text_controls_S = font1.render("S - slam", False, BARVA3)
10    text_controls_Move = font1.render("use arrows to move",
         False, BARVA3)
11
12    text_controls_New = font1.render("Esc - play again", False,
          BARVA3)
```

```
13      text_game_over = font.render("Game Over", False, BARVA5)
14      text_highscore = font1.render(f"Highcore: {str(hra.
           highscore)}", False, BARVA5)
```

Zdrojový kód 10.9: Texty

Příkazem screen.blit vložím do okna text skóre, souřadnice [5,0] označují místo v okně, odkud se bude text vytvářet.

```
1       screen.blit(text, [5, 0])
```

Zdrojový kód 10.10: Nakreslení skóre

10.6 Stavy hry

10.6.1 Gameover

Hra se nastaví do stavu gameover poté, co hráč prohraje. Nad herní plochu program napíše text Game Over. Doleva pak vloží tlačítko Menu a jeho text, jak začít novou hru. Pod to ještě napíše nejlepší hráčův výsledek.

```
1       if hra.status == "gameover":
2           screen.blit(text_game_over, [135, 20])
3           pygame.draw.rect(screen, BARVA4, [5,60,70,20])
4           screen.blit(text_menu, [7,54])
5           screen.blit(text_controls_New, [5, 24])
6           screen.blit(text_highscore, [5, 36])
```

Zdrojový kód 10.11: Stav "gameover"

10.6.2 Game

Stav game je úvodním stavem hry. Hra zobrazí tlačítka Start a Menu.

```
1       if hra.status == "game":
2           pygame.draw.rect(screen, BARVA4, [5,30,70,20])
3           screen.blit(text_start, [11,24])
4           pygame.draw.rect(screen, BARVA4, [5,60,70,20])
5           screen.blit(text_menu, [7,54])
```

Zdrojový kód 10.12: Stav "game"

10.6.3 Menu

Po kliknutí na tlačítko Menu na úvodní obrazovce hry nastává stav menu. Hra vypíše pomocí jakých kláves ji má hráč ovládat.

```
    if hra.status == "menu":
        pygame.draw.rect(screen, BARVA4, [5,30,70,20])
        screen.blit(text_start, [11,24])
        screen.blit(text_controls_A, [5, 54])
        screen.blit(text_controls_D, [5, 64])
        screen.blit(text_controls_S, [5, 74])
        screen.blit(text_controls_Move, [5, 84])
```

Zdrojový kód 10.13: Stav "menu"

10.6.4 Menu2

Stav menu2 nastává, když po prohře hráč klikna na tlačítko Menu. Tento stav je velmi podobný stavu menu. Program ale ještě vypíše nejlepší skóre a jak začít novou hru.

```
    if hra.status == "menu2":
        screen.blit(text_controls_A, [5, 54])
        screen.blit(text_controls_D, [5, 64])
        screen.blit(text_controls_S, [5, 74])
        screen.blit(text_controls_Move, [5, 84])
        screen.blit(text_controls_New, [5, 104])
        screen.blit(text_highscore, [5, 116])
```

Zdrojový kód 10.14: Stav "menu2"

10.7 Ukončení programu

Příkaz pygame.quit() se nachází už mimo hlavní programovou smyčku. Tento příkaz ukončuje pygame. doplnit

```
    pygame.display.flip()
    hodiny.tick(fps)

pygame.quit()
```

Zdrojový kód 10.15: Ukončení programu

Závěr

Cílem teoretické části práce bylo představit historii Tetrisu. V této části práce vidím jako hlavní přínos přiblížení minulosti Tetrisu českému publiku, neboť jsem během hledání zdrojů pro tuto část na žádný český důvěryhodnější zdroj než Wikipedii nenarazila.

Cílem praktické části bylo vytvořit funkční klon Tetrisu v programovacím jazyce Python. dopsat

Samozřejmě by se dal program ještě vylepšit. Program by kromě momentálního tetrimina mohl generovat ještě například čtyři následující dílky, které by ukázal hráči. Hráč by díky tomu mohl efektivněji umisťovat dílky a možná by tím dosáhl i lepšího výsledku.

V klasické verzi Tetrisu je také hold. Hold je místo, kam si hráč může "schovat" tetrimino, dokud se nenaskytne vhodná chvíle ho použít. Opět by se tím zlepšila pravděpodobnost hráče dosáhnout vyššího skóre.

Obě tato zlepšení jsou ale už poměrně složitá pro implementování do programu. Zároveň si myslím, že hra je dostatečně zajímavá a zábavná na to, aby se bez těchto úprav obešla.

Bibliografie

1. ACKERMAN, Dan. *The Tetris Effect: The Game that Hypnotized the World*. 1. vyd. PublicAffairs, 2016. ISBN 9781610396110.
2. BROWN, Brian "Box". *Tetris: the games people play*. 1. vyd. First Second, 2016. ISBN 9781626723153.

Seznam obrázků

Seznam tabulek

Přílohy

A Vlastní program

```
import pygame
import random


barvy = [
    (0, 0, 0),
    (251, 248, 204),
    (255, 207, 210),
    (241, 192, 232),
    (207, 186, 240),
    (163, 196, 243),
    (180, 243, 248),
    (185, 251, 192)
]

class Tetrimino(object):
    tvary = [
        [[1,5,9,13],[4,5,6,7]],
        [[4,5,9,10],[2,6,5,9]],
        [[6,7,9,10],[1,5,6,10]],
        [[1,2,5,9],[0,4,5,6],[1,5,8,9],[4,5,6,10]],
        [[1,2,6,10],[5,6,7,9],[2,6,10,11],[3,5,6,7]],
        [[1,4,5,6],[1,4,5,9],[4,5,6,9],[1,5,6,9]],
        [[1,2,5,6]]
    ]
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.typ = random.randint(0, len(self.tvary) - 1)
        self.barva = self.typ + 1
        self.r = 0

    def okoli(self):
        return self.tvary[self.typ][self.r]

    def otoc_R(self):
```

```
37          self.r = (self.r + 1) % len(self.tvary[self.typ])
38      def otoc_L(self):
39          self.r = (self.r - 1) % len(self.tvary[self.typ])
40
41
42  class Tetris(object):
43      highscore = 0
44      x = 100
45      y = 60
46      zoom = 20
47      tetrimino = None
48
49      def __init__(self, height, width):
50          self.height = height
51          self.width = width
52          self.pole = []
53          self.score = 0
54          self.status = "game"
55          for i in range(height):
56              new_line = []
57              for j in range(width):
58                  new_line.append(0)
59              self.pole.append(new_line)
60
61      def new_tetrimino(self):
62          self.tetrimino = Tetrimino(3, 0)
63
64      def prekryti(self):
65          kryt = False
66          for i in range(4):
67              for j in range(4):
68                  if i * 4 + j in self.tetrimino.okoli():
69                      if i + self.tetrimino.y > self.height - 1
                           or j + self.tetrimino.x > self.width - 1
                            or j + self.tetrimino.x < 0 or \
70                           self.pole[i + self.tetrimino.y][j +
                                self.tetrimino.x] > 0:
71                              kryt = True
72          return kryt
73
74      def plna_line(self):
75          lines = 0
76          for i in range(1, self.height):
77              nuly = 0
78              for j in range(self.width):
79                  if self.pole[i][j] == 0:
80                      nuly += 1
81              if nuly == 0:
82                  lines += 1
```

```
                for k in range(i, 1, -1):
                    for l in range(self.width):
                        self.pole[k][l] = self.pole[k - 1][l]
        self.score += lines
        if self.score > self.highscore:
            self.highscore = self.score

    def slam(self):
        while not self.prekryti():
            self.tetrimino.y += 1
        self.tetrimino.y -= 1
        self.stop()

    def go_down(self):
        self.tetrimino.y += 1
        if self.prekryti():
            self.tetrimino.y -= 1
            self.stop()

    def stop(self):
        for i in range(4):
            for j in range(4):
                if i * 4 + j in self.tetrimino.okoli():
                    self.pole[i + self.tetrimino.y][j + self.
                       tetrimino.x] = self.tetrimino.barva
        self.plna_line()
        self.new_tetrimino()
        if self.prekryti():
            self.status = "gameover"

    def go_side_R(self):
        x = self.tetrimino.x
        self.tetrimino.x += 1
        if self.prekryti():
            self.tetrimino.x = x
    def go_side_L(self):
        x = self.tetrimino.x
        self.tetrimino.x -= 1
        if self.prekryti():
            self.tetrimino.x = x

    def otoc_R(self):
        old_r = self.tetrimino.r
        self.tetrimino.otoc_R()
        if self.prekryti():
            self.tetrimino.r = old_r
    def otoc_L(self):
        old_r = self.tetrimino.r
        self.tetrimino.otoc_L()
```

```
131             if self.prekryti():
132                 self.tetrimino.r = old_r
133 
134 
135 pygame.init()
136 
137 BARVA1 = (50, 50, 50)
138 BARVA2 = (0, 0, 0)
139 BARVA3 = (161, 121, 226)
140 BARVA4 = (207, 186, 240)
141 BARVA5 = (255, 173, 179)
142 size = (400, 500)
143 screen = pygame.display.set_mode(size)
144 pygame.display.set_caption("Tetris")
145 
146 konec = False
147 hodiny = pygame.time.Clock()
148 fps = 25
149 hra = Tetris(20, 10)
150 pocitadlo = 0
151 zmacknuti = False
152 
153 
154 while not konec:
155 
156     if (hra.tetrimino is None) and (hra.status == "start"):
157         hra.new_tetrimino()
158 
159     pocitadlo += 1
160     if (pocitadlo % fps == 0 or zmacknuti) and hra.status == "
        start":
161         hra.go_down()
162 
163     for event in pygame.event.get():
164         if event.type == pygame.QUIT:
165             konec = True
166         if event.type == pygame.KEYDOWN:
167             if hra.status != "gameover":
168                 if event.key == pygame.K_d:
169                     hra.otoc_R()
170                 if event.key == pygame.K_a:
171                     hra.otoc_L()
172                 if event.key == pygame.K_DOWN:
173                     zmacknuti = True
174                 if event.key == pygame.K_LEFT:
175                     hra.go_side_L()
176                 if event.key == pygame.K_RIGHT:
177                     hra.go_side_R()
178                 if event.key == pygame.K_s:
```

```
                    hra.slam()
            if hra.status == "gameover" or hra.status == "menu2
               ":
                if event.key == pygame.K_ESCAPE:
                    hra.__init__(20, 10)
                    hra.status = "start"
        if event.type == pygame.KEYUP:
                if event.key == pygame.K_DOWN:
                    zmacknuti = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            mys = pygame.mouse.get_pos()
            if hra.status == "menu" or hra.status == "game":
                if 5 <= mys[0] <= 70 and 30 <= mys[1] <= 50:
                    hra.status = "start"
            if 5 <= mys[0] <= 70 and 60 <= mys[1] <= 80:
                if hra.status == "game":
                    hra.status = "menu"
                if hra.status == "gameover":
                    hra.status = "menu2"

    screen.fill(BARVA1)
    for i in range(hra.height):
        for j in range(hra.width):
            pygame.draw.rect(screen, BARVA2, [hra.x + hra.zoom
               * j, hra.y + hra.zoom * i, hra.zoom, hra.zoom],
               1)
            if hra.pole[i][j] > 0:
                pygame.draw.rect(screen, barvy[hra.pole[i][j]],
                    [hra.x + hra.zoom * j + 1, hra.y + hra.zoom
                    * i + 1, hra.zoom - 2, hra.zoom - 2])
    if hra.tetrimino is not None:
        for i in range(4):
            for j in range(4):
                p = i * 4 + j
                if p in hra.tetrimino.okoli():
                    pygame.draw.rect(screen, barvy[hra.
                       tetrimino.barva], [hra.x + hra.zoom * (j
                        + hra.tetrimino.x) + 1,
                        hra.y + hra.zoom * (i + hra.tetrimino.y
                           ) + 1, hra.zoom - 2, hra.zoom - 2])

    font = pygame.font.SysFont('rockwell', 25, True, False)
    font1 = pygame.font.SysFont('rockwell', 12, True, False)
    text = font.render(f"Score: {str(hra.score)}", False,
       BARVA3)
    text_start = font.render("Start", False, BARVA3)
    text_menu = font.render("Menu", False, BARVA3)

    text_controls_A = font1.render("A - turn left", False,
```

```
        BARVA3)
219     text_controls_D = font1.render("D - turn right", False,
        BARVA3)
220     text_controls_S = font1.render("S - slam", False, BARVA3)
221     text_controls_Move = font1.render("use arrows to move",
        False, BARVA3)
222
223     text_controls_New = font1.render("Esc - play again", False,
         BARVA3)
224     text_game_over = font.render("Game Over", False, BARVA5)
225     text_highscore = font1.render(f"Highcore: {str(hra.
        highscore)}", False, BARVA5)
226     screen.blit(text, [5, 0])
227
228     if hra.status == "gameover":
229         screen.blit(text_game_over, [135, 20])
230         pygame.draw.rect(screen, BARVA4, [5,60,70,20])
231         screen.blit(text_menu, [7,54])
232         screen.blit(text_controls_New, [5, 24])
233         screen.blit(text_highscore, [5, 36])
234     if hra.status == "game":
235         pygame.draw.rect(screen, BARVA4, [5,30,70,20])
236         screen.blit(text_start, [11,24])
237         pygame.draw.rect(screen, BARVA4, [5,60,70,20])
238         screen.blit(text_menu, [7,54])
239     if hra.status == "menu":
240         pygame.draw.rect(screen, BARVA4, [5,30,70,20])
241         screen.blit(text_start, [11,24])
242         screen.blit(text_controls_A, [5, 54])
243         screen.blit(text_controls_D, [5, 64])
244         screen.blit(text_controls_S, [5, 74])
245         screen.blit(text_controls_Move, [5, 84])
246     if hra.status == "menu2":
247         screen.blit(text_controls_A, [5, 54])
248         screen.blit(text_controls_D, [5, 64])
249         screen.blit(text_controls_S, [5, 74])
250         screen.blit(text_controls_Move, [5, 84])
251         screen.blit(text_controls_New, [5, 104])
252         screen.blit(text_highscore, [5, 116])
253
254     pygame.display.flip()
255     hodiny.tick(fps)
256
257 pygame.quit()
```

Zdrojový kód A.1: tetris.py

B Blokové schéma