Gymnázium, Praha 6, Arabská 14

Programování

ROČNÍKOVÁ PRÁCE

2020

Michaela Pelantová

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

ROČNÍKOVÁ PRÁCE

Předmět: Programování
Téma: Hra Breakout

Autorka: Michaela Pelantová
Třída: 4. E
Školní rok: 2019/2020
Vedoucí práce: Mgr. Jan Lána
Třídní učitel: Mgr. Jana Urzová, Ph.D.

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 Praze dne _____

_____

vlastnoruční podpis autora

ANOTACE

Tato ročníková práce se zabývá tvorbou a problematikou známé hry Breakout. Práce obsahuje popis jednotlivých metod, seznam použitých technologií a návod na ovládání hry. Text je doplněn o ilustrační obrázky a ukázky kódu.

ANNOTATION

This work deals with the creation and problems of the famous video game Breakout.
This work contains a description of individual methods, a list of used technologies and a manual. The text is supplemented with illustrations and code samples.

ZADÁNÍ PRÁCE:

Popis:
Naprogramujte hru Breakout.

Upřesnění zadání:

- hra bude mít více levelů
- bořením bricků bude mít hráč občas možnost získat nějaké bonusy

Platforma:

- Java

OBSAH

ÚVOD

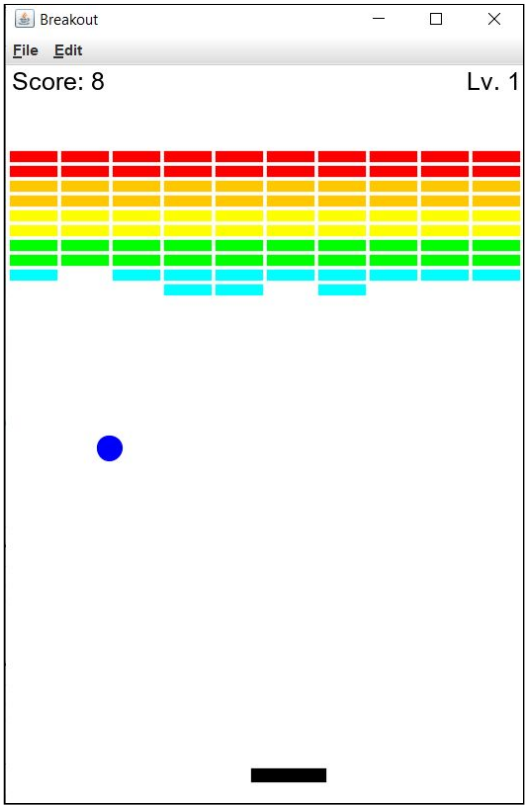
Breakout je jednoduchá, ovšem stále populární hra. Ve hře jde o to zbořit pomocí míčku a odrážecí plošinky všechny cihličky (dále už jen "cihly").
Cílem této práce bylo vytvořit hru Breakout s vlastními pravidly a s jednoduchým ovládáním, kterou by si v budoucnu mohl zahrát každý.

Vývojové prostředí

Celý program je napsán v programovacím jazyce Java, ve vývojové platformě Eclipse. Tato platforma byla zvolena pro svou jednoduchost. K vytvoření potřebných tlačítek (nakonec nepoužitých) jsem zvolila online editor *Piskel*.

Vzhled aplikace

Celá aplikace je v rámci jednoho malého okénka. Uživatel pohybuje do stran plošinkou, která je umístěna v dolní části okna. Velkou část okna zabírá cihlová zeď, která se postupně boří. Skóre lze vidět v levém horním rohu a aktuální level se nachází naproti v pravém horním rohu. Míček je na začátku uprostřed okna, ale jakmile hra začne, míček začne "létat" po celém okně a odrážet se *(viz obr. 1.1)*. Pokud míček spadne mimo plošinku, nebo se dokončí level, hra se zastaví a uprostřed okna se objeví text s instrukcemi.



Obrázek 1.1: Odražený letící míček.

1 Podrobný popis kódu

1.1 Vysvětlení jednotlivých metod

1.1.1 bricks

Celkem jsou tyto metody tři - *bricks(Color.c), bricks2()* a *bricks3()*. Všechny tyto metody vytváří "cihlovou zeď", kterou se hráč musí probourat. Jediné, čím se tyto metody liší, jsou barvy. Celkový počet cihel je nastaven na 100 kusů (deset sloupců a deset řad vytvořených pomocí *forcyklů*). Tyto metody jsou tři, aby bylo možné je na sebe v jednotlivých levelech vrstvit a bylo je možné od sebe rozeznat.

Metoda *bricks(Color.c)* tvoří zeď, která se nachází jako samotná v prvním levelu. Zároveň tvoří úplně nejspodnější vrstvu ve všech ostatních levelech. Pro zpestření mají jednotlivé dvojice řad cihel různé barvy *(viz obr. 1.2)*.

Metoda *bricks2()* tvoří zeď z šedých cihel, která tvoří druhou vrstvu ve druhém a třetím levelu.

Nakonec metoda *bricks3()* tvoří zeď z bílých cihel, která je úplně nejsvrchnější vrstvou ve třetím a posledním levelu.

```
for (double j = BRICK_Y_OFFSET; j < 10 * (BRICK_HEIGHT + BRICK_SEP) + BRICK_Y_OFFSET; j = j + BRICK_HEIGHT
        + BRICK_SEP) {

    // sets the colors of bricks
    if (j < 2 * (BRICK_HEIGHT + BRICK_SEP) + BRICK_Y_OFFSET) {
        c = Color.RED;
    } else if (j < 4 * (BRICK_HEIGHT + BRICK_SEP) + BRICK_Y_OFFSET) {
        c = Color.ORANGE;
    } else if (j < 6 * (BRICK_HEIGHT + BRICK_SEP) + BRICK_Y_OFFSET) {
        c = Color.YELLOW;
    } else if (j < 8 * (BRICK_HEIGHT + BRICK_SEP) + BRICK_Y_OFFSET) {
        c = Color.GREEN;
    } else {
        c = Color.cyan;
    }
```

Obrázek 1.2: Nastavení barev jednotlivých řádků cihel

1.1.2 paddle

Odrážecí plošinka je vytvořena pomocí metody *paddle()*. Jedná se o prostý malý obdélník umístěný v dolní části obrazovky. Pohyby plošinky do stran kopírují pohyby kurzoru uzivatele díky metodě *mouseMoved(MouseEvent e)*.

1.1.3 ball

V první řadě tato metoda vytváří míček, kterým se boří jednotlivé cihly. Je zde definována počáteční rychlost, a pomocí podmínek, i odrážení míčku od stran herního okna.

Dále je zde ošetřen případ, kdy by míček neodrazila plošinka, ale spadl by mimo ní. V tomto případě se hra zastaví, a objeví se nápis *"Game over!"* společně se skóre, kterého hráč dosáhl, a pokynem *"Click to restart the level" (viz obr. 1.3)*. Pokud si tedy hráč

přeje spustit level znovu, stačí kliknout, skóre se vyresetuje na nulu, cihly se doplní a daný level začne odznova.

Obrázek 1.3: Pozastavená hra s instrukcemi

1.1.4 score

Skóre, které může hráč vidět v levém horním rohu okna, je tvořeno ze dvou částí - z nápisu *"Score:"* a proměnné *points*. Proměnná *points* je úplně na začátku nastavena na nulu. Zvyšuje se pak pokaždé, když se hráči podaří pomocí míčku zbořit jednu cihlu. Maximální dosažitelné skóre se rovná počtu cihel. V každém levelu je tedy maximální skóre jiné.

1.1.5 checkCollision

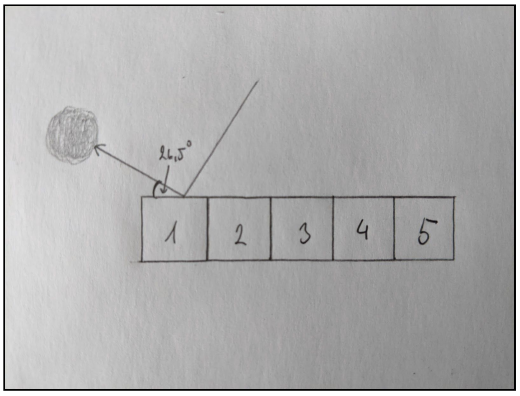
Metoda *checkCollision()* je asi tou nejsložitější metodou celého programu. Tato metoda kontroluje a ovládá, co se stane, pokud se míček dotkne něčeho jiného než jsou okraje obrazovky.

První sada podmínek zajišťuje, že pokud se míček dotkne něčeho jiného než jsou cihly, pouze změní svůj směr (odrazí se). Zároveň, pokud se odráží od čehokoliv, kromě plošinky, pak platí, že úhel odrazu se rovná úhlu dopadu.

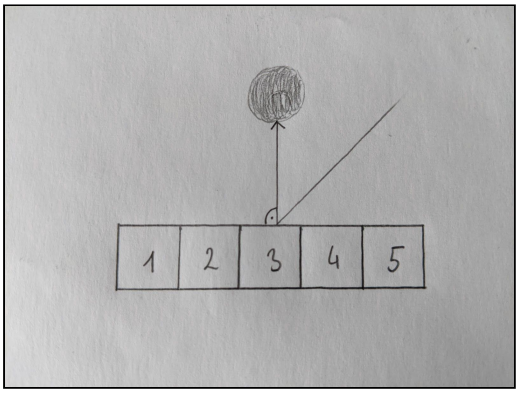
Pokud se míček odráží od plošinky, jeho úhel odrazu závisí na tom, kam na plošince dopadl. Plošinka je rozdělena na pět různých částí (její délka je rozdělena na pětiny) a podle toho se míček může odrazit pěti různými směry:

1. Když míček dopadne na první pětinu plošinky, odrazí se doleva pod úhlem zhruba 25,5 stupně. *(viz obr. 1.4)*

2. Jestliže míček dopadne na plošinku v rozmezí mezi první a třetí pětinou, odrazí se opět doleva, ovšem pod úhlem zhruba 63,5 stupně.
3. Pokud míček spadne do střední části plošinky, odrazí se pouze kolmo nahoru pod úhlem 90 stupňů od plošinky. *(viz obr. 1.5)*
4. Dále, pokud dopadne do čtvrté pětiny, odrazí se pod úhlem zhruba 63,5 stupně doprava.
5. Nakonec, když míček dopadne do poslední pětiny (na druhý konec plošinky), odrazí se pod úhlem 26,5 stupně doprava.



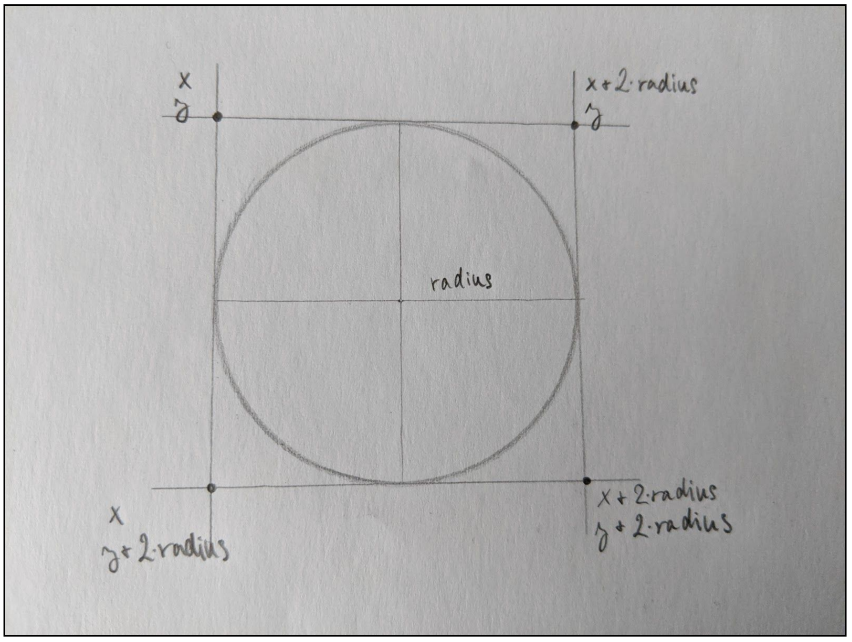
Obrázek 1.4: Znázornění odrazu míčku



Obrázek 1.5: Znázornění odrazu míčku

Úhel odrazu od plošinky a zároveň i samotná rychlost míčku závisí na dvou proměnných - *vx* pro horizontální pohyb do stran (po ose x) a *vy* pro vertikální pohyb nahoru a dolů (po ose y). Tyto proměnné jsou na začátku stejné, ovšem po dopadu na plošinku se mění - hodnota *vx* nebo *vy* se podle potřeby zvýší nebo sníží.

Další soubor podmínek rozhoduje o tom, co se stane, pokud míček narazí na cihlu. Protože míček je sice kulatý, ale jeho krajní body leží přeci jen kousek mimo něj *(viz obr. 1.6)*, bylo třeba vytvořit podmínky pro každý bod i stranu zvlášť.



Obrázek 1.6: Znázornění okrajových souřadnic míčku

Po dotyku míčku s cihlou se daná cihla odstraní (zboří se), změní se směr letu míčku (odrazí se) a zároveň se hráči přičte skóre *(viz obr. Ukázka kódu 1)*. Obrázek *Ukázka kódu 1* ukazuje pouze část kódu - jen to, co se stane, když se míček dotkne cihly pouze jedním rohem.

```
vy = vy * -1; // changes y movement when hits brick

// removes brick when hit
if (getElementAt(ball.getX(), ball.getY()) != null) {
    remove(getElementAt(ball.getX(), ball.getY()));
    points++;
    score.setLabel("Score: " + points);
} else if (getElementAt(ball.getX() + BALL_RADIUS * 2, ball.getY()) != null) {
    remove(getElementAt(ball.getX() + BALL_RADIUS * 2, ball.getY()));
    points++;
    score.setLabel("Score: " + points);
} else if (getElementAt(ball.getX(), ball.getY() + BALL_RADIUS * 2) != null) {
    remove(getElementAt(ball.getX(), ball.getY() + BALL_RADIUS * 2));
    points++;
    score.setLabel("Score: " + points);
} else if (getElementAt(ball.getX() + BALL_RADIUS * 2, ball.getY() + BALL_RADIUS * 2) != null) {
    remove(getElementAt(ball.getX() + BALL_RADIUS * 2, ball.getY() + BALL_RADIUS * 2));
    points++;
    score.setLabel("Score: " + points);
}
```

Obrázek 1.7: Část podmínek metody checkCollision()

1.2 Seznam použitých konstant

- APPLICATION_WIDTH - určuje šířku okna
- APPLICATION_HEIGHT - určuje výšku okna
- BOARD_WIDTH - určuje šířku herní plochy
- BOARD_HEIGHT - určuje výšku herní plochy
- NBRICKS_PER_ROW - určuje počet cihel v jednom řádku
- NBRICK_ROWS - určuje počet řádků cihel (jak “tlustá” bude zeď)
- BRICK_SEP - určuje velikost mezer mezi jednotlivými cihlami
- BRICK_WIDTH - určuje šířku jedné cihly
- BRICK_HEIGHT - určuje výšku jedné cihly
- BRICK_Y_OFFSET - určuje jak velké je odsazení “cihlové zdi” od horního okraje herní plochy
- PADDLE_WIDTH - určuje šířku odrážecí plošinky
- PADDLE_HEIGHT - určuje výšku odrážecí plošinky
- PADDLE_Y_OFFSET - určuje v jaké výšce se odrážecí plošinka nachází
- BALL_RADIUS - určuje poloměr míčku

2 Problémy

2.1 Menu

Původně to nebylo v zadání, ale napadlo mě, že by bylo dobré vytvořit jakési menu, kde by se hráč mohl volně pohybovat. Součástí toho menu byla možnost zvolit si konkrétní level, který si by hráč chtěl zrovna zahrát. Dále by tam byla možnost zobrazit si nápovědu (jak se hra ovládá a jak funguje, co je cílem...).

Menu samo o sobě fungovalo. Díky metodám *welcomeScreen()*, *selectLevelScreen()* a *mouseClicked(MouseEvent e)* se dalo překlikávat z jedné stránky na druhou pomocí tlačítek, které byly vyrobeny v online editoru *Piskel*. Problém však nastal, když se menu propojilo se samotnou hrou. Jakmile hráč kliknul na tlačítko, kterým by si zvolil level, hra se nespustila a na místo toho vše zmizelo.

Bohužel se tento problém nepodařilo vyřešit, takže hra žádné menu nemá.

2.2 Pozastavení hry

Většina her, pokud to zrovna není hra pro více hráčů, má možnost hru pozastavit a opět spustit (případně další funkce, jako je možnost restartování, nebo ukončení/vrácení do hlavního menu). Proto jsem se rozhodla i ve své hře tuto funkci umožnit.

Původní nápad byl takový, že hráč by mohl hru pozastavit stisknutím mezerníku na klávesnici. To umožňovala metoda *keyPressed(KeyEvent e)*. Pomocí podmínek bylo nadefinováno, že pokud se míček hýbe a zmáčkne se mezerník, míček se zastaví (výše zmíněné proměnné, v podkapitole ***checkCollision***, *vx* a *vy* se nastaví na nulu).

Pozastavení fungovalo, ovšem pak nastal problém při znovuspuštění - jak dát míček znovu do pohybu přesně v tom směru, v jakém se pohyboval před zastavením. Šlo o to někde si zapamatovat hodnoty proměnných *vx* a *vy* před tím, než se změní na nulu.

Bohužel se ani tento problém nepodařilo vyřešit ani přesto, že jsem tušila, v čem tento problém spočívá.

3 Návod na ovládání

Po spuštění hry se uživateli otevře nové okno o velikosti 420x600 pixelů. Celá hra se ovládá pouze myší. Jediné, co hráč musí udělat, aby hra začala, je kliknout kamkoliv do herního okna a míček se začne pohybovat. Hráč musí míček odrážet pomocí plošinky, která se pohybuje spolu s myší. Jakmile hráč zboří všechny všechny cihly v daném levelu, hra se zastaví a dá pokyn pro spuštění dalšího levelu - opět kliknutím. Pokud se hráči podaří zbořit všechny cihly i ve třetím levelu, hra se zastaví a na obrazovce se objeví gratulace k dokončení hry: *Congratulations for completing the game!*

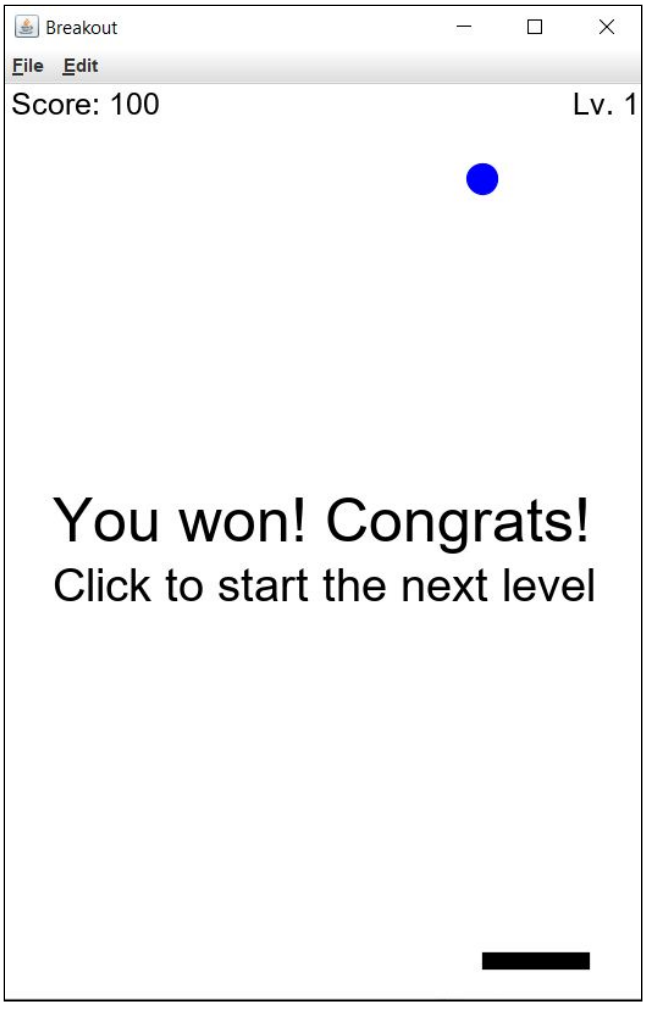
ZÁVĚR

Hra je funkční a připravena pro hráče. Přechody mezi jednotlivými levely jsou bez potíží a i hráč, který si nepřečetl, jak se tato hra ovládá, ji určitě ovládat zvládne díky pokynům v průběhu hry. Hra je (po troše úsilí a lidské trpělivosti) dohratelná.

Přes uspokojivý výsledek má hra ještě hodně prostoru pro vylepšení. Jedním z mnoha je i právě ono menu, které by celkovému vzhledu přidalo na větším dojmu propracovanosti a určitě i pohodlnějšímu užívání. Dále by se dala přidat možnost pozastavení, resetování či ukončení hry. Pak také více životů a samozřejmě i více zajímavějších levelů, kterých jsem kvůli nedostatku času nestihla vytvořit tolik.

Bohužel se mi do hry nepodařilo přidat bonusy, které by hráč mohl občas náhodně získat, protože se mi nepodařil zprovoznit "random generátor".

Přes všechno jsem s výsledkem poměrně spokojená, ovšem v budoucnu se k tomuto projektu určitě vrátím a pokusím se ho ještě vylepšit.



Obrázek 4.1: Level complete

ZDROJE A POUŽITÁ LITERATURA

MouseListener [online]. [5.3.2020]. Dostupné na: https://docs.oracle.com/javase/7/docs/api/java/awt/event/MouseListener.html

KeyListener [online]. [10.2.2020]. Dostupné na: https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyListener.html

Wikipedia [online]. [1.4.2020]. Dostupné na: https://en.wikipedia.org/wiki/Breakout_(video_game)

Požadavky na ročníkový projekt a maturitní práci z programování [online]. [29.4.2020]. Dostupné na: https://vyuka.gyarab.cz/kahoun/download/rocnikovy_projekt_pozadavky.pdf

Eclipse [online]. [23.11.2020]. Dostupné na: https://www.eclipse.org/downloads/

The ACM Java Libraries [online]. [20.4.2020]. Dostupné na: https://cs.stanford.edu/people/eroberts/jtf/javadoc/student/index.html