GYMNÁZIUM JÁNA ADAMA RAYMANA

TETRIS

ROČNÍKOVÁ PRÁCA

Konzultant: Mgr. Peter ČECH **Autor: Vladimír JANČÁR**

Prešov 2023

Abstrakt:

Tetris je klasická videohra, ktorú poznajú milióny ľudí po celom svete od jej vzniku v osemdesiatych rokoch. Cieľom tejto práce bolo vytvoriť hrateľnú kópiu hry pomocou programovacieho jazyka Python a knižnice Pygame v integrovanom vývojovom prostredí (IDE) Visual Studio Code. Funkčná a vizuálna stránka hry sú postavené na knižnici Pygame, zatiaľ čo logika hry je založená na manipulácii a úprave zoznamov. K vytvoreniu sme si vybrali Tetris práve preto, lebo poskytuje perfektnú príležitosť ukázať znalosti v preberanom učive informatiky na strednej škole, najmä prácu so zoznamami. Okrem toho sme do projektu zakomponovali základnú prácu s triedami a koncepty významné pre tvorbu hier ako napríklad „delta time“.

Práca je rozdelená na teoretickú a praktickú časť. Teoretická časť začína vysvetlením hry Tetris, knižnice Pygame a spôsobu, akým sme sa hru snažili naprogramovať. Následne sú v nej vysvetlené princípy a mechaniky hry, ich spracovanie a implementácia v kóde. Praktická časť detailne rozoberá štruktúru a rozdelenie kódu a riešenia konkrétnych problémov, s ktorými sme sa pri programovaní stretli. Opisuje tiež naše vlastné prídavky do hry, ako je napríklad farebná schéma herného poľa. Práca obsahuje vysvetlenie nášho spracovania videohry Tetris v programovacom jazyku Python pomocou zoznamov a hernej knižnice Pygame. Čitateľ by mal byť po prečítaní schopný využiť princípy obsiahnuté v tejto práci a použiť ich pri tvorení vlastnej verzie hry.

Kľúčové slová: Tetris, Python, Pygame, zoznamy, Visual Studio Code

Prehlásenie

Prehlasujem, že som túto prácu vypracoval samostatne a že som uviedol všetku použitú literatúru a všetky použité informačné zdroje.

Obsah

Úvod

Tetris sme v tomto projekte neprogramovali prvýkrát v živote. Dávnejšie sme vytvorili jednoduchú napodobeninu podľa Timura Bakibayeva a jeho online návodu, ktorý sme uviedli v zozname bibliografickými odkazov. V tom čase sme ešte neboli schopní vytvoriť hru bez toho, aby sme opakovali každý krok návodu. Hlavné koncepty tvorenia Tetrisu sme si zapamätali a využili ich aj v tejto už kompletne samostatnej práci.

Hra funguje na jednoduchom princípe. Herné pole je matica s rozmermi 10x20, ktorá obsahuje farebnú informáciu o každom z dvesto políčok. Následne do nej stačí vložiť padajúci tvar a zabezpečiť jeho pohyb pomocou základnej funkcionality jazyka Python. Vizuálne prvky hry, ako je vykreslenie herného poľa alebo vypisovanie skóre, sme implementovali pomocou grafickej časti knižnice Pygame. Tá v hre zabezpečuje aj vykreslenie textúr, ktoré sme vytvorili pomocou aplikácie Adobe Photoshop 2023. Medzi textúry patrí pozadie a všetok statický (nemenný) text.

1 Teoretická časť

Funkčnosťou sme sa snažili priblížiť klasickej verzii Tetrisu. Nespravili sme presnú kópiu, ako sme pôvodne plánovali. Inšpirovali sme sa rôznymi verziami hry a skombinovali sme ich s vlastnými nápadmi. Mali sme tak kreatívnu slobodu a nemuseli sme dopodrobna kopírovať mechaniku, či výzor konkrétnej verzie hry. Audio sme pri tvorení hry úplne vynechali, pretože sa nám nezdalo dôležité a nevedeli by sme ho vytvoriť sami, takže už by všetky časti projektu neboli vytvorené nami.

1.1 Tetris

Cieľom hry Tetris je umiestňovať padajúce tvary, tzv. tetromíny, do správnej pozície na hernom poli (10x20) a získať čo najvyššie skóre. To sa dá dosiahnuť otáčaním a posúvaním daných tetromínov tak, aby zapadli medzi ostatné už umiestnené bloky. Ak sa hráčovi podarí vyplniť riadok, získa skóre a riadok zmizne. Pri vyplnení viacerých riadkov naraz sú hráčovi pridelené body navyše. Cieľom hry je teda predísť naplneniu herného poľa. V prípade, že už v hernom poli nie je miesto pre nový padajúci tvar, hra končí.

1.2 Pygame

Pygame je knižnica, určená najmä k tvorbe hier. Je založená na knižnici SDL (Simple DirectMedia Layer), ktorá zjednodušuje prácu s grafikou, zvukom a vstupom na nízkej úrovni. Pygame je nadstavbou tejto knižnice a poskytuje ešte jednoduchší prístup k uvedeným funkcionalitám. Obsahuje množstvo modulov pre rôzne účely, ako je napríklad práca s časom, vykresľovanie tvarov, vypisovanie textu, prehrávanie zvuku alebo spracovanie vstupu. Výhodou Pygame je jeho kompatibilita s rôznymi operačnými systémami a to, že ide o open source projekt. Na začiatku programu sme ju importovali ako *pg,* takže sme neskôr nemuseli pred každú vstavanú funkciu Pygame písať *pygame.*

1.2.1 Spracovanie vstupu

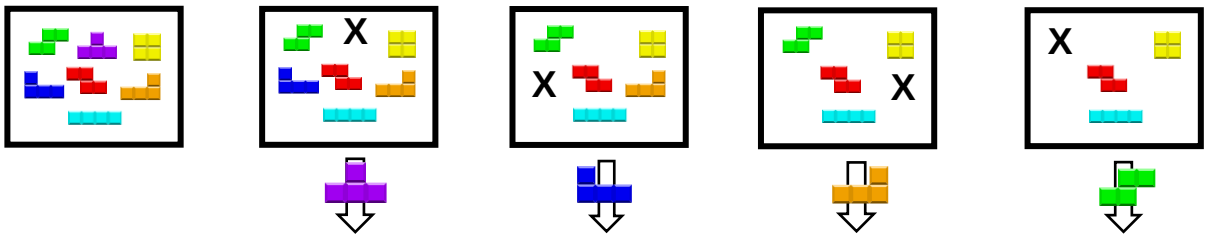
Knižnica Pygame poskytuje jednoduchý spôsob spracovania vstupu cez svoj event systém. Ten umožňuje zaznamenať rôzne druhy vstupu ako stláčanie kláves a tlačidiel myši. Po spustení Pygame v programe sa vytvorí poradovník, do ktorého sa ukladajú všetky vstupy v rozmedzí okna programu. Program môže tento poradovník prečítať pomocou funkcie *pygame.event.get()* a informácie o vstupe sa dajú využiť na ovládanie hry.

1.3 Program

Program sme sa snažili písať takým štýlom, aby nebol závislý od jedného „enginu“, v tomto prípade audiovizuálnej knižnice Pygame. To znamená, že sme používali funkcie z Pygame len pre vykresľovanie herného poľa, vypísanie textu a spracovanie vstupu. Nepoužívali sme napríklad vstavané funkcie pre kolízie, či „sprite“ triedy. Namiesto toho sme celé herné pole vytvorili pomocou zoznamov. Podstatou toho je, že hlavná časť hry, samotné herné pole a práca s ním, nie je závislá od použitej hernej knižnice. Bolo by teda veľmi jednoduché prepísať program, aby využíval inú knižnicu, keďže takáto knižnica slúži len na jediný účel – na vykreslenie matice na obrazovku.

1.4 Pseudonáhodný generátor tvarov

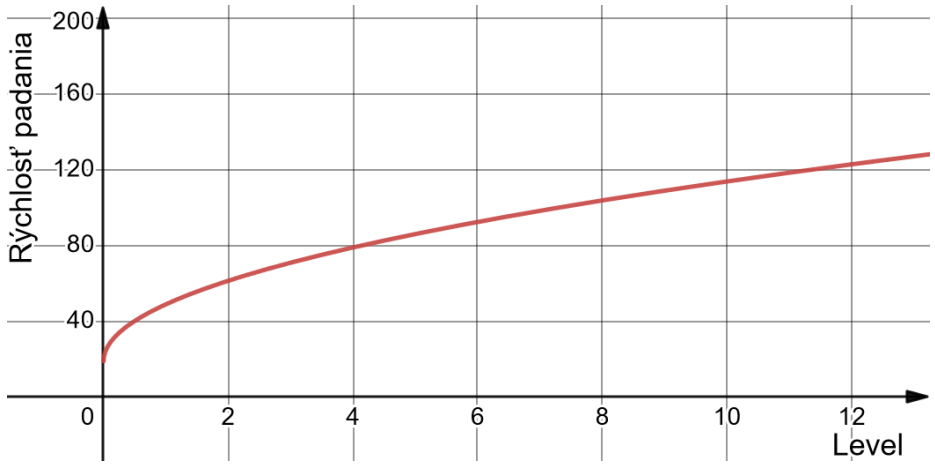
Padajúce tvary nie sú generované úplne náhodne. Aby sa nestalo, že padne rovnaký tvar niekoľkokrát za sebou a iný nepadne vôbec, využíva pôvodná hra Tetris systém pseudonáhodného generovania tetromínov. Tento systém funguje tak, že každých sedem generovaní (existuje sedem tvarov) sa vytvorí „balíček“, ktorý obsahuje každý tvar. Následne sa z neho náhodne vyberie jeden. Vytiahnutý tvar už v danom balíčku nie je, takže nasledujúci bude určite iný než predošlý. Takto to pokračuje, až kým sa balíček nevyprázdni a nevytvorí sa nový (obr. 1). Vďaka tomuto systému sa objaví každý tetromín minimálne raz za sedem generovaní, čím sa zlepšuje hrateľnosť.



obr. 1: pseudonáhodné generovanie tvarov

1.5 Progresivita hry

S každým levelom sa zvyšuje rýchlosť padania. Zrýchľovanie znázornené grafom (obr. 2) má podobu logaritmickej funkcie, teda vo vyšších leveloch sa náročnosť mení menej zjavne. Keby sa rýchlosť padania zvyšovala aritmeticky, hra by bola po určitej dobe nehrateľná. Level sa zvýši každých päť vyplnených riadkov a vzťah pre určenie rýchlosti je $\mathbf{30}\sqrt{\boldsymbol{x}}\mathbf{+19}$, kde *x* je level.



obr. 2: graf znázorňujúci závislosť rýchlosti padania od levelu

1.6 Ovládanie

Hra sa ovláda rovnako ako klasický Tetris. Šípkami vpravo a vľavo hráč hýbe padajúcim tvarom do strán, hornou šípkou ho otáča a spodnou zrýchľuje pád, medzerníkom spôsobí okamžité spadnutie. Stlačením *„ESC“* sa hra zastaví a na kraji sa objaví legenda ovládania. V tejto časti sme sa odtrhli od klasickej verzie hry a pridali sme vlastné nastavenia. Hráč môže stláčaním *„X“* a *„C“* meniť farebné schémy herného poľa a stlačením *„H“* zapnúť alebo vypnúť pomoc – obrys tvaru ukazujúci, kde tvar spadne. Tetromín sa nedá posúvať do strán držaním šípok z toho dôvodu, že je tak príliš jednoduché netrafiť sa.

2 Praktická časť

Celý kód sme napísali v jazyku Python (verzia 3.10.4) v prostredí Visual Studio Code. Hlavný program *Tetris.py* sme prerobili trikrát a vždy sme ho zorganizovali lepšie ako predtým. Pôvodne sme neplánovali používať objektovo orientované programovanie, no v tretej verzii programu sme zakomponovali štyri triedy. Nechceli sme to však komplikovať, a preto sme využívali iba základnú štruktúru tried pre uchovávanie premenných a funkcií. Využitie tried výrazne zlepšilo prehľadnosť kódu a zjednodušilo jeho úpravu. Vo finálnej verzii sme už vďaka tomu nepociťovali potrebu rozdeľovať kód do viacerých súborov.

Priečinok *Tetris* obsahuje hlavný program *Tetris.py* a ďalšie priečinky, ako je napríklad priečinok *versions*, v ktorom sa nachádzajú staršie verzie hry. Predchádzajúce verzie boli kvôli prehľadnosti rozdelené do viacerých súborov, pretože sme v nich nepoužívali triedy. Vedľajšie súbory týchto verzií sa nachádzajú v priečinku *modules*.

2.1 Kód

Po definovaní funkcií, tried a premenných sa zavolá funkcia *main()*. Táto funkcia spustí nekonečný cyklus, v ktorom sa volajú ostatné funkcie potrebné pre chod hry. Najprv funkcia *event_handler()* spracuje vstup od hráča. Následne sa aktualizuje poloha pozadia pomocou *update_BG()*. Ďalej sa na základe hodnoty premennej *Tetris().game_state* zavolá jedna z dvoch funkcií. Ak má hodnotu „playing“, aktualizuje sa herné pole funkciou *update_game()*. Ak má hodnotu „blinking“, znamená to, že hráč vyplnil riadok a blikanie tohto riadku sa zabezpečí funkciou *update_blinking()*.

Keď sa spracuje vstup a aktualizuje pozadie a herné pole, funkcia *draw()* rozhodne o tom, čo presne treba vykresliť. Hlavný cyklus má na konci obmedzenie snímok za sekundu *pygame.time.Clock().tick(FPS)*. To nedovolí cyklu zopakovať sa za sekundu viackrát, než je hodnota premennej *FPS*. Okrem toho vráti hodnotu delta time, takže sa pomocou nej na záver aktualizuje premenná *dt* (viac v sekcii 2.2.4 delta time).

2.1.1 Štruktúra kódu

Všetky komentáre a názvy funkcií a premenných sú v angličtine z dôvodu konvenčnosti a kvôli tomu, že sú väčšinou kratšie než v slovenčine. Pre prehľadnosť sme všetky funkcie otypovali. Každú z nich sprevádza informácia o tom, aký dátový typ pre vstup akceptuje a aký typ vráti. Pri úprave kódu to šetrí veľa času a na základe typu výstupu je často na prvý pohľad jasné, aký má funkcia účel.

Zdrojový kód sme rozdelili na štyri veľké odseky. Prvé dva, GAME FUNCTIONS a DRAW FUNCTIONS, obsahujú definície funkcií pre ovládanie hry a vykresľovanie. Odsek SETUP začína zavolaním funkcie *pygame.init()*, ktorá spustí časť programu viditeľnú pre hráča. Sekcia je určená k zavedeniu premenných a konštánt, z ktorých niektoré obsahujú funkcie, ktoré vyžadujú spustenie „enginu“ Pygame. Je to potrebné napríklad pri načítaní obrázku ikonky programu. Na záver sú definované triedy a po nich už nasleduje spustenie hlavnej funkcie *main()*.

2.2 Herné prvky

Pri tvorbe hry sme sa stretli s niekoľkými problémami, najmä v súvislosti s pohybom tetromínov a implementáciou teoretických konceptov do kódu. Všetky tieto problémy sa nám podarilo vyriešiť pomocou základných funkcionalít Pythonu a knižnice Pygame. Napríklad pohyb a rotáciu tetromínov sme riešili manipuláciou so zoznamom. Okrem toho sme optimalizovali hrateľnosť pre počítače s rôznym výkonom pomocou tzv. delta time. V tejto časti práce sme sa detailne venovali riešeniu týchto problémov.

2.2.1 Stav hry

Stav hry ovplyvňuje celý priebeh programu. Premenná *Tetris().game_state* uchováva jeden z piatich stavov: start, playing, paused, blinking, game_over. Pri spustení programu bude stav hry „start“, čo znamená, že program čaká na vstup od hráča. Počas hry môže hráč prepínať medzi stavmi „playing“ a „paused“. Zastavenie sme zámerne definovali ako stav, aby sa kvôli nemu nemusel prerušiť hlavný cyklus. Program tak môže neustále spracovávať vstup a všetko vykresľovať.

Blikanie (stav „blinking“) sa automaticky zapne pri vyplnení riadku a samo sa po určitom intervale prepne na „playing“. Keď sa hra skončí, nastane stav „game_over“, ktorý robí v podstate to isté čo „start“, ale vypíše „GAME OVER“ namiesto „Press SPACE to start“.

2.2.2 Herné pole

Herné pole je matica (10x20) vyplnená číslicami 0 až 8, vďaka ktorým vie funkcia *draw_game_field()* pri vykresľovaní určiť, akú má mať každý štvorec v poli farbu. Špeciálnym prípadom je číslo 0, ktoré vyjadruje prázdne políčko. To je dôležité pri manipulácii s maticou, napríklad keď program zisťuje, či má pod sebou padajúci tvar prázdne miesto – 0 a môže spadnúť. Podľa súradníc v objekte *Shape()* program vie, ktoré z čísel v poli tvoria padajúci tetromín. Každý cyklus sa z poľa odstránia, aktualizuje sa ich poloha a opäť sa doň vložia.

Podľa vstupu hráča sa s padajúcim tvarom manipuluje pomocou funkcií v triede *Shape()*. Ide napríklad o pohyb do strán alebo kontrolu, či je možné ho vykonať. Funkcie týkajúce sa celého poľa ako odstránenie plného riadku alebo kompletný reset sa nachádzajú v triede *Tetris()*.

Po dopadnutí tvaru sa vygeneruje nový. Ten je potrebné pridať na vrchol matice. V zozname *shapes* sú uložené predlohy jednotlivých tvarov a ich rotácie. Sú to krátke zoznamy obsahujúce čísla 0 až 15. Predstavujú polohu každého zo štyroch blokov, ktoré tvoria tetromín v matici 4x4 (obr. 3). Takto vznikne dvojrozmerná reprezentáciu každého tvaru a jeho rotácie. Nakoniec ho stačí premiestniť do hracieho poľa.

predloha [1, 4, 5, 6]

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

predloha [1, 2, 6, 10]

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

obr. 3: reprezentácia tvarov v matici na základe predlôh

2.2.3 Pohyb tetromínov

Pohyb a rotáciu tetromínov sme implementovali pomocou základnej úpravy zoznamu. Padajúci tvar má pridelené súradnice, vďaka čomu program pozná jeho polohu bez potreby kontroly celého herného poľa. Po stlačení klávesy pre pohyb alebo rotáciu sa zavolá príslušná funkcia z triedy *Shape()*, ktorá danú úpravu vykoná. Najprv sme vytvorili systém, ktorý pred vykonaním pohybu skontroloval jeho validitu, no nepodarilo sa nám doladiť ho tak, aby vždy fungoval bezchybne. Preto sme zaviedli nový spôsob, pri ktorom funkcia najprv vykoná zmenu polohy a pokiaľ funkcia *Shape().unplaceable()* zistí, že tetromín sa do herného poľa umiestniť nedá, zmena sa vráti späť.

2.2.4 Delta time

Každý počítač beží rôzne a my sme nepoužívali vstavané časovače, takže náš vlastný časovač by fungoval na každom počítači a pod iným zaťažením rozdielne. Presne z tohto dôvodu sa pri tvorení hier používa tzv. delta time (*dt*), teda časový rozdiel medzi poslednými dvoma snímkami. Keď vieme, aká je odozva počítača, môžeme tomu prispôsobiť pohyb objektov v hre, v tomto prípade časovanie padania.

Čím výkonnejšie je zariadenie, tým je hodnota delta time nižšia a rýchlosť opakovaní cyklov programu vyššia. Budú teda nepriamo úmerné. Keď napríklad každý cyklus meníme číselný údaj o diferenciu, vynásobíme diferenciu hodnotou delta time a zabezpečíme tak, že hodnota sa bude zvyšovať vždy rovnako rýchlo, bez ohľadu na rýchlosť opakovania cyklov. Na konci každého cyklu sa premenná *dt* aktualizuje.

2.2.5 Časovač

Knižnica Pygame má v sebe zabudovanú funkciu na vytváranie časovačov pre volanie funkcií. Tie nám však spôsobovali viac problémov ako úžitku. Bežia nezávisle od rýchlosti programu a občas sa stávalo, že program nestíhal, no časovače neboli ovplyvnené. Preto sa niekedy nejaká udalosť stala skôr, než mala. Okrem toho je potrebné tieto časovače manuálne vypínať. Snažili sme sa vytvoriť systém, kde by sa navzájom vypínali a zapínali. Teoreticky je to skvelé riešenie, ale v praxi to nefungovalo práve preto, že hlavný cyklus v programe bol „usmerňovaný“ premennou *dt* (delta time), zatiaľ čo časovače nie. Pre naše účely teda neboli vhodné, a preto sme si vytvorili vlastný systém pre prácu s časom relatívne k rýchlosti programu.

Trieda *Shape()* obsahuje premennú *Shape().falling_timer*, ktorá uchováva informáciu o tom, koľko času prešlo od posledného spadnutia. V každom cykle sa zvýši o hodnotu *dt* a ak presiahne hodnotu premennej *Tetris().falling_speed*, zavolá sa funkcia *Shape().drop(),* časovač opäť zmení svoju hodnotu na 0 a proces sa opakuje. Premenná sa nachádza v triede *Shape(),* pretože sa používa hlavne pre časovanie padania tvaru. Okrem toho sme ju použili aj pri blikaní riadkov.

2.2.6 Skórovanie

Pri spustení hry sa zo súboru *data.xml* načíta doposiaľ najvyššie zaznamenané skóre. V tejto verzii hry sme použili bodovací systém z arkádovej verzie Tetrisu, ktorá bola vydaná v roku 1988 spoločnosťou SEGA. Podľa počtu riadkov, ktoré hráč vyplnil naraz, sa pripočíta 100, 400, 900 alebo 2000 bodov. Pokiaľ ho hráč presiahne, predošlé najvyššie skóre sa prepíše.

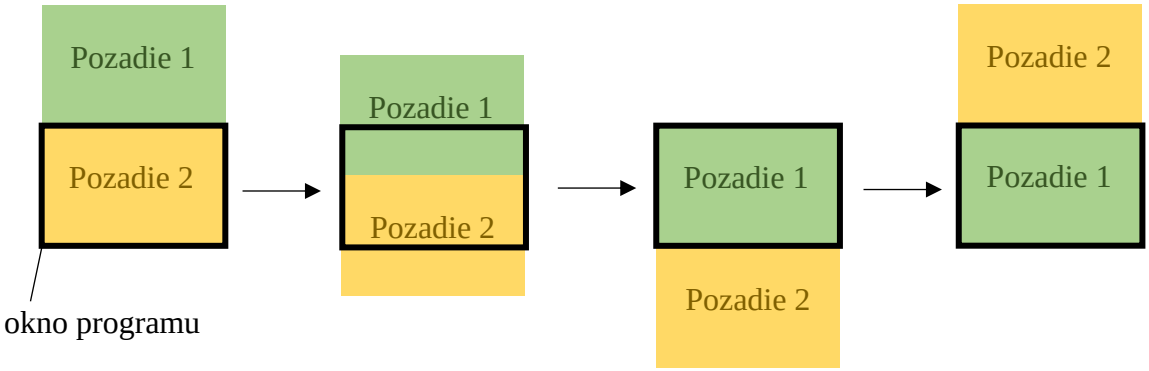
2.3 Vizuálne prvky

Za vykresľovanie je zodpovedná funkcia *draw()*, ktorá sa zavolá v každom cykle. Vždy vykreslí pozadie, herné pole s jeho ohraničením, skóre a level. O vykreslení ostatných prvkov funkcia rozhodne na základe stavu hry. Napríklad keď sa hra skončí, nevykreslí sa nasledujúci tvar ani pomocný obrys, ale zobrazí sa nápis „GAME OVER“ a hráč naďalej vidí zamrznuté herné pole. Na konci každého cyklu sa všetko ukáže na obrazovke pomocou funkcie *pygame.display.flip().*

Funkcia *draw()* pre svoje účely volá iné funkcie, ako napríklad *draw_game_field(),* ktorá vykreslí herné pole na základe čísel v ňom. Dôležitou súčasťou tohto procesu je aj funkcia *get_colour()*, ktorá vracia RGB hodnotu – množinu troch čísel vyjadrujúcu farbu štvorca. Táto funkcia je nevyhnutná hlavne kvôli farebným schémam, ktoré sa v hre dajú meniť. Podľa aktuálnej farebnej schémy a polohy daného políčka sa rozhodne, akú farbu vráti.

2.3.1 Pozadie

Pozadie tvoria dva rovnaké obrázky hviezdnej oblohy, ktoré sa pomaly hýbu dole. Majú rozmery okna a sú uložené nad sebou. V momente, keď sa spodný obrázok dostane pod spodnú hranicu okna, vrchný kompletne vyplňuje pozadie. Vtedy sa spodný obrázok presunie nad vrchnú hranicu okna a proces sa opakuje. Súradnice obrázkov sú uložené v slovníku *Background* a nikdy sa neprestávajú aktualizovať.



obr. 4: znázornenie pohybu pozadia

Pohyblivé pozadie sme pridali do hry čisto pre vizuálny efekt a preto, lebo Tetris pochádza zo zlatej éry arkádových hier, z ktorých mnohé mali futuristickú a vesmírnu tematiku. Spôsob, akým sme zaistili pohyb pozadia je inšpirovaný našou záverečnou prácou z informatiky z tretieho ročníka, kde sme sa snažili vytvoriť hru odohrávajúcu sa vo vesmírnom prostredí.

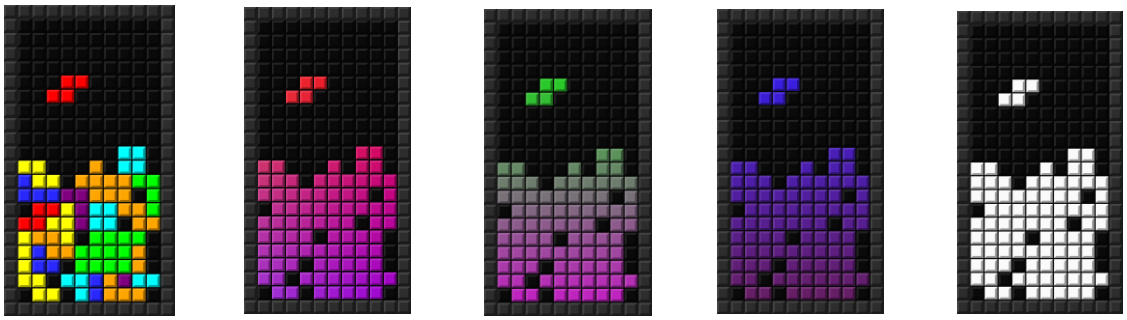
2.3.2 Text

Práca s textom v Pygame je podľa nás veľmi nepraktická. Jediné, čo Pygame umožňuje je zmena farby, fontu, veľkosti a polohy textu. Niektoré fonty knižnica nepodporuje a každý text je nutné osobitne zadefinovať. Jednoduchší a praktickejší spôsob je vykresľovať priehľadné obrázky obsahujúce text. Tento prístup umožňuje bezhraničnú úpravu písma, a preto sme ho použili pri všetkých nápisoch v hre. Na číselné údaje sme však použili vypisovanie textu pomocou Pygame, pretože ich bolo potrebné často aktualizovať, čo by bolo pri manipulácii s obrázkami priveľa práce.

2.3.3 Farebné schémy

Funkcia *get_colour()* rozhoduje akú farbu vráti podľa farebnej schémy. Ak má premenná *Tetris.colour_scheme* hodnotu 1, farba závisí od hodnoty na danej pozícii v hernom poli (napríklad číslo 1 znamená, že sa tam nachádza červený štvorec). V ostatných prípadoch funkcia vráti farbu, ktorá podmieňuje cieľový efekt schémy. Prechod farieb v hernom poli sme dosiahli tak, že jednotlivé hodnoty RGB každého štvorca sa upravujú podľa jeho pozície v hernom poli.

Napríklad pri farebnej schéme 2 sa budú hodnoty RGB padajúceho bloku postupne meniť tak, že po každom riadku bude červená hodnota o 5 menšia a modrá hodnota bude o 11 väčšia ako predtým.



Obr. 5: schémy 1 – 5

2.3.4 Textúry

Všetky textúry použité v hre sa nachádzajú v priečinku *images*. Na ich tvorbu sme použili aplikáciu Adobe Photoshop 2023 a uložili sme ich vo formáte png, aby podporovali priehľadnosť. Niektoré z nich obsahujú text, iné slúžia ako masky, ktoré pridávajú štvorcom v hernom poli efekt tieňovania. V hre sme použili dve takéto masky – jedna zakrýva celé herné pole, druhá pokrýva len jeden štvorec a je vykresľovaná nad štvorcami, ktoré tvoria nasledujúci tvar, zobrazený na pravej strane obrazovky.

Textúry nie sú nevyhnutné pre vykresľovanie herného poľa, pretože hlavná časť – farebné štvorce sú tvorené použitím Pygame. Stále sú však dôležitou súčasťou vizuálneho dizajnu hry a prispievajú k celkovému hernému zážitku.

Záver

Vytvorený program prekročil naše očakávania. Najťažšia časť projektu bola ukončiť jeho tvorenie, pretože nám napadalo ešte množstvo vecí, ktoré by sme do neho radi pridali. Hra by sa dala rozšíriť napríklad o nastavenie vlastného ovládania, menu, vizuálne efekty alebo vylepšené bodovacie mechaniky. Fungovanie hry nie je závislé od herného enginu, pretože ten má na starosti iba zaznamenávanie vstupu a vykresľovanie. Preto kód potenciálne využijeme v budúcnosti pri tvorení lepšej verzie hry, napríklad keď sa budeme učiť používať vymoženosti objektovo orientovaného programovania.

Aj napriek výslednému úspechu však nie je projekt bez chýb. Neboli sme schopní dostatočne presne napodobniť progresívne zvyšovanie náročnosti podľa klasickej verzie Tetrisu, a preto naša hra pôsobí pomalším a menej vyváženým dojmom ako originál. Vadí nám aj ovládanie pohybu tetromínov. Nie je možné ich posúvať držaním tlačidla, pretože sme nevedeli doladiť rýchlosť tak, aby nebol ich pohyb príliš rýchly a zároveň ani príliš pomalý. Problém s ovládaním, ktoré sme implementovali je, že opakované stláčanie kláves je pri vysokých rýchlostiach padania celkom náročné a nepohodlné. Zo svojich chýb sme sa však aj veľa naučili. Po troch kompletných prerábkach programu sme si uvedomili podstatu prehľadného a dobre čitateľného kódu. Projekt teda svoj účel splnil a podarilo sa nám ním predviesť naše zručnosti v programovacom jazyku Python.

Zoznam bibliografických odkazov

BAKIBAYEV, T. How to write Tetris in Python. Level Up Coding; 2020-5-18 [online]. [cit. 2023-04-23]. Dostupné z: https://levelup.gitconnected.com/writing-tetris-in-python-2a16bddb5318

PRISPIEVATELIA GITHUBU, pep-0008. GitHub, Inc; 2023-2-25. [online]. [cit. 2023-04-25]. Dostupné z: https://github.com/python/peps/blob/main/pep-0008.txt

PRISPIEVATELIA TETRISWIKI, Scoring. TetrisWiki, The Tetris Encyclopedia; 2023-3-18. [online]. [cit. 2023-04-25]. Dostupné z: https://tetris.wiki/Scoring

PRISPIEVATELIA WIKIPÉDIE, Delta timing. Wikipedia, The Free Encyclopedia; 2022-12-9. [online]. [cit. 2023-04-25]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Delta_timing&oldid=1128279575

PYGAME ORG, Pygame documentation. [online]. [cit. 2023-04-25]. Dostupné z: https://www.pygame.org/docs/

Zoznam príloh

Príloha A CD s ročníkovou prácou