Gymnázium, Praha 6, Arabská 14

Předmět Programování, Vyučující Jan Lána

GAME BREAKOUT

Ročníkový projekt

Tomáš Holub 1.E

2016/2017

1 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

Jméno: _____

Dne _____ v _____.

2 Úvod

2.1 anotace v českém jazyce

Hra breakout je jednoduchá hra, ve které se pohybuje micek po hracím poli odráží se od stěn, obdélníků a destičky, kterou ovládá hráč. Cílem hry je zničit všechny obdélníky, které se nachází v horní části hracího pole. po zničení všech obdélníků, hra však nekončí. Hráč je omezen na tři životy. Život ztrácí tím, že míček vypadne mimo hrací plochu ve spodní části hracího pole. Hráč sice ztrácí jeden život, ale skóre mu zůstává stejné. Hra se tedy opakuje do té doby, dokud hráč neztratí poslední život.

2.2 anotace v anglickém jazyce

Breakout game is a simple game in which the ball moves across the playing field. The ball is bouncing from the rectangle walls and the player's control plate. The goal of the game is to destroy all the rectangles that are located in the upper part of the playing field. After the destruction of all rectangles, the game does not end. The game is limited to three lives. Player loses life by the ball falling out of the playing area in the bottom of the playing field. The player loses one life but his score remains the same. The game is repeated until the player loses his last life.

2.3 Zadání

zadání mé ročníkové práce znělo přesně takto „Počítačová hra s ovládací destičkou, míčkem a několika čtverečky. Hráč ovládá destičku na spod obrazovky, kterou odráží míček, který ničí čtverečky na horní části obrazovky. Míček se odráží od stěn, destičky a čtverečků. Odráží se pod úhlem dopadu. Možnost přepnout na automatické ovládání počítačem, který odrazí kuličku pod nejlepším úhlem, aby získal nevíce bodu v určité situaci.”

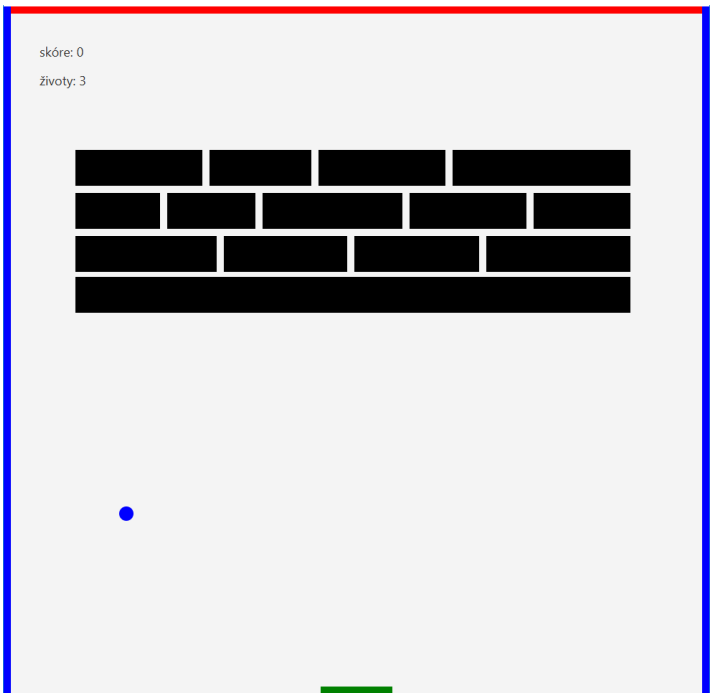
3 Samotná práce

3.1 použitá knihovna

pro vytvoření své ročníkové práce jsem si zvolil knihovnu java FX z důvodu, že jsme ji probírali na hodinách programování

3.2 Grafika

Do vytvořené scény jsem přidal tři stěny, míček, patnáct obdélníků, destičku hráče, ukazatel skóre a ukazatel životů



3.3 ovládání

pomocí metody keyevent, která rozpoznává, jaké je nyní zmačknuté tlačítko jsem vytvořil ovládání hry

při zmáčknutí mezerníku se hra aktivuje

při zmáčknutí šipky vpravo se destička posune vpravo

při zmáčknutí šipky vlevo se destička posune vlevo

při zmáčknutí tlačítka enter se destička začne posouvat sama na správnou stranu

3.4 pohyb míčku

pohyb míčku a všechny jeho odrazy se provádějí v ActionEventu Timelinu, který se opakuje každých pět milisekund. Počáteční směr míčku souřadnice X je náhodně generován při zmáčknutí mezerníku (aktivace hry). Počáteční X pozice míčku je taktéž generována náhodně při ztrátě života nebo na začátku hry. K míčku se každých pět milisekund přičte jeden pixel k X souřadnici a jeden pixel k Y souřadnici.

3.5 Odraz micku

3.5.1. Odraz od stěn

Odrazu od stěn jsem docílil tak, že jsem si vytvořil proměnou bounds, do které jsem načetl všechny čtyři stěny scény, poté jsem vytvořil podmínky kdy se má míček odrazit. Tento odraz byl ze všech ostatních odrazů nejlehčí, a tak jsem se jej snažil napodobit i u dalších odrazů, ale tam to bylo příliš zdlouhavé zmatečné, že jsem nakonec ostatní odrazy udělal jinak.

```
  //pohybkulicky
circle.setLayoutY(circle.getLayoutY() + deltaY);
circle.setLayoutX(circle.getLayoutX() + deltaX);

Bounds bounds = canvas.getBoundsInLocal();
boolean leftWall = circle.getLayoutX() <= (bounds.getMinX() + circle.getRadius() + velikostdesek);
boolean topWall = circle.getLayoutY() <= (bounds.getMinY() + circle.getRadius()) + velikostdesek;
boolean rightWall = circle.getLayoutX() >= (bounds.getMaxX() - circle.getRadius() - velikostdesek);
 if (topWall) {

     deltaY = deltaY * -1;
 }
      //odraz od prave strany nebo leve
 if (rightWall || leftWall) {
     deltaX = deltaX * -1;
 }
```

3.5.2. Odraz od destičky

Pro odraz od destičky jsem si musel vytvořit tři podmínky. První dvě podmínky ověřují, zda se míček nachází v zóně destičky a třetí podmínka ověřuje, zda má míček stejnou Y souřadnici. Třeí podmínka jednou z deseti pokusů nefungovala, Y souřadnice míčku se nerovnala s Y souřadnicí destičky, proto jsem musel vytvořit další podmínku. Pokud je rychlost micku Y souřadnice vetší, než vzdálenost od destičky přičti chybějící vzdálenost.

```
double l = wall[3].getY() - circle.getRadius() - circle.getLayoutY();
    //podminky pro odraz kulicky od desticky
boolean odraz = Math.round(circle.getLayoutY()) == (wall[3].getY() - circle.getRadius());
boolean odraz1 = circle.getLayoutX() >= (wall[3].getX() + wall[3].getLayoutX() + circle.getRadius());
boolean odraz2 = circle.getLayoutX() <= (wall[3].getX() + wall[3].getLayoutX() + wall[3].getWidth() + circle.getRadius

if (odraz1 && odraz2) {
    if (l < deltaY) {
        circle.setLayoutY(circle.getLayoutY() + l);

    }
}
```

3.5.3. Odraz od obdélníků

Tento odraz byl nejtěžší částí programu. Zkoušel jsem všechny různé možnosti a stále jsem nemohl přijít na správné řešení. Několik hodin jsem jen koukal na program a přemýšlel jaké je řešení. Už jsem to chtěl vzdát, když jsem objevil na internetu metodu intersects, ale ta mi také nefungovala. Později jsem přišel na to, že tato metoda nefunguje, protože míček není hranaté těleso. Vytvořil jsem tedy čtverec, který se neustále vypočítává na místě míčku, když se tento objekt protne s obdélníkem stačí už jen zjistit z jaké strany letí, aby věděl, kam se má míček odrazit.

3.6 Obdélníky

3.6.1. Vytvoření obdélníků

Rozhodl jsem se do hry přidat patnáct obdélníků. Každému obdélníku jsem vygeneroval náhodnou šířku, aby hra nebyla tak moc repetitivní. První obdélník se umístí sto pixelů od zdi vpravo, za tento obdélník se o deset pixelů při řádí nový do té doby, dokud další obdélník

nepřesahuje vzdálenost sto pixelů od pravé stěny, když poslední obdélník není na konci řady jeho velikost se zvětší tak, aby dokončil řadu.

```
static int minX = 100;
static int maxX = 200;
        // vytvoření obdelníků do hracího pole
public void blocks(Rectangle e[]) {
    for (int i = 0; i < block.length; i++) {

        int t = rdn.nextInt(maxX - minX) + minX;
        block[i] = new Rectangle(vzdalenostbloku + n, rada1, t, 50);
        canvas.getChildren().add(block[i]);
        double d = n + block[i].getWidth();
        double o = wall[2].getX() - vzdalenostbloku - block[i].getX();
        if (d > wall[2].getX() - vzdalenostbloku) {
            n = 0;
            rada1 = rada1 + 60;
            block[i].setX(vzdalenostbloku + n);
            block[i].setY(rada1);
        }
        if (d < wall[2].getX() - vzdalenostbloku && d + 200 > wall[2].getX() - vzdalenostbloku) {
            block[i].setWidth(o);
        }
        if (pocetbloku == 1) {
            block[i].setWidth(wall[2].getX() - vzdalenostbloku - block[i].getX());
        }

        n = t + 10 + n;
        pocetbloku--;
        jeblock[i] = true;

    }
}
```

3.6.2. Ničení obdélníků

pro hru je velmi důležité, aby se obdélníky při doteku s míčkem zničili. Já jsem tento úkol vyřešil způsobem, že jsem do podmínky, kde se míček odráží zařadil i ničení obdélníků. Vždy při doteku míčku se obdélník odebere ze scény a hráči se přidá skóre

```
//odrazzprava
if (odraz3 && odraz4 && odraz6 && jeblock[i] && kolize) {
    deltaX = deltaX * -1;
    canvas.getChildren().remove(block[i]);
    jeblock[i] = false;
    skore++;
    remainingblocks--;
}
```

příklad odrazu z pravé strany obdélníku

3.6.3. dotvoření obdélníků

Jelikož v mé hře hráč hraje tak dlouho, dokud má dostatek životů je třeba obnovovat obdélníky po tom co jsou všechny zničeny, z tohoto jsem vytvořil remainingblocks, kde je nahraný počet obdélníků, při každém odebrání obdélníku se odečte jednička, když se tato remainingblocks rovná nule celá metoda zobrazená na předešlé stránce se zopakuje.

3.7 Destička

3.7.1. Manuální pohyb destičky

Hráč má možnost manuálně ovládat destičku pomocí šipek doleva a doprava. Destička se na zmáčknutí jedné z těchto šipek posune o dvacet pět pixelů na stranu dané šipky

```
        //pohyb doleva
if (event.getCode() == LEFT && play) {
    vlevo = true;
    // wall[1] = leva stena
    double i = (wall[3].getX() + wall[3].getLayoutX() - wall[1].getX() - wall[1].getWidth());
    if (i >= rychlostdesticky) {
        wall[3].setLayoutX(wall[3].getLayoutX() - rychlostdesticky);

    } else {
        wall[3].setLayoutX(wall[3].getLayoutX() - i);
    }

} else {

    vlevo = false;
}
     //pohyb doprava
if (event.getCode() == RIGHT && play) {
    vpravo = true;
    // wall[2] = prava stena
    double i = wall[2].getX() - (wall[3].getX() + wall[3].getLayoutX() + wall[3].getWidth());
    if (i >= rychlostdesticky) {
        wall[3].setLayoutX(wall[3].getLayoutX() + rychlostdesticky);
    } else {
        wall[3].setLayoutX(wall[3].getLayoutX() + i);
    }
} else {
    vpravo = false;
}
```

3.7.2. Automaticky pohyb destičky

Tuto možnost jsem do hry přidal pouze kvůli tomu, že jsem to napsal do zadání, tato funkce ve hře se mi nelíbí, protože jde o podvod, ve kterém se hráč nemusí soustředit. Jde o to, že hráč musí podržet klávesu enter, aby se destička pohybovala sama a nepropustila ani jeden míček. Hráč tak může získat neomezené skóre.

4 Závěr

S výsledkem své práce jsem spokojen, i když jsem práci nestihl dokončit v termínu odevzdání. Nevěřil jsem, že program dokážu naprogramovat, myslím si, že na můj první větší projekt je tento program kvalitní, i když občas něco stále nefunguje, ale i tak jsem ho dokázal sám hrát kolem hodiny. Do budoucna bych v programu chtěl doladit zbylé chyby, které se v programu vyskytují.

Zdroje

https://docs.oracle.com/en/

https://stackoverflow.com/

http://stackoverflow.com/questions/20022889/how-to-make-the-ball-bounce-off-the-walls-in-javafx

https://docs.oracle.com/javase/7/docs/api/java/awt/Rectangle.html