Gymnázium, Praha 6, Arabská 14

Programování

RUSH HOURS

Filip Hruška, 2.E

Duben 2022

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

Anotace

Cílem této práce bylo naprogramovat hru rush hour, která je logickou hrou pro jednoho hráče a jako nadstavbu napsat solver. V dokumentaci této práce se dozvíte charakteristiku hry, interpretaci do kódu a popis hlavních metod programu a řešené problémy.

Obsah

1 Úvod

1.1 Zadání práce

Mým cílem je naprogramovat hru, ve které je za úkol uvolnit cestu hlavnímu obdélníku pomocí posouváním jiných obdélníků a její solver.

Hra se dá představit jako hrací pole s mnoha obdélníky, ve které se hlavní blok zablokován jinými, a proto je zapotřebí pohybovat bloky před blokem hlavním, aby se hlavní blok dokázal dostat na konec.

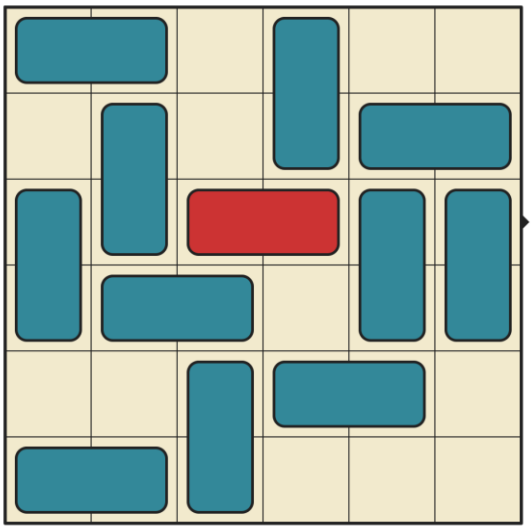
1.2 Charakteristika hry

Hra rush hours je v původní podobě desková hra, ve které se hráč snaží vyjet z hrací desky hlavním vozidlem (viz obrázek 1), v případě mé hry jde o obdélník. Jde o hru logickou pro jednoho hráče.

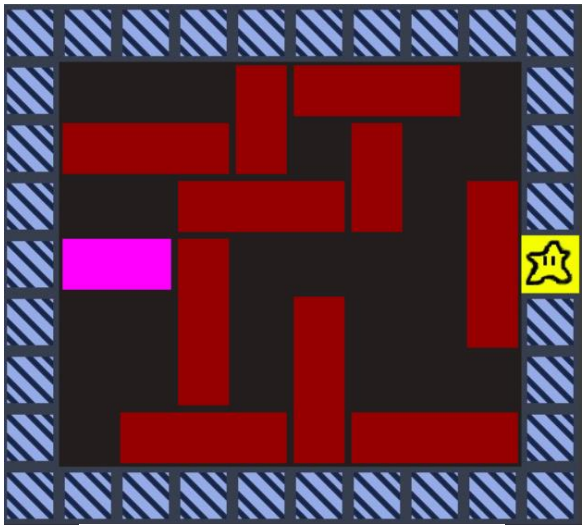
Na obrázku níže (obrázek 1) jde vidět, jak daná hra vypadá s použitím obdélníku místo původních vozidel. Červený obdélník je obdélník, který se snažíme uvolnit za pomocí pohybování jiných obdélníků. Cíl, do kterého se snažíte dostat červeným obdélníkem se v tomto případě nachází na třetím řádku na konci a je označen nenápadnou šipkou.

Moje rozhraní jde vidět na obrázku 2, na kterém jde vidět řešení vizuální stránky a způsobu vykreslení obdélníků. Obrázek na konci desky, do které se chceme dostat je zároveň tlačítko pro návrat zpátky do hlavního menu nebo na výběr hraných map. Pro hru pro jednoho hráče jsem vytvořil pět map.

V originální hře je rozhraní desky 6 x 6 bloků, ale v mé interpretaci hry jsem vytvořil souměrné pole 7 x 8.



Obrázek 1: ukázka hrací desky



Obrázek 2: ukázka hrací plochy v programu

2 Použité technologie

Program jsem napsal v programovacím jazyce python, konkrétně v pythonu 3.9 s použitím editoru zdrojového kódu Visual studio code, které je vlastněno Microsoftem a má mnoho možností rozšíření. Pro vykreslování veškerých obdélníků a okna jako takového jsem využil knihovnu pygame, která umožňuje jednoduše vykreslit objekty a obsahuje mnoho užitečných metod pomáhajících při psaní kódu.

Obrázek 3: logo pythonu

Obrázek 4: logo knihovny pygame

3 Řešené problémy

3.1 Volba jazyka

Hned při plánování a vymýšlení tématu ročníkové práce jsem se setkal s problémem, jaký jazyk použít. Pro řešení tohoto problému jsem nemusel váhat dlouho, abych dospěl k závěru, že python knihovna pygame je několikrát jednodušší a praktičtější než java FXML.

3.2 Velikost hrací desky

Velikost hrací desky je velice zásadní pro strategii ve hře. Moje rozhodnutí změnit původní rozhraní bylo kvůli nesymetričnosti. Kvůli změně rozhraní se ale změnila složitost výpočtu řešení. Další důvod změny rozhraní byla snaha o odlišení se od původní hry, jelikož poté hra funguje jinak ve smyslu postupu řešení.

Rozdíl velikostí hracích desek lze vidět na obrázcích 1 a 2 viz kapitola charakteristika hry.

3.3 Datové struktury solveru

Při psaní nějakého solveru/bota nebo dalších nadstaveb k programům je zřejmé, že je potřeba nejprve naprogramovat základ. Přesně takovýmto způsobem jsem postupoval, ale dospěl jsem do bodu, kdy jsem zjistil, že je hra napsaná nešikovně pro realizaci napsání solveru, a proto jsem, kvůli nedostatku času, začal kombinovat různé metody, pole nebo funkce, abych mohl v programování dále pokračovat.

Dalším problémem spojeným s datovými strukturami byl problém vytvoření objektů ze stringu, ve kterém je po řádcích zapsaná hrací deska. Tento problém řeší jedna metoda, která čte string po jednotlivých řádcích a vytváří objekty typu Obdélník a zapisuje je do pole.

4 Popis hlavních metod

4.1 Metoda draw

Jde o metodu, ve které se nachází veškeré vykreslování. Tato metoda je rozdělená do více částí.

4.1.1 První část

V první části metody jde o vykreslování bloků, se kterými se pohybuje v průběhu hry. Tato část metody pracuje s parametry zvoleného obdélníku, jelikož s jiným se v dané chvíli nemůže pohybovat, a vykreslí jej. Zároveň se zaobírá přidělováním barev pro objekty. Barvy se mění při zvolení objektu kliknutím a hlavní obdélník má barvy odlišné, aby byl rozeznatelný od ostatních.

4.1.2 Druhá část

V druhé části se metoda zabývá vykreslování uživatelského rozhraní. Jde o různá menu nebo o obrazovku, která se zobrazí po dohrání. Na těchto obrazovkách se objevují tlačítka, která mění proměnou, podle které se metoda orientuje. Konkrétně jde o obrazovky jménem: game, bot, end, main, levels, game a bot jsou řešeny v první části. V případě, že je proměnná nastavená na end, tak program vypíše game over a po kliknutí se proměnná na vypisování obrazovek přepíše na main.

V případě, že je proměnná nastavená na main, tak jsou na obrazovku vykresleny dvě tlačítka pro volení herního módu, buď hra pro jednoho nebo solver.

V posledním případě se na obrazovku vypíše pět obdélníků reprezentující jednotlivé levely (mapy).

Pro pohyb mezi obrazovkami lze použít malý žlutý obdélník přesně na pozici cílového bodu mapy. Po kliknutí se obrazovky přepínají z game/bot na levels, z levels na main.

4.2 Metoda select

Metoda select obstarává veškerá tlačítka, na které lze v různých obrazovkách klikat. Jelikož se v knihovně pygame nedají jednoduše definovat tlačítka jako například v javaFXML, tak je definuji jako objekt typu obdélník a kontroluji, zdali se při kliknutí pozice myši nachází nad daným obdélníkem. V této metodě se objevují veškeré podmínky na kontrolování tohoto případu.

Zároveň tato metoda zajišťuje přepínání mezi obdélníky, aby byly zvolené. Konkrétně je to vyřešeno přes smyčku for, která prochází všechny objekty v poli a kontroluje, jestli se pozice myši pří kliknutí nenachází nad jedním z obdélníků.

4.3 Metoda move_cars

V metodě move_cars se odehrává veškerý pohyb obdélníků při hraní módu pro jednoho hráče. Pro pohyb jednotlivých obdélníků je zapotřebí zvolit obdélník, kterým chci pohnout a poté pomocí šipek hýbat obdélníkem. Každý obdélník má zvolený směr pohybu podle jeho orientace (na výšku/šířku).

Metoda funguje jen v případě, že je proměnná na změnu obrazovky nastavená na game. Metoda začne fungovat po stlačení klávesy a poté se podmínky ptají, zdali se byla stlačena jedna z šipek, aby mohla dále posuzovat směr pohybu.

Aby se mohl obdélník pohnout, musí mít před sebou volno, což se kontroluje přes dvě podmínky. V první se podmínka ptá, zdali je před daným obdélníkem zeď a druhá kontroluje kolize s jinými obdélníky a pokud podmínka vrátí true, tak se obdélník posune na původní pozici.

4.4 Metoda readMap_createObj

Tato metoda zajišťuje vytvoření objektů ze stringu (textu), ve kterým je zapsaná mapa. Metoda prochází daný string po řádcích a ptá se, jestli se nějaký znak nerovná něčemu jinému než: znak #, F, nebo mezeře, jelikož tyto znaky reprezentují něco jiného (# stěna, F finální bod mapy), tak se naskytují dvě možnosti: obdélník je orientován na šířku nebo na výšku. Pokud je orientován na šířku, tak se stejný znak nalezený předtím musí rovnat znaku vpravo od něho a pokud toto neplatí, tak je stejný znak na stejné pozici akorát o řádek níže. Při procházení si program pamatuje délku hledaného obdélníku. Po nalezení všech znaků se vytvoří objekt typu Obdélník.

5 Solver

Jde o metodu, která dokáže vyřešit problém zadaný v argumentu metody. Pokud metoda daný problém nedokáže vyřešit, tak je možné, že daná mapa nemá řešení, nebo se rekurze metody zanořila až příliš daleko a v tomto důsledku přestala fungovat. Argument, který metoda dostane má stejné parametry, jako mapy v módu pro jednoho hráče.

Jak už bylo zmíněno, metoda pracuje na principu rekurze, což znamená, že metoda volá sama sebe, dokud se problém nevyřeší nebo dokud se nevyplýtvá operační paměť zařízení vyhrazená pro běh programu. V tomto programu je rekurze napsaná tak, že pokud se nějaký obdélník dokáže pohnout, tak se pohne a daná kombinace pole se zapíše do pole. Pokud se daná kombinace v poli už vyskytuje, což znamená, že v předchozích krocích byly obdélníky ve stejném stavu, tak se pohne jiný obdélník, aby mohla rekurze pokračovat.

Metoda pro řešení různých kombinací mapy používá proměnou tracker, kterou přepisuje po každém tahu a následně jí vepisuje do argumentu stejné metody (rekurze). Proměnná tracker se vytváří společně s proměnnou mapa v metodě readMap_createObj.

6 Postup instalace

Pro spuštění hry je zapotřebí si stáhnou soubor zipak, který naleznete v mém repozitáři: https://github.com/gyarab/2021-2e-hruska-rush_hour a poté spustit soubor main.exe.

7 Závěr

Se svojí prací jsem spokojen i přes nějaké nedostatky. Zadání práce jsem splnil. Solver je napsán formou rekurze, což je velmi pomalý a nekonzistentní způsob. Další rozvoj programu bych směřoval do optimalizace solveru, aby jen nezkoušel všechny možné varianty, ale vyhodnocoval nejlepší možný tah a ten následně provedl. Dalším vylepšením by mohlo být vylepšení grafického vzhledu aplikace ve smyslu přidání textur na obdélníky ve hře, aby vypadaly jako vozidla, jako tomu je v originální verzi.

8 Použité zdroje

[1] 101 Computing. Rush Hour Backtracking Algorithm [online]. 2021 [cit. 2022-04-29]. Dostupné z: https://www.101computing.net/rush-hour-backtracking-algorithm/

[2] Pygame Tutorial #1 - Basic Movement and Key Presses, 2017, In: *YouTube* [online]. 7. 11. 2017 [cit. 2022-04-28]. dostupné z https://www.youtube.com/watch?v=i6xMBig- pP4&ab_channel =TechWithTim. Kanál uživatele Tech With Tim