

Gymnázium, Praha 6, Arabská 14

Ročníková práce z programování



2019/2020

4.E Otakar Kodytek

Arabská 14, 160 00, Praha 6, Vokovice

Předmět: programování

Téma: Seaport clicker

Autor: Kodytek Otakar

Třída: 4.E

Školní rok: 2019/2020

Třídní učitelka: Mgr. Jana Urzová

Prohlášení

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, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Dne

Podpis

Anotace

Tato ročníková práce se zabývá naprogramováním imaginárního hráče internetové hry Seaport, která je určena jak pro mobilní zařízení, tak pro počítače. Imaginární hráč bude na počítači ovládat uživatelskou myš, čímž bude napodobovat hru člověka, bez jeho přítomnosti. Součástí práce je také dokumentace a textový soubor s návodem k použití aplikace a registrace do této hry.

Anotation

This term work applies with programming imaginary player of game Seaport, which was created for mobile devices as well as for computer. Imaginary player will act as human player, because of control of user's computer mouse, even without human. Another part of term work is documentation and text file with instructions about my program and registration to Seaport game.

1. Obsah

1. Obsah	1
2. Úvod	2
3. Nastavení hry.....	3
3.1. Registrace do hry.....	3
3.2. Nastavení obrazovky	3
4. Software	4
4.1. Celek Bota	4
4.2. Funkce porovnávání obrazu.....	4
4.3. Funkce pixReader.....	5
4.4. Funkce screen_Check	5
4.5. Funkce pixel_Comparison.....	6
4.6. Funkce nova_vyroba.....	6
4.7. Funkce pattern_read.....	7
4.8. Program načítání předloh.....	8
4.9. Soubor s předlohami.....	8
5. Závěr	8
5.1. Problémy s vývojem.....	8
5.2. Proč nebyla použita knihovna	9
6. Zdroje	10

2.Úvod

Motivací k výběru tohoto tématu byla moje chvilková záliba ve hře Seaport v době vybírání tématu k ročníkové práci. Prvotní myšlenkou bylo spojit usnadnění postupu ve hře a zároveň splnit ročníkový projekt.

Seaport je strategická hra, ve které je cílem hráče starat se o svůj vlastní přístav. Hráč může vysílat lodě, spouštět výrobu surovin ve svém městě, suroviny využívat k plnění úkolů, nebo pomocí nich vylepšovat a nakupovat lodě a budovy.

Bot, který hraje hru místo běžného hráče, je naprogramován v jazyce java a je určen pro počítač. Bot ovládá hru pomocí uživatelské myši, tudíž je v době používání obtížné počítač využívat jinak. Program je tedy určen pouze pro noční využití, nebo za předpokladu nepoužívání myši.

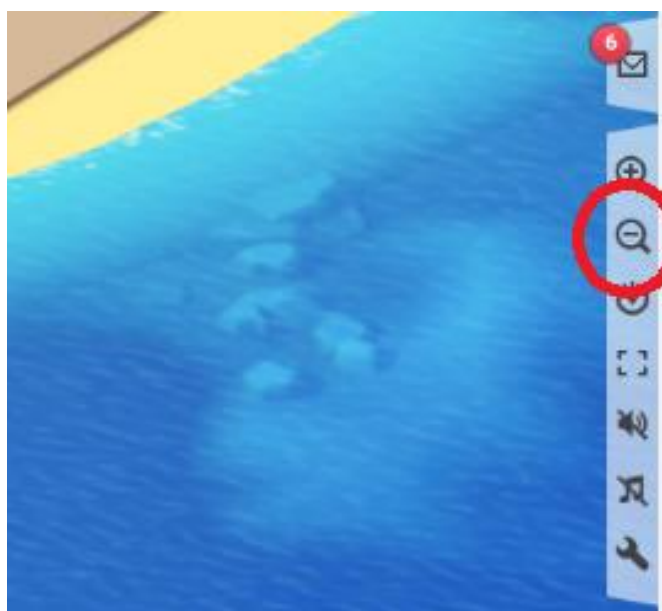
3. Nastavení hry

3.1. Registrace do hry

Hra Seaport je online hra, do které je nutné se před spuštěním registrovat. Registrace je možná na stránce <https://portal.pixelfederation.com/cs/seaport/about/>. V pravém horním rohu lze kliknout na tlačítko registrovat, případně přihlásit se, jestliže je již hráč registrovaný. Registraci lze učinit přes email, Facebook, nebo se lze zaregistrovat pomocí existujícího Google účtu. Po přihlášení již lze hru spustit.

3.2. Nastavení obrazovky

Z důvodu porovnávání obrazovky je třeba hru po spuštění maximálně oddálit na pravé straně pomocí lupy se znaménkem minus (viz obr. 2). To samé je nutné vykonat při překliknutí na mapu světa pomocí symbolu mapy v pravém spodním rohu obrazovky (vpravo dole na obr. 2). Bot v aktuální verzi (během data odevzdání 30.4.) může hru hrát pouze pokud je na počítači zvoleno rozlišení FullHD (1920 x 1080). Obrázky, které program porovnává s obrazovkou jsou totiž nahrány v tomto rozlišení při maximálním oddálení hry.



Obrázek 1

4. Software

4.1. Celek Bota

Samotný bot, jako celek je hlavní část této ročníkové práce. Tomuto programu je nutno zadat velikost obrazovky, kterou má kontrolovat. Zvolení rohů se provádí najetím myši na zvolený bod a potvrzením pomocí tlačítka ENTER. Nejprve je třeba zvolit levý horní roh, a to pod horní lištou s množstvím surovin a napravo od levé lišty s majákem, hvězdou a domem. Druhý bod, které je třeba vybrat je pravý spodní roh, který musí být položen napravo a vespod od mapy (viz obr. 2). Oba výběry hráč vybere již zmíněným tlačítkem enter. Následně se okno programu po časové prodlevě zavře a program již běží na pozadí. V každém lichém kompletním cyklu program načte zvolenou obrazovku, vyzvedne suroviny a zahájí jejich novou výrobu, vyzvedne rybářské lodě, pokusí se nalézt barely ve vodě, zkontroluje, jestli není otevřeno náhodně nějaké pole a přepne na mapu světa. V sudých cyklech bot ovládá mapu světa, vybírá lodě a pokusí se vyplout s lodí novou.



Obrázek 2

4.2. Funkce porovnávání obrazu

Bot má v externím textovém souboru nahrány RGB hodnoty čtverce 3x3 pixely. Tyto hodnoty slouží jako předloha, kterou se snaží bot na obrazovce nalézt. Bot nejprve vybere jednu předlohu a porovnává první pixel, každá z RGB barev může nabývat hodnoty od 0 do 255 bodů. Každý bod rozdílu je počítán a jestliže je celkový rozdíl u všech barev dohromady menší než určitá hodnota (aktuálně nastavena na 50), program porovná další pixely, u kterých platí stejné kritérium. Toto opatření je využito ke zvýšení rychlosti. Program porovnává takto většinou pouze 1 pixel místo 9.

Následně má každý objekt nastavenou citlivost, neboť hýbající se objekty, jako například barely mají rozptyl od předlohy řádově větší než objekty statické, jako například zlato vybírané v radnici. O načítání obrazu se v programu stará `BufferedImage` a o ovládání myši funkce `Robot`.

4.3. Funkce `pixReader`

Tato funkce dostane zadány souřadnice, na kterých následně načítá barvu pixelu. Funkce vrátí hodnotu barvy, která je vybrána v parametrech.

4.4. Funkce `screen_Check`

`Screen_Check` nejprve zaznamená aktuální stav obrazovky (podobné screenshotu) a otevře textový soubor obsahující předlohy. Program také vynuluje (změní na false) hodnoty booleanů, které zjišťují, zda byl určitý materiál již vyzvednut. Následně prochází obrazovku pomocí 2 vnořených for cyklů, které procházejí obrazovkou v osách x a y. Na každém pixelu načtou pomocí funkce `pixReader` RGB barvy čtverce 3x3 okolo pixelu a následně s těmito hodnotami zavolá funkci `pattern_read`. Na konci obou cyklů program opět uzavře textový soubor předloh.

Tato funkce je volána v nekonečném cyklu v metodě `main` a volá všechny ostatní využívané funkce.

```
int colorss[][] = new int[square_size][3];
int vysledek = 0;
for (int q = 0; q < patterns; q++) {
    drevo_nalezeno = false;
    lod_vyzvednuta = false;
    lod_vybrana = false;
    kamen_nalezeno = false;
    for (int x = x_screen_start; x < x_screen_end; ++x) {
        for (int y = y_screen_start; y < y_screen_end; ++y) {
            for (int i = 0; i < compare_size; i++) { //x axis move
                for (int j = 0; j < compare_size; j++) { //y axis move
                    int cc[] = {pixReader(x, y, i, j, 0, image), pixReader(x,
                        colorss[i * compare_size + j] = cc;
                }
            }
            pattern_read(colorss);
        }
    }
    new_pattern = true;
}
reader.close();
```

Obrázek 3 – `screen_check`

4.5. Funkce pixel_Comparison

Pixel_Comparison porovnává 2 pixely, které dostane zadány v parametrech, jeden z nich je pixel, který je načten z obrazovky, druhý je předloha. Tato funkce vrátí integer hodnotu, která udává, jak moc je pixel z obrazovky rozdílný od pixelu v předloze. Rozdíl je počítán v absolutní hodnotě pomocí funkce Math z důvodu případného sčítání v ostatních funkcích nemůže hodnota zůstat záporná.

4.6. Funkce nova_vyroba

Jak lze usoudit z názvu, tato funkce zahajuje novou výrobu v továrnách, které vyrábí určitý materiál (dřevo, kov nebo kámen). Jestliže je nějaký z těchto materiálů vyzvednut, ihned se zavolá tato funkce, která nejprve klidne na budovu, následně zvolí ikonu materiálů a pokud lze materiál vyrobit (hráč má dostatek surovin), započne výroba, v případě nedostatku materiálu funkce najde na obrazovce křížek a výrobu ukončí. V této funkci je nejvíce znát prodleva, neboť je třeba čekat, než hra načte nové okno, které může být zmapováno a prohledáno.

```
public static void nova_vyroba(int x, int y, String material) throws InterruptedException {
    clicker.mouseMove(x, y + 40);
    TimeUnit.SECONDS.sleep(1);
    clicker.mousePress(mask);
    clicker.mouseRelease(mask);
    TimeUnit.SECONDS.sleep(1);
    clicker.mouseMove(x, y + 110);
    clicker.mousePress(mask);
    clicker.mouseRelease(mask);
    if (material.contains("dřevo")) {
        int[][] col_d = {{0, 0, 0}};
        int[] patt_d = {231, 161, 59};
        for (int i = x - 40; i < x + 40; ++i) {
            for (int j = y - 40; j < y + 40; ++j) {
                int cc[] = {pixReader(x, y, i, j, 0, image), pixReader(x, y, i, j, 1, image)};
                col_d[0] = cc;
                if (pixel_comparison(col_d, patt_d, pix_number: 0) == 0) {
                    clicker.mouseMove(i, j);
                    System.out.println(i + " " + j + col_d + patt_d);
                    clicker.mousePress(mask);
                    clicker.mouseRelease(mask);
                }
            }
        }
    }
}
```

Obrázek 4

Na obrázku 4 je vidět vyhledání výroby dřeva, neboť program občas chybně vyhledá budovu radnice, místo budovy pily (na radnici je velmi podobná sekvence pixelů jako u dřeva).

```

int[][] col = {{0, 0, 0}};
int[] patt = {0, 117, 239};
TimeUnit.SECONDS.sleep( timeout: 1);
boolean vyrobit = false;
image1 = clicker.createScreenCapture(screenRectangle);
first_loop:
for (int i = x_screen_start; i < x_screen_end; ++i) {
    for (int j = y_screen_start; j < y_screen_end; ++j) {
        int cc[] = {pixReader( x: 0, y: 0, i, j, c: 0, image1), pixReader( x: 0, y:
        col[0] = cc;
        if (pixel_comparison(col, patt, pix_number: 0) == 0) {
            clicker.mouseMove( x: i + 10, y: j + 10);
            System.out.println(i + " " + j + col + patt);
            clicker.mousePress(mask);
            clicker.mouseRelease(mask);
            vyrobit = true;
            break first_loop;
        }
    }
}
}

```

Obrázek 5

Na obrázku 5 program hledá tlačítko na zahájení výroby, pokud ho nenalezne (barva bude šedá, protože není dostatek surovin), další část kódu vyhledá křížek k zavření okna.

4.7. Funkce pattern_read

Pattern_read je nejrozsáhlejší a z pohledu uživatele vykonává nejvíce práce. Pattern_read je spuštěná na každou sekvenci pixelů. Funkce nejprve načítá ze souboru předloh, data z řádku rozdělí do pole, integerů, případně zachová ve Stringu jako název. Toto pole je následně jako parametr vloženo funkci pixel_comparison. Pixel_comparison vrátí pattern_readeru rozdíl pixelů, a pokud je vrácená hodnota menší než 40, funkce zkusí porovnat další pixel. Jestliže není rozdíl příliš vysoký u žádného pixelu, funkce na základě jména předlohy ještě jednou vyhodnotí, zda je nalezený objekt opravdu správný následně vykoná všechny operace, které jsou s tímto objektem spojeny.

```

for (int j = 0; j < square_size; j++) {
    String[] strs = lines[j].trim().split( regex: "\\s+");
    for (int i = 0; i < 3; i++) {
        a[i] = Integer.parseInt(strs[i]);
    }
    int x = pixel_comparison(colors, a, j);
    if (x > 40) {
        total_difference = Integer.MAX_VALUE;
        break outerloop;
    } else {
        total_difference += x;
    }
}
}

```

Obrázek 6

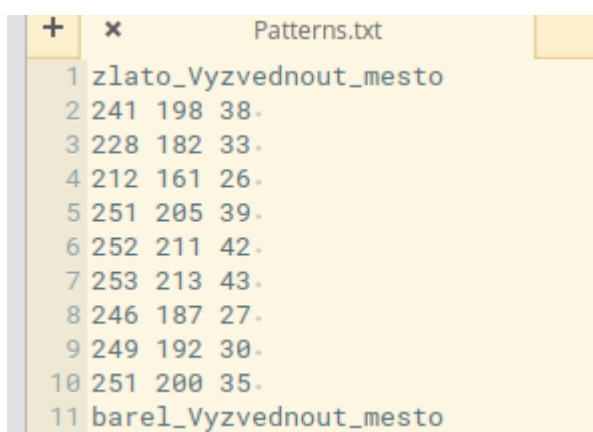
Výše je část funkce pattern_read, která za předpokladu podobné barvy všech 9 pixelů následně vyhodnocuje nalezený objekt.

4.8. Program načítání předloh

Pro načítání předloh byl vždy využit stejný program, který byl dle potřeby modifikován. Nejpoužívanější a přiložená verze otevře po spuštění okno, do kterého se vepíše název objektu, který chce uživatel uložit. Po zmáčknutí tlačítka Enter se do souboru uloží data o čtverci 3x3 pixely, konkrétně hodnoty RGB všech těchto pixelů. Tyto hodnoty bot v potaz nebere, neboť by bylo třeba změnit u bota počet předloh, které jsou v souboru uloženy. Jakákoli úprava souboru s předlohami může poškodit práci bota. Například změna posloupnosti u jednotlivých předloh by mohla zapříčinit kompletní neproveditelnost jednotlivých funkcí (funkce mapa a kotva musí být na posledním místě).

4.9. Soubor s předlohami

Předlohy jsou uloženy v běžném textovém souboru. Každá předloha je uložena na 10 po sobě jdoucích řádcích. První řádek obsahuje název předlohy, která je uložena a určena k načtení. Dalších 9 řádek obsahuje data o jednotlivých pixelech. Dalším řádkem ihned začíná další jméno předlohy.



```
+ x Patterns.txt
1 zlato_Vyzvednout_mesto
2 241 198 38.
3 228 182 33.
4 212 161 26.
5 251 205 39.
6 252 211 42.
7 253 213 43.
8 246 187 27.
9 249 192 30.
10 251 200 35.
11 barel_Vyzvednout_mesto
```

5. Závěr

5.1. Problémy s vývojem

Vývoj této aplikace byl provázen mnoha komplikacemi. Program nejprve porovnával pouze 1 pixel a hledal absolutní shodu. Tento postup se mi totiž osvědčil u sbírání barelu a rybářů. Bohužel u sbírání jednotlivých materiálů bylo toto porovnání nedostačující. Začal jsem tedy hledat knihovnu, která by mi obrazovku s předlohou porovnála. Od tohoto jsem z níže popsanych důvodů upustil a začal jsem vytvářet

vlastní program pro rozpoznávání obrazu. Během tohoto vytváření jsem zjistil dvě velmi nepříjemné zprávy.

První je, že se prostředí hry mění podle aktuálních událostí. Vývojáři hry chtějí hráče zaujmout, tudíž mění prostředí, cestovatele, za kterými lze vyslat lodě a také vzhled barelů. Bohužel jsem nepřišel na způsob, jak tento problém efektivně řešit, tudíž jsem odkázán na úpravu předloh a citlivosti porovnání až po změně programu. Vykonat tuto práci je bohužel nutné ručně, což znatelně poškozuje možnost tento program upravit pro použití jiných osob.

Druhým zásadním problémem bylo, že většina objektů není statická, ale mění většinou polohu, některé dokonce i svůj náklon (například barel), tudíž je porovnání čtverce 3x3 pixelu poměrně neefektivní, protože se stává, že barel není nalezen, pokud je obrazovka zaznamenaná v době, kdy je náklon barelu daleko od předlohy. Tento problém mě napadlo vyřešit přidáním další jedné až dvou předloh, aby byl barel nasnímán ve více různých polohách, bohužel časová náročnost programu je již takto velmi vysoká, tudíž jsem se rozhodl zachovat současný stav, neboť samotný přínos barelů není do hry příliš vysoký.

5.2. Proč nebyla použita knihovna

Nabízela se způsob použití knihovny, která by hledala shodu na obrazovce s dodanou předlohou. Tento způsob mě zprvu lákal, neboť jsem za ním viděl ušetřeno mnoho času, pravděpodobně nižší časovou náročnost programu, větší přesnost. Bohužel jsem ročníkovou práci nedělal během celého roku, abych se stihl naučit s nějakou knihovnou, ale vždy jsem pracoval pouze krátký čas s dlouhými pauzami, tudíž bylo obtížné myslet na porozumění rozpoznávacím knihovnám, které jsou poměrně složité. Druhý důvod byl, že jsem neshledával přílišný přínos tohoto způsobu mému sebezdokonalení. Tímto způsobem bych se naučil zacházet s pár funkcemi v jedné knihovně, ale obsah mé práce by dle mého úsudku nebyl příliš znatelný. Třetím důvodem bylo, že jsem začal porovnávat nejprve statické objekty, tudíž jsem si zprvu neuvědomil, jak obtížná práce bude program doladit pro pohybující se objekty.

6.Zdroje

- Obrázky 1 a 2 - hra Seaport, Pixel Federation,
<https://portal.pixelfederation.com/cs/seaport/>
- Načítání myši - <https://stackoverflow.com/questions/1439022/get-mouse-position>
- Získání barvy pixelů - <https://stackoverflow.com/questions/22391353/get-color-of-each-pixel-of-an-image-using-bufferedimages>
- Dílčí nápady a učení - <https://stackoverflow.com/>