

**Vyšší odborná škola a Střední průmyslová škola, Šumperk,  
Gen. Krátkého 1, 787 29 Šumperk**



**Obor vzdělání: Informační technologie (18-20-M/01)**

# **MATURITNÍ PRÁCE**

**z odborných předmětů**

**Školní rok: 2016/2017  
Třída: IT4A**

**Tomáš Mencner**

**Vyšší odborná škola a Střední průmyslová škola, Šumperk,  
Gen. Krátkého 1, 787 29 Šumperk**



**Obor vzdělání: Informační technologie (18-20-M/01)**

# **ZADÁNÍ MATURITNÍ PRÁCE**

## **z odborných předmětů**

<b>Žák:</b>	<b>Tomáš Mencner</b>
<b>Téma maturitní práce:</b>	<b>Návrh a vývoj programu</b>
<b>Obsah maturitní práce:</b>	<b>1. Objektový návrh 2. Návrh a tvorba GUI 3. Vývoj programu 4. Ozvučení 5. Testování</b>
<b>Další náležitosti zadání:</b>	<b>K:/sablony/maturitní práce</b>
<b>Datum zadání:</b>	<b>30. září 2016</b>

**Datum odevzdání: 31. března 2017**

**Vedoucí práce: Mgr. Zdeněk Přikryl      Podpis:**

## **PROHLÁŠENÍ**

**Prohlašuji, že jsem maturitní práci z odborných předmětů vypracoval samostatně a k vypracování jsem použil pouze zdroje uvedené v seznamu literatury.**

**Datum: 29. března 2017**

**Žák: Tomáš Mencner**

**Podpis:**

## KONZULTAČNÍ LIST

Žák: Tomáš Mencner

Třída: IT4A

Téma maturitní práce: Návrh a vývoj programu

---

Datum kontroly: \_\_\_\_\_

Obsah kontroly: Objektový návrh

Hodnocení kontroly: 1 2 3 4 5 Podpis vedoucího práce: \_\_\_\_\_

Datum kontroly: \_\_\_\_\_

Obsah kontroly: Návrh a tvorba GUI

Hodnocení kontroly: 1 2 3 4 5 Podpis vedoucího práce: \_\_\_\_\_

Datum kontroly: \_\_\_\_\_

Obsah kontroly: \_\_\_\_\_

Hodnocení kontroly: 1 2 3 4 5 Podpis vedoucího práce: \_\_\_\_\_

Datum kontroly: \_\_\_\_\_

Obsah kontroly: \_\_\_\_\_

Hodnocení kontroly: 1 2 3 4 5 Podpis vedoucího práce: \_\_\_\_\_

Datum kontroly: \_\_\_\_\_

Obsah kontroly: Kontrola dokumentace před odevzdáním

Hodnocení kontroly: 1 2 3 4 5 Podpis vedoucího práce: \_\_\_\_\_

# Obsah

<b>1 Úvod</b>	<b>6</b>
<b>2 Objektový návrh</b>	<b>7</b>
2.1 StartingClass	7
2.2 Pacman	8
2.3 Point	9
2.4 Block	9
2.5 Class Ghost	10
<b>3 Návrh a tvorba GUI</b>	<b>17</b>
3.1 GIMP	17
3.2 Tvorba grafiky	18
3.3 Přehled vytvořených textur	19
<b>4 Programy použité při tvorbě</b>	<b>20</b>
4.1 Eclipse 4.6.2	20
4.2 Sublime Text 2	21
4.3 Git	21
4.4 Bitbucket	22
4.4.1 Verzování	22
4.4.2 Ztráta dat	23
4.5 Audacity	24
<b>5 Vývoj hry</b>	<b>25</b>
5.1 Seznam úkolů	25
5.2 Ukázka kódu	26
5.3 Ovládání hry	27
<b>6 Testování</b>	<b>28</b>
6.1 Popis a řešení problému	28
<b>7 Závěr</b>	<b>29</b>
7.1 Instrukce ke spuštění	30
<b>8 Zdroje</b>	<b>31</b>

# 1 Úvod

Maturitní práce z programování byla pro mne jednoznačnou volbou. Za 4 roky studia informačních technologií již odhalíte své silné a slabé stránky. Svůj talent jsem našel ve snadném chápání matematiky a programování. Tyto dvě dovednosti můžu velice dobře využít při tvorbě své maturitní práce.

Během zadávání maturitní práce jsem si sám stanovil, že si to ztížím výběrem jiného programovacího jazyku než zbytek třídy, a proto budu odkázán sám na sebe. Nelíbila se mi myšlenka dostávat zdrojový kód na stříbrném podnose, protože se chci zdokonalit. Stanovil jsem si pár pravidel a cílů na této práci. Maturitní práci naprogramuji v jazyce Java, použiji pouze základní javovské knihovny a naučím se s verzovacím programem. Dovednost verzování mi bude do budoucna velice užitečná a pojistím si tak ztrátu veškeré práce, usnadním si orientaci v kódu v závislosti na čase a uvidím jak postupuji nebo se mohu vrátit k určité verzi, když něco pokazím.

Maturitní práci beru jako veliký přínos a pokusím se na ní využít veškeré znalosti, které jsem během studia získal a zároveň své znalosti prohloubit za pomoci samostudia v průběhu vývoje aplikace. Jsem zastáncem myšlenky, že člověk se nejlépe učí za pochodu a proto očekávám osvojení mnoha nových dovedností.

Jako téma maturitní práce jsem si vybral okopírování hry z roku 1994, kterou vydala firma ZONER software pod názvem Pampuch pro platformu Windows. Na této hře jsem vyrůstal a občas bych si ji moc rád zahrál, abych se mohl vrátit do dětských let.

## 2 Objektový návrh

atribut - Jedná se o proměnou určitého typu, jenž udává jakou informaci proměnná může nést.

metoda - Algoritmus, který může mít vstupní parametry a může mít návratovou hodnotu.

### 2.1 StartingClass

- atributy
  - Pacman pacman
  - Image currentPacman
  - Image deadPacman, deadPacman2
  - Image point
  - Image ghost
  - Image block1, block2, block3, block4, currentBlock
  - Image gameOver
  - Image paused
  - URL base //pro získání cesty projektu
  - Animation animRight
  - Animation animLeft
  - Animation animUp
  - Animation animDown
  - boolean pacmanIsDead
  - boolean pause, gameOverPlayed
  - int startLives
  - int score
  - int bestScore
  - int level
  - int lives
  - String bestPlayer
  - ArrayList<Point> points
  - ArrayList<Block> blocks
  - ArrayList<Ghost> ghosts
  - AudioClip coin, gameOverSound
- metody
  - init //startovní nastavení projektu
  - start //spuštění hry
  - run //nekonečný cyklus hry
  - animate
  - update
  - paint
  - keyPressed //ovládání hry a pacmana
  - keyReleased
  - keyTyped
  - addPoint
  - addBlock

- addGhost
- pacmanPointCollision
- pacmanBlockCollision
- loadMap
- reloadLevel
- restartGame
- loadBestPlayer
- writeBestPlayer

## 2.2 Pacman

- atributy
  - int centerX
  - int centerY
  - int startingCenterX
  - int startingCenterY
  - int SPEED
  - int speedX
  - int speedY
  - int direction
  - int changeDirection
  - boolean wait
  - Rectangle rect
  - boolean canUp, canLeft, canRight, canDown
- metody
  - Pacman //konstruktor
  - update //updatuje pozici pacmana
  - changeDirection
  - blockCollision
  - moveRight
  - moveLeft
  - moveUp
  - moveDown
  - stop
  - getCenterX
  - getCenterY
  - getSpeedX
  - getSpeedY
  - setCenterX
  - setCenterY
  - setSpeedX
  - setSpeedY
  - getDirection
  - setDirection
  - getChangeDirection
  - setChangeDirection
  - updateRect



- setPacmanToStartingPosition
- getCanUp
- getCanDown
- getCanLeft
- getCanRight

## 2.3 Point

- atributy
  - int x, y
- metody
  - Point
  - getX
  - getY
  - setX
  - setY

## 2.4 Block

- atributy
  - int x, y
  - Rectangle rect
- metody
  - Block
  - getX
  - getY
  - setX, setY
  - getRect

## 2.5 Class Ghost

```
package pacman;

import java.awt.Rectangle;
import java.util.ArrayList;

public class Ghost {
    private int centerX;
    private int centerY;
    private int startingCenterX, startingCenterY;
    private int SPEED = 3;
    private int speedX;
    private int speedY;
    private int direction;
    private Rectangle rect = new Rectangle(0, 0, 0, 0);
```

Konstruktor Ghost, který má pouze dva parametry a to pozici x a y.

```
    public Ghost(int x, int y) {
        centerX = x;
        centerY = y;
        startingCenterX = x;
        startingCenterY = y;
        speedX = 0;
        speedY = 0;
        direction = 0;
        rect.setRect(x, y, 30, 30);
    }
```

V metodě update dochází k řízení duchovy pozice.

Proměnná centerX je nositel informace, kde se duch nachází na ose x. K této proměnné se přičítá jeho aktuální rychlost po ose x a ta může nabývat hodnot {-3;0;3}. Tyto čísla jsou vyjádřeny v pixelech. Toto platí i pro osu y.

Větev if a else if kontroluje kolizi s hranicí hrací plochy. Pro upřesnění levý horní roh má souřadnice [0;0] a pravý dolní [720;480]. Na hrací plochu jsem si zvolil políčka o velikosti 30\*30 pixelů. Na ose x se nachází 24 políček, což je ve výsledku 720 pixelů a 16 políček na ose y, což stanovuje hodnotu 480 pixelů. První větev kontroluje kolizi se spodní hranou. Hodnota hranice však nemá hodnotu 480 ale pouze 450, protože obrázky se vykreslují od levého horního rohu. Jakmile je pozice ducha mimo hrací plochu, duch je navrácen na okraj hrací plochy a zvolí si nový směr.

Metoda `updateRect()` pouze aktualizuje pozici neviditelného čtverce, ve kterém se duch nachází. Tento čtverec je použit při detekování kolizí s jinými objekty.

```
public void update(int pacmanX, int pacmanY) {
    centerX += speedX;
    centerY += speedY;

    if (centerY + speedY >= 451) {
        centerY = 450;
        chooseDirection(pacmanX, pacmanY);
    } else if (centerY + speedY <= -1) {
        centerY = 0;
        chooseDirection(pacmanX, pacmanY);
    } else if (centerX + speedX <= -1) {
        centerX = 0;
        chooseDirection(pacmanX, pacmanY);
    } else if (centerX + speedX >= 691) {
        centerX = 690;
        chooseDirection(pacmanX, pacmanY);
    }

    updateRect();
}
```

For cyklus postupně prochází všechny duchy. Podmínka `if (i != j)` zařizuje, aby se nekontrolovali duchové se stejným `id` z dynamického pole `ghosts`. Když je splněna tato podmínka vyhodnocuje se další, ve které se zjišťuje zdali kolidují dva vybraní duchové. Metoda `intersects` zjišťuje jestli čtverec jednoho ducha se nachází ve čtverci druhého ducha. Pokud ano, nastaví se souřadnice `x`, `y` dle pravidel stanovených v metodě `setXYAfterGhostCollision`, zvolí se nový směr a pokračuje se v detekci kolizí u dalšího ducha.

Poslední částí kódu je zjišťování kolizí se stěnami.

```
public void ghostCollision(ArrayList<Ghost> ghosts,
    ArrayList<Block> blocks, int j,
    int pacmanX, int pacmanY) {
    for (int i = 0; i < ghosts.size(); i++) {
        Ghost gh = (Ghost) ghosts.get(i);
        if (i != j) {
            if (rect.intersects(gh.rect)) {
                setXYAfterGhostCollision();
                chooseDirection(pacmanX, pacmanY);
            }
        }
    }
    blockCollision(blocks, pacmanX, pacmanY);
}
```

Metoda `blockCollision` postupně ověří kolizi se všemi zdmi za pomoci čtverců. Když dojde ke kolizi se zdí vybere se nový směr.

```
public void blockCollision(ArrayList<Block> blocks, int pacmanX,
                           int pacmanY) {
    for (int i = 0; i < blocks.size(); i++) {
        Block b = (Block) blocks.get(i);
        if (rect.intersects(b.getRect())) {
            chooseDirectionAfterBlockCollision(pacmanX,
                                                pacmanY);
        }
    }
}
```

Metoda `setXYAfterGhostCollision` je použita až po kolizi s duchem, jak již vyplývá z názvu.

U ducha nejprve proběhne metoda `update`, která změní jeho souřadnice buďto v ose x nebo y o 3 pixely. Jakmile nastane kolize s duchem dvě, duch jedna se navrátí o 3 pixely v příslušném směru.

Pokud `Direction` nabývá hodnoty 1, jeho pohyb je po ose y o souřadnice se odečítají. Jinými slovy pohybuje se nahoru. 2 - míří dolů. 3 - míří vlevo. 4 - míří vpravo.

A jako poslední se aktualizuje pozice čtverce.

```
public void setXYAfterGhostCollision() {
    switch (getDirection()) {
        case 1:
            centerY += 3;
            break;
        case 2:
            centerY -= 3;
            break;
        case 3:
            centerX += 3;
            break;
        case 4:
            centerX -= 3;
            break;
    }
    updateRect();
}
```

Metoda, která podle určitých pravidel rozhoduje o směru ducha. Metodu používám u kolizí.

Nejprve se duch zastaví a poté se rozhoduje kam půjde. Pokud jeho směr byl nahoru nebo dolů, tak se rozhoduje jestli zvolí směr doprava či doleva a pokud byl směr vlevo či vpravo, tak volí mezi směrem nahoru a dolů. Tyto směry si vybírá na základě pozice pacmana.

```
public void chooseDirection(int pacmanX, int pacmanY) {
    stop();

    switch (getDirection()) {
        case 1:
        case 2:
            if (pacmanX < centerX) {
                setDirection(3);
                moveLeft();
            } else {
                setDirection(4);
                moveRight();
            }
            break;
        case 3:
        case 4:
            if (pacmanY < centerY) {
                setDirection(1);
                moveUp();
            } else {
                setDirection(2);
                moveDown();
            }
            break;
    }
}
```

Po kolizi se stěnou se duch navrátí o 3 pixely v opačném směru než se pohybuje a zvolí nový směr.

```
public void chooseDirectionAfterBlockCollision(int pacmanX,
        int pacmanY) {
    switch (getDirection()) {
        case 1:
            centerY += 3;
            break;
        case 2:
            centerY -= 3;
            break;
        case 3:
            centerX += 3;
            break;
        case 4:
            centerX -= 3;
            break;
    }

    updateRect();
    chooseDirection(pacmanX, pacmanY);
}
```

Duchové se po načtení mapy rozhodují, kterým směrem se vydají po spuštění hry. Směr se určuje na základě pozice pacmana.

```
public void chooseStartingDirection(int pacmanX, int pacmanY) {
    if (pacmanY == centerY) { // up
        setDirection(1);
    } else if (pacmanX >= centerX) { // right
        setDirection(4);
    } else if (pacmanX < centerX) { // left
        setDirection(3);
    }
}

public void moveRight() {
    speedX = SPEED;
}

public void moveLeft() {
    speedX = -1 * SPEED;
}

public void moveUp() {
    speedY = -1 * SPEED;
}

public void moveDown() {
    speedY = SPEED;
}
```

```

public void stop() {
    speedX = 0;
    speedY = 0;
}

public int getCenterX() {
    return centerX;
}

public int getCenterY() {
    return centerY;
}

public int getSpeedX() {
    return speedX;
}

public int getSpeedY() {
    return speedY;
}

public void setCenterX(int centerX) {
    this.centerX = centerX;
}

public void setCenterY(int centerY) {
    this.centerY = centerY;
}

public void setSpeedX(int speedX) {
    this.speedX = speedX;
}

public void setSpeedY(int speedY) {
    this.speedY = speedY;
}

public int getDirection() {
    return direction;
}

public void setDirection(int direction) {
    this.direction = direction;
}

public void updateRect() {
    rect.setRect(centerX, centerY, 30, 30);
}

public Rectangle getRect() {
    return rect;
}

```

Nastaví ducha na startovní pozici, zastaví ho a nastaví hodnotu směru jako 0, která označuje začátek hry.

```
public void setGhostToStartingPosition() {  
    centerX = startingCenterX;  
    centerY = startingCenterY;  
    stop();  
    setDirection(0);  
}  
}
```

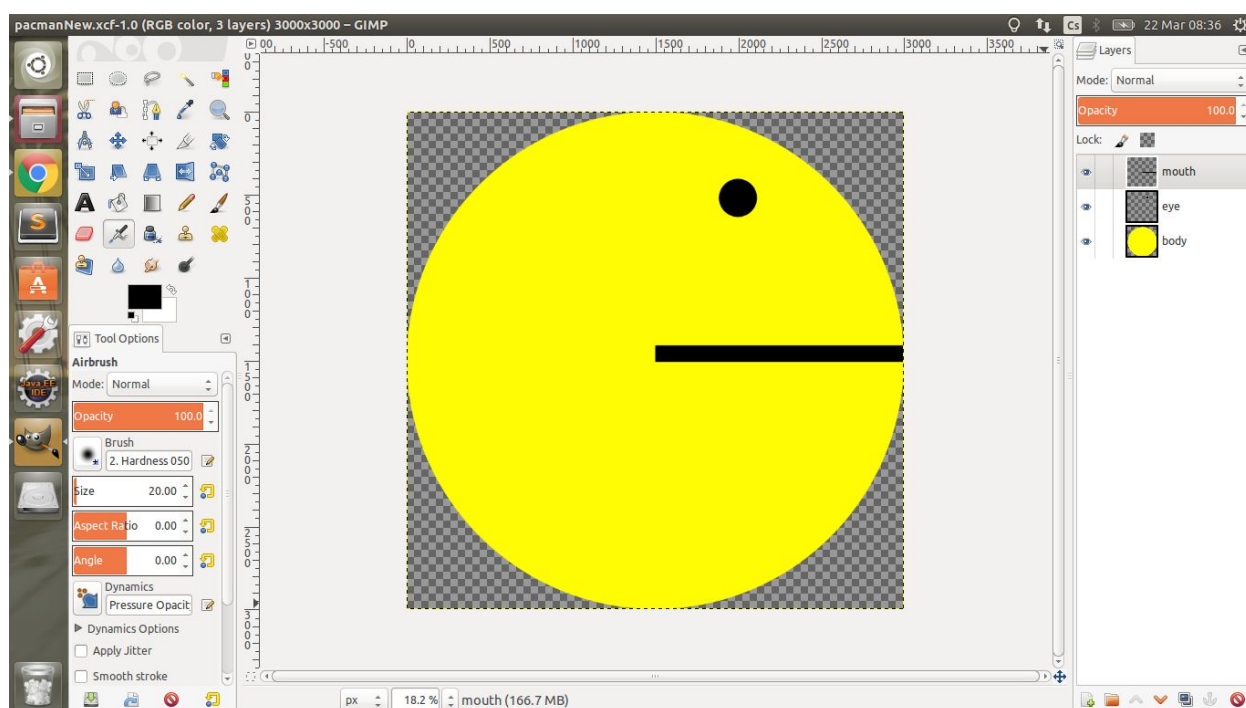


## 3 Návrh a tvorba GUI

### 3.1 GIMP



Pro tvorbu svých jednoduchých textur jsem si zvolil program, který se nazývá GIMP. Tuto volbu jsem učinil z několika důvodů. Je multiplatformní a to je pro linuxového uživatele jako jsem já dobrá zpráva. Práce s ním je velice jednoduchá a intuitivní a jako poslední a velkou výhodou vidím v GNU GPL licenci, díky které je zcela zdarma.



#### Tvorba obrázku v programu GIMP 2.8

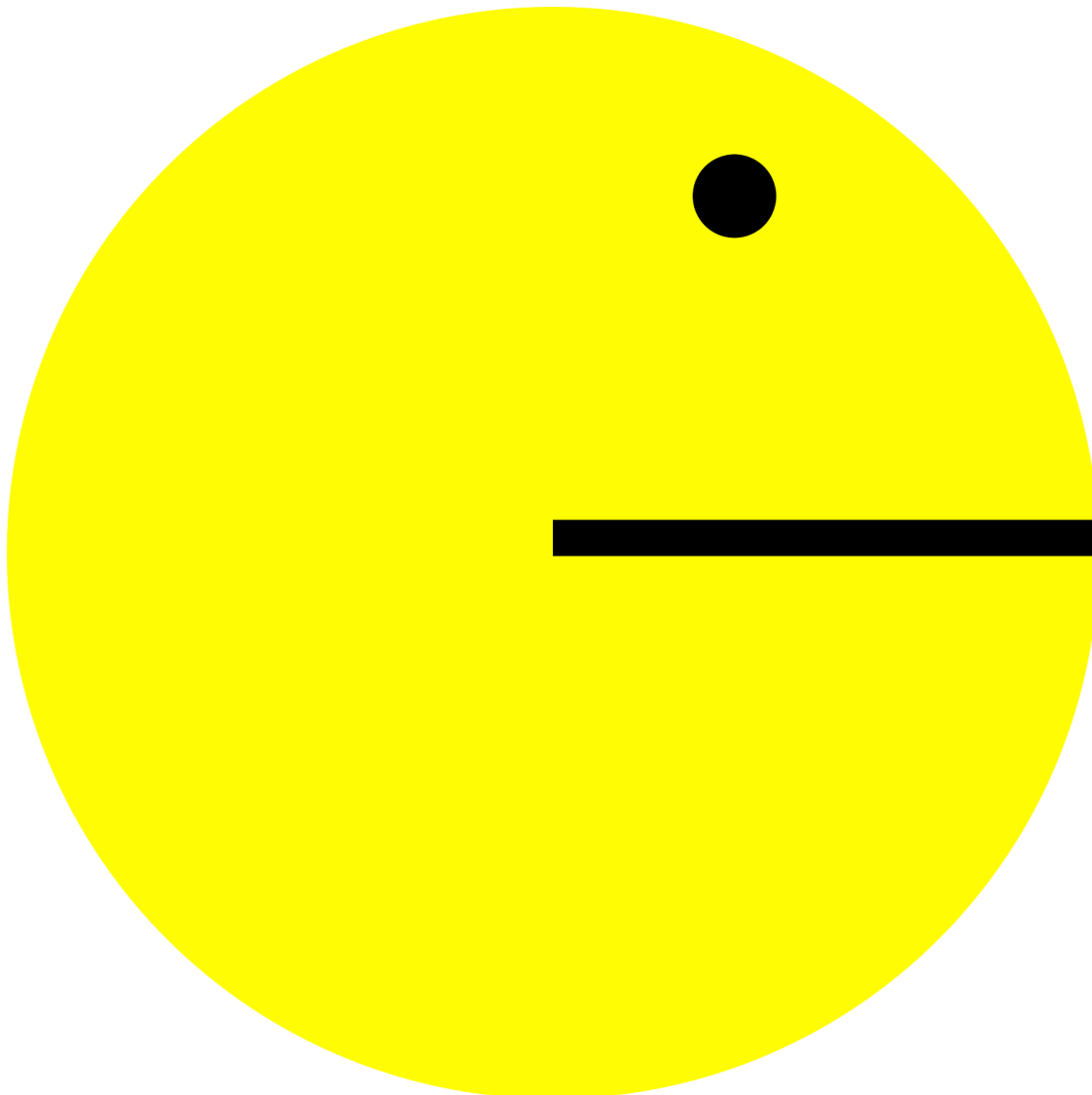
Vlevo nahoře se nachází panel nástrojů, který je velice podobný panelu nástrojů z programu na rastrovou grafiku od Adobe jménem Photoshop. Hned pod nástroji se nachází Tool Options, kde si můžeme nastavit nástroj, který chceme zrovna používat.

Vprostřed obrázku máme pracovní plochu, na které upravujeme a vytváříme finální obrázky.

Vpravo se nachází panel s vrstvami.

Toto rozpoložení je moje vlastní. Každý uživatel si ho může přizpůsobit dle svých potřeb a může zvolit i jiné lišty, pokud je využívá často.

## 3.2 Tvorba grafiky

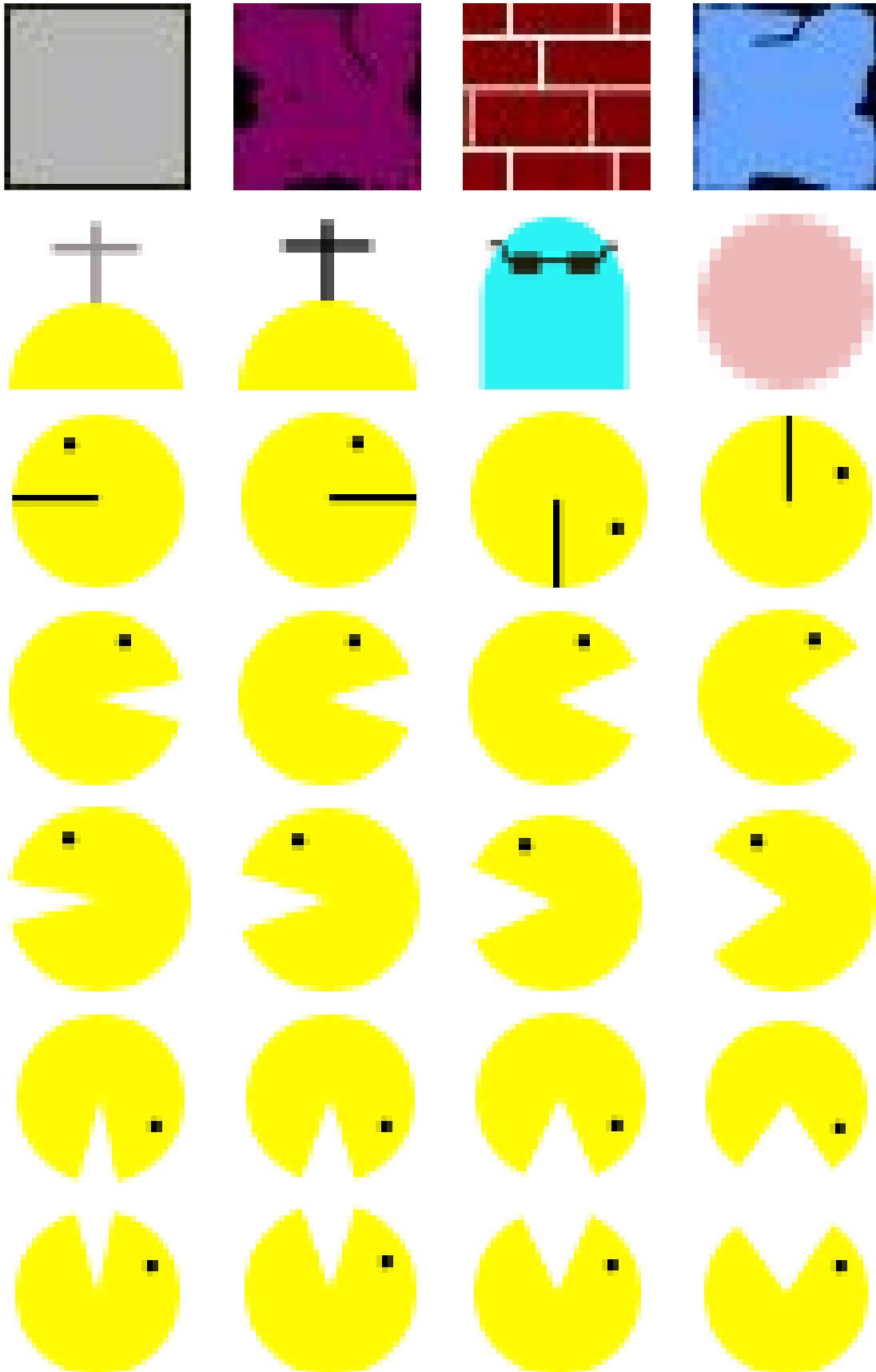


### Design postavy pacmana

Tvorba grafiky pro mojí hru byla velice jednoduchá. Založil jsem si novou vrstvu o velikosti 3000x3000 pixelů. Zvolil jsem si výběr kruh a za pomoci nástroje plechovka jsem ho vyplnil žlutou barvou. Pak už zbývalo pouze za pomoci kruhu a obdelníku vytvořit oko a ústa mému hlavnímu hrdinovi.

Návrh hlavního hrdiny byl tedy hotov. Dalším krokem bylo jej zmenšit a to stokrát aby měl požadovanou velikost 30x30 pixelů. Poslední krok bylo vyexportovat tento zmenšený obrázek ve formátu .png a to z důvodu průhledného pozadí, které tento formát podporuje.

### 3.3 Přehled vytvořených textur

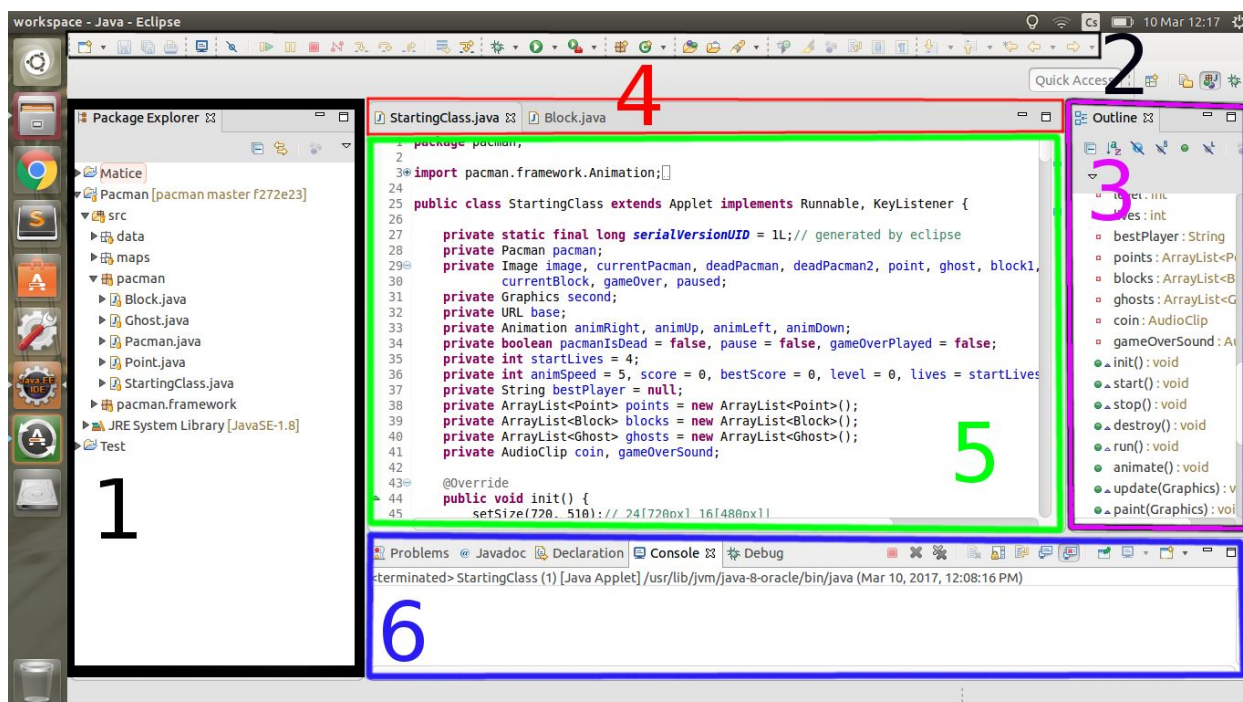


## 4 Programy použité při tvorbě

### 4.1 Eclipse 4.6.2



„Eclipse je open source vývojová platforma, která je pro většinu lidí známa jako vývojové prostředí (IDE) určené pro programování v jazyce Java. Flexibilní návrh této platformy dovoluje rozšířit seznam podporovaných programovacích jazyků za pomoci pluginů, například o C++ nebo PHP. Právě pluginy umožňují toto vývojové prostředí rozšířit například o návrh UML, či zápis HTML nebo XML.“<sup>[1]</sup>



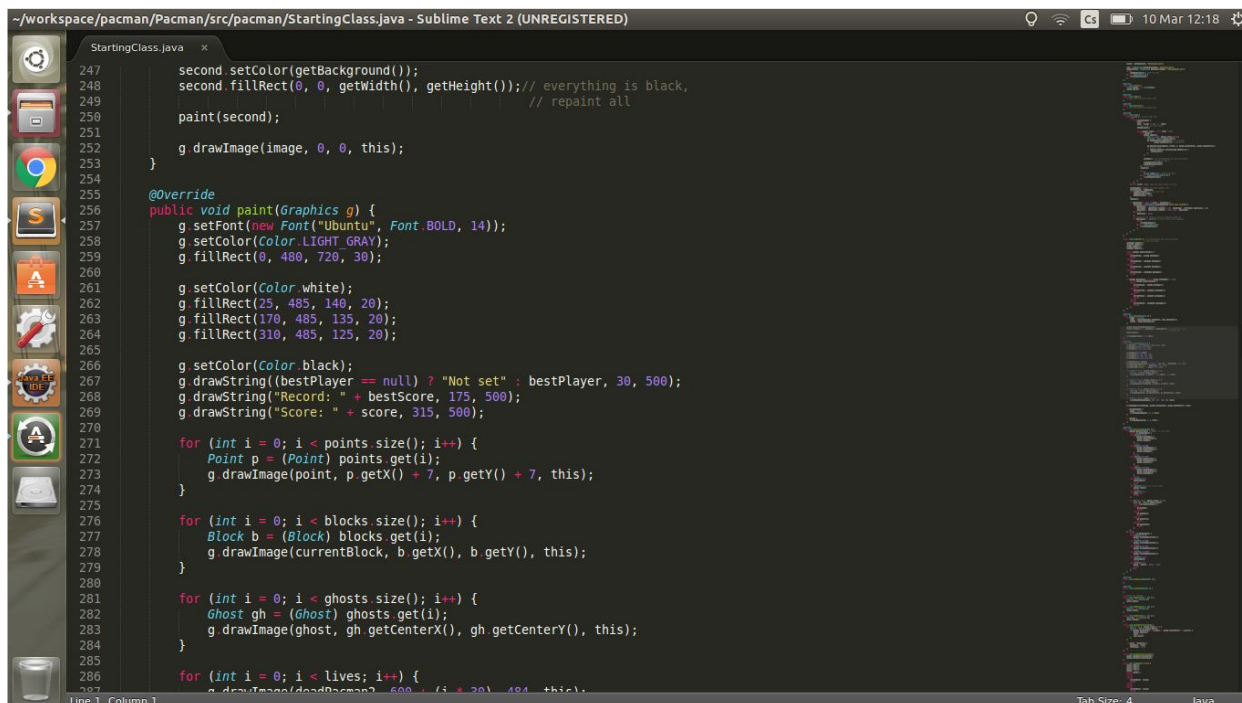
#### Vývojové prostředí Eclipse verze 4.6.2

1. Seznam projektů a jejich soubory
2. Panel nástrojů
3. Přehled atributů a metod třídy, ve které pracujeme
4. Otevřené soubory, které editujeme
5. Textový editor
6. Konzole pro výpis dat, chyb a zachycených výjimek

## 4.2 Sublime Text 2



Jedná se o multiplatformní textový editor. Je velice přehledný a zná syntaxi mnoha programovacích jazyků. Pokud by uživateli nestačil samotný textový editor je možnost si vybrat z velkého množství doplňků.



### Ukázka kódu v textovém editoru Sublime Text 2

Jak můžete vidět na obrázku text je barevný a lépe se v něm orientuje.

Vlevo je číslování řádků a vpravo je zmenšený vzhled kódu a průsvitný obdelník ukazující v jaké části kódu se zrovna nacházíme.

V pravém dolním rohu si můžeme vybrat s jakým programovacím jazykem budeme pracovat a editor se pak přizpůsobí vybrané syntaxi tohoto jazyku.

## 4.3 Git

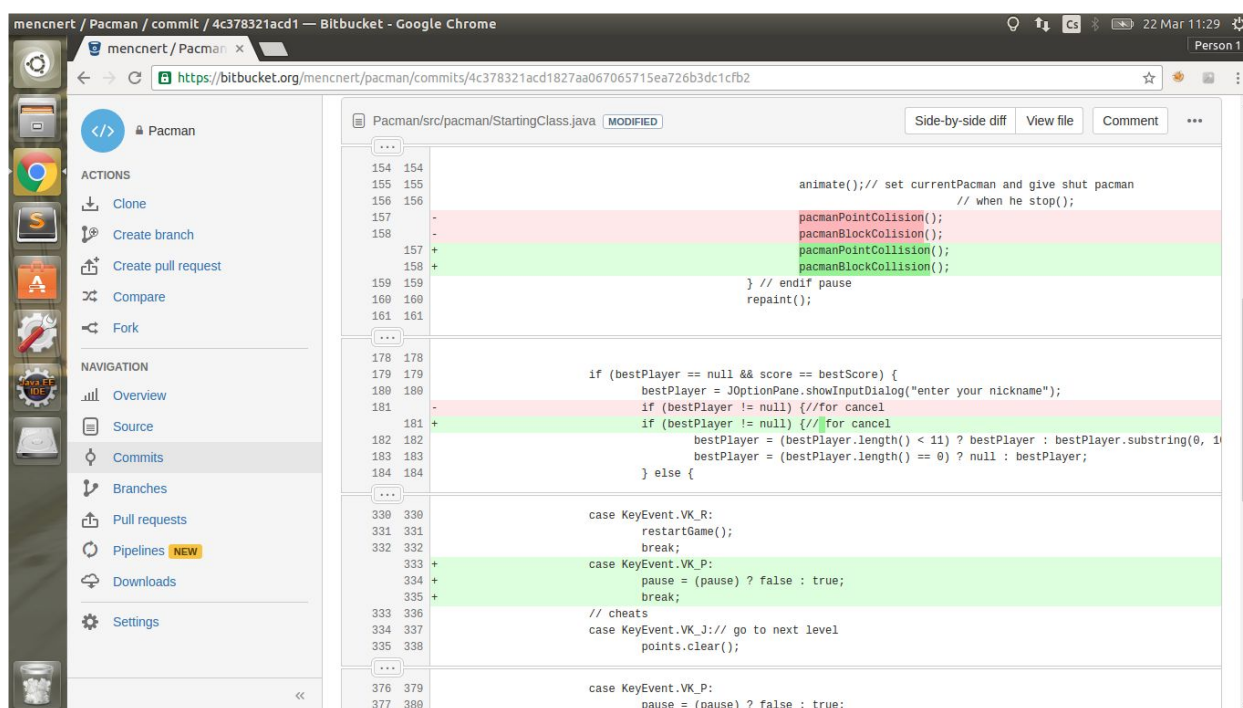


*„Git je v informatice distribuovaný systém správy verzí vytvořený Linusem Torvaldsem pro vývoj jádra Linuxu. Návrh Gitu byl ovlivněn projekty BitKeeper (dříve používán pro vývoj jádra Linuxu) a Monotone. V současnosti (2016) je projekt Git používán mnoha známými projekty, například: jádro Linuxu, X.Org nebo Ruby on Rails. Projekt spravuje Junio Haman, je šířen pod GPL verze 2 a jedná se o svobodný software.“[2]*

## 4.4 Bitbucket



Bitbucket je webová služba podporující verzovací program Git. Prostředí této služby je velice pohodlné pro práci s projekty. Na projektech může pracovat více lidí a díky programu Git máte vždy aktuální verzi. Velice přehledně na Bitbucket můžete sledovat jak se program vyvíjí a jednotlivé změny provedené v kódu. U privátního projektu je služba zdarma do 5 lidí a u veřejných projektů je počet osob neomezený.



**Ukázka: commit 4c37832**

### 4.4.1 Verzování

Provedl jsem několik změn ve třídě StartingClass.java a vytvořil commit za pomoci příkazů:

```
/workspace/pacman$ git add -A
```

```
/workspace/pacman$ git commit -m "zpráva, kterou chceme popsat změny"
```

```
/workspace/pacman$ git push -u origin master
```

Zadáme heslo, které máme na službě Bitbucket a commit je odeslán na serveru, kde se zobrazí ve výpisech jak můžeme vidět na obrázku před textem.

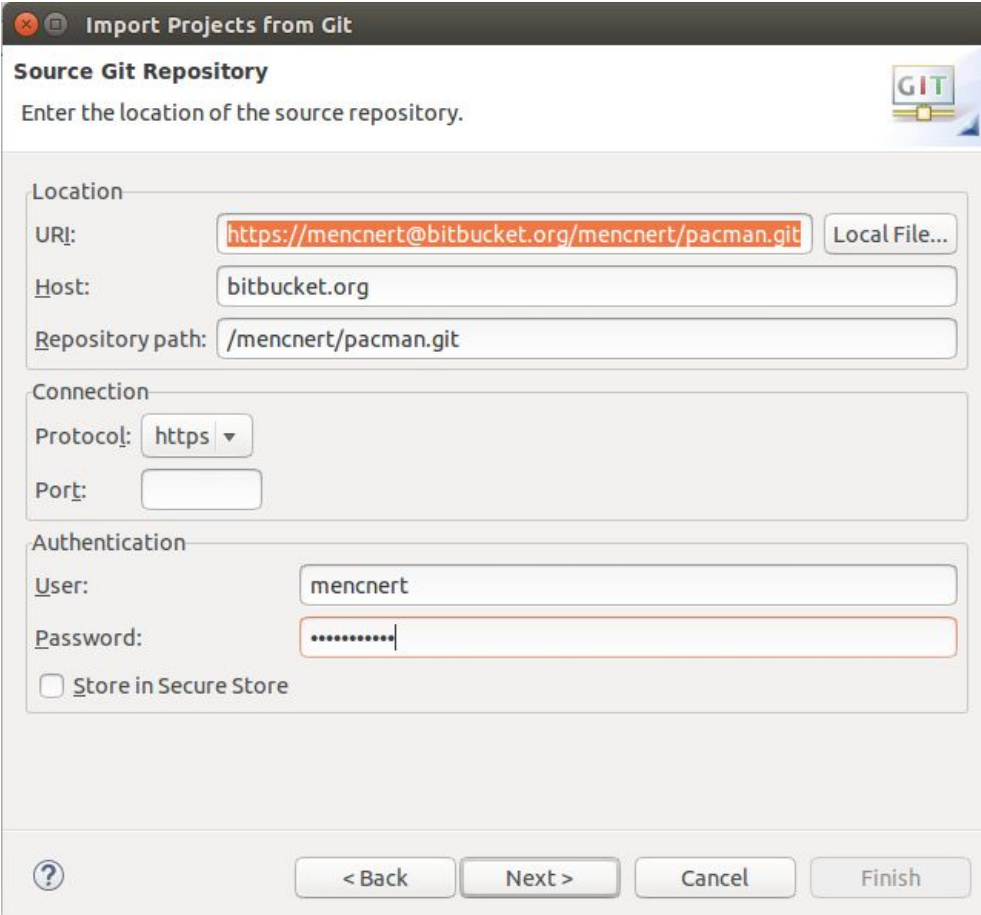
Pokud pracujete ve více lidech nebo na více počítačích, stačí zadat příkaz:

```
/workspace/pacman$ git pull origin master
```

Po provedení autentizace nastává autorizace v podobě stažení nejnovější verze projektu a můžete pracovat.

## 4.4.2 Ztráta dat

Během vývoje jsem simuloval i ztrátu všech dat abych zjistil co všechno je zapotřebí udělat pro naběhnutí do starých kolejí. Nesimuloval jsem však ztrátu celého systému, ale pouze projektu, který jsem smazal. Nemusel jsem tedy instalovat nový operační systém ani programové vybavení počítače, což určitě zabere pár hodin a přidá pár vrásek. Pro mě to znamenalo jít na Bitbucket a za pomoci jednoho kliknutí si nechat vygenerovat příkaz **“git clone https://mencnert@bitbucket.org/mencnert/pacman.git”** v terminálu se nastavit do složky s projekty a pouze příkaz provést. Následně v programu Eclipse importovat existující Git repository. Nebo využít importování za pomoci URL adresy.



### Ukázka importování za pomoci URL adresy.

Ztráta projektu pro mě tedy znamenala maximálně pěti minutové zdržení.

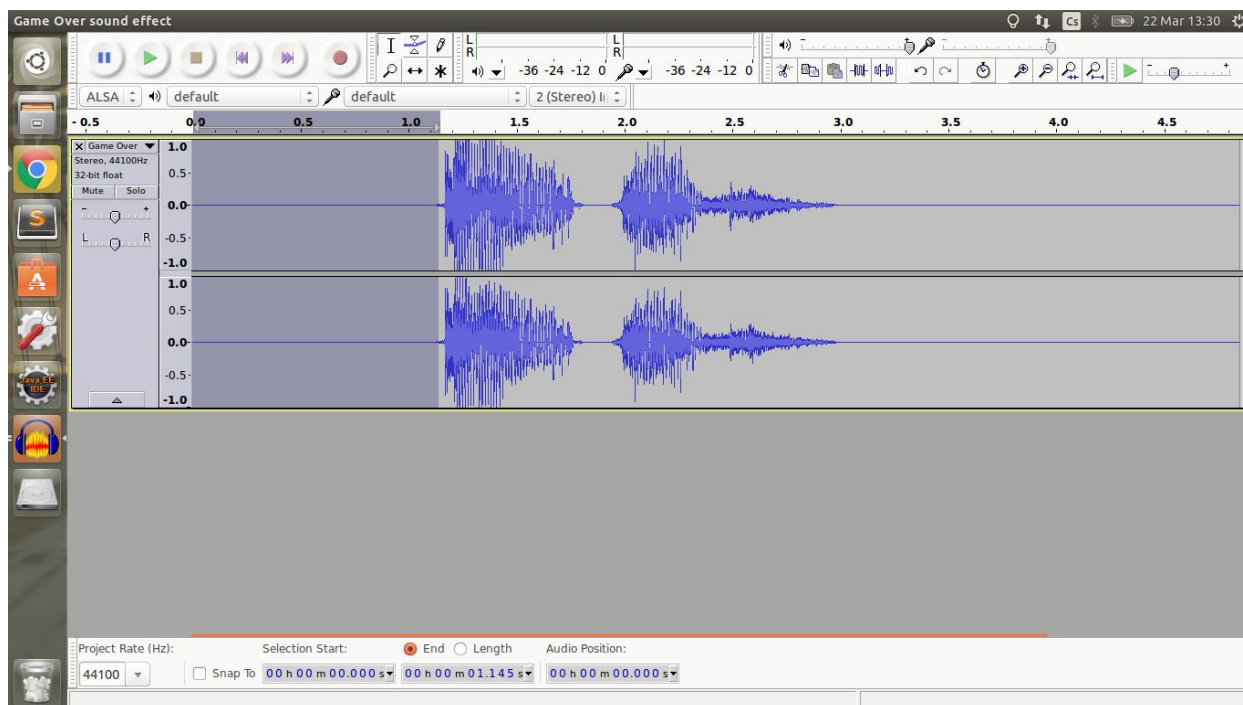
Verzování projektu ovšem nemůžete brát jako všemocné. Je doporučeno si dělat zálohy celého projektu. Může se stát, že vlastním zapříčiněním si smažete veškerou práci ze služby Bitbucket a tím je vaše práce nenávratně zničena, protože služba Bitbucket nevytváří zálohy pro takový případ.



## 4.5 Audacity



„Audacity je svobodný/open source, multiplatformní editor digitálního zvuku.“<sup>[3]</sup>



### Úprava zvuku v programu Audacity.

Moje práce s Audacity byla velice krátká. Jediné co jsem upravoval na stažených zvucích uvedených v seznamu zdrojů byla délka zvukového záznamu. Jakmile mi nahrávka pasovala pro můj účel, exportoval jsem ji ve formátu wav.

Třída pro přehrávání zvuku AudioClip obsažená v knihovně java.applet podporuje formát wav. Při programování jsem zkoušel přehrávat formát mp3 ale později jsem zjistil, že tento formát třída AudioClip nepodporuje.



## 5 Vývoj hry

### 5.1 Seznam úkolů

Před samotným programováním jsem si sepsal seznam, kterým jsem se v průběhu vývoje řídil.

- vykreslování map ze souborů
- výpočet startovních souřadnic všech objektů
- **pohyb pacmana po matici**
- **kolize pacmana s okrajem hrací plochy**
- **kolize pacmana se stěnami**
- kolize pacmana s růžovými body
- počítání skóre
- pohyb duchů po matici
- kolize duchů se stěnami
- kolize duchů s jinými duchy
- kolize pacmana s duchy
- po kolizi pacmana s duchy umístit všechny pohyblivé objekty na startovní pozice a čekat na stisknutí klávesy
- vytvoření levelového systému s nekonečným cyklem
- zapsat všechny levely do souborů
- výpis nejlepšího hráče
- ozvučení

## 5.2 Ukázka kódu

Ukázka kódu se týká tří tučně zvýrazněných bodů ze seznamu úkolů.

```
public void changeDirection(ArrayList<Block> blocks) {
    if (this.getDirection() != this.getChangeDirection()) {
        if (this.getCenterX() % 30 == 0 &&
            this.getCenterY() % 30 == 0) {
            switch (this.getChangeDirection()) {
                case 1:// up
                    wait = false;
                    for (int i = 0; i < blocks.size(); i++) {
                        Block b = (Block) blocks.get(i);
                        if ((getCenterX() == b.getX() &&
                            getCenterY() - 30 == b.getY()) ||
                            getCenterY() == 0) {
                            wait = true;
                        }
                    }

                    if (!wait) {
                        this.stop();
                        this.setDirection(1);
                        this.moveUp();
                    }
                    break;
                case 2:// down
                    ...
                    break;
                case 3:// left
                    ...
                    break;
                case 4:// right
                    ...
                    break;
            } // end switch
        } // end if for x%30 && y%30
    }
}
```

Proměnná `Direction` určuje aktuální směr a proměnná `changeDirection` je nositel informace, který směr si uživatel nově zvolil. Jakmile uživatel chce změnit směr a podmínka vstoupí v platnost, nastává vyhodnocení nové podmínky, která čeká až se bude pacman nacházet na souřadnicích, které jsou dělitelné třiceti. Tímto zajistíme pohyb v předem určených dráhách. Pro další vysvětlení kódu si definujeme `direction = 3` (pacman se pohybuje po ose x doleva) a `changeDirection = 1` (uživatel chce nastavit směr nahoru). První podmínka je splněna a druhá

bude splněna jakmile se pacman bude nacházet na souřadnicích dělitelných třiceti. Pacman se pohybuje po třech pixelech, tak že podmínka nemůže selhat.

Druhá podmínka je splněna během maximálně devíti herních cyklů a switch rozhoduje, která část kódu bude vykonána. Proměnná `changeDirection` nabývá hodnoty 1. Provede se první blok přepínače.

V prvním bloku se vyresetuje hodnota proměnné `wait`, která je typu `boolean`. Proměnná `wait` rozhoduje zda pacman může změnit směr. V současné chvíli nabývá hodnoty `false`. Nastává cyklus, který projde všechny instance typu `Block`. Postupně nahrává objekty z pole objektů a ověřuje, zda-li souřadnice `x` od pacmana a bloku (zdi) se schodují a zároveň pacmanova souřadnice `y` zmenšená o 30 se schoduje s `y`-ovou souřadnicí bloku nebo se pacman nachází u horního okraje hrací plochy, poté se `wait` nastaví na `true` a nedojde ke změně směru.

Kód řídí hladký průběh změny směru. Nachází-li se pacman hned vedle stěny a stěna mu brání ve změně směru, pouze kolem ní projíždí až do chvíle kdy je cesta volná nebo narazí do jiné stěny. V případě, že se pacman nachází na horní hraně hrací plochy, pokračuje ve směru až do chvíle kdy narazí a zastaví se.

Pro jiné směry je princip stejný, pouze se liší přičítání a odečítání hodnot od souřadnic `x` a `y`, které kontrolujeme.

## 5.3 Ovládání hry

šipka nahoru - pohyb nahoru

šipka dolů - pohyb dolů

šipka vlevo - pohyb vlevo

šipka vpravo - pohyb vpravo

r - restart celé hry

p - pauza

j - přeskočí level

l - doplní životy

Písmena `j`, `l` jsou tzv. *cheaty*. Do hry jsem je přidal pro snažší pohyb ve hře při kontrolování práce. Cheaty lze použít pouze při startu hry, když vše stojí.

## 6 Testování

Při vývoji hry se vyskytla chyba při kolizi duchů a zhruba týden jsem se snažil zjistit čím je způsobená. Až dlouhé testování, debugování a studování každého detailu kódu, který s kolizemi souvisí ukázalo čím je způsobená.

### 6.1 Popis a řešení problému

Při kolizi nastávalo několik problémů. Občas se začal duch pohybovat směrem, který není vůbec přijatelný ke zbytku kódu, občas se z části dostali do bloku a jejich pohyb se naprosto zastavil a nebo dva duchové v uličce ohraničené bloky se zpomalili a trvalo jim nepřiměřeně dlouho, než uličku při nezměněném směru opustili.

Problém byl v tom, že jsem zapomínal aktualizovat pozici rectanglu po tom co duch kolidoval a posunul se o 3 pixely zpět.

- 1) Duch 1 a duch 2 jsou vedle sebe bez kolize.
- 2) Duch1 se posune o 3 pixely a zjistí, že koliduje s duchem2 a tak se vrátí o 3 pixely zpět a vybere si nový směr. Avšak rectangl zůstane posunut o 3 pixely.
- 3) Duch2 se posune a kolize rectanglů těchto dvou duchů je již 6 pixelů a on se vrátí o 3 pixely ale rectangl ducha1 a rectangl ducha2 zůstanou a překrývají se o 6 pixelů.
- 4) Duch1 se posune v nově zvoleném směru a pozice rectanglu se nastaví na správnou pozici ale nastane kolize s rectanglem od ducha2 kvůli posunu 3pixelů a dojde ke změně směru.
- 5) Rectangl ducha2 se nastaví na správnou pozici a nedochází ke kolizi.
- 6) Duch1 míří do ducha2 a koliduje...

Řešení bylo mnohem snazší, než zjistit čím je chyba způsobena. Stačí aktualizovat pozici rectanglu po navrácení ducha o 3 pixely v opačném směru než se pohyboval dokud nekolidoval.

## 7 Závěr

Maturitní práce pro mne byla velkým přínosem. Hodně jsem si rozšířil obzor v problematice programování a naučil jsem se samostatně řešit problémy a vyhledávat informace k problémům. Vyhledávání informací považuji za nejdůležitější dovednost, kterou jsem nabyl v průběhu práce. Nebýt vyhledávání, tak jenom těžko bych svou práci dokončil. Vyhledávání také vděčím za dovednost verzování, která je velkým přínosem do stávajících dovedností řízení projektu.

Programování je činnost, která mě baví a do budoucna uvažuji o větších a propracovanějších programech, které by mohly být užitečné nejen pro mě ale i pro ostatní. Osobně upřednostňuji práci v týmu, kde každý dělá to co umí nejlépe a nemusí řešit problémy, kterým nerozumí a ani ho nebaví je řešit. Mezi mé klady patří programování a vymýšlení algoritmů založených na matematice, logickém uvažování a mezi zápory naopak, práce s grafikou, design a tvorba zvuku. Proto jsem svou práci založil na jednoduché grafice a složitějších algoritmech.

Rád bych poděkoval panu Mgr. Zdeňku Přikrylovi mladšímu za konzultace na hodinách a za jeho trpělivost se všemi studenty během celého školního roku, který je také naším posledním na VOŠ a SPŠ v Šumperku.

Můj program je téměř k nerozeznání od původní hry jménem pampuch. Princip chování duchů jsem přejal z původní hry a sám naprogramoval. Devět map z deseti je původních. Úvodní mapu jsem si vytvořil vlastní.

Můj program by jistě mohl být optimalizovanější za pomoci vhodnější architektury programu, principů Booleovy algebry a jiných konceptů. Dále nevyužívám principy dědičnosti, které bych mohl využít u několika tříd. Avšak tyto poznatky jsou velkým plusem pro další projekty, kde již budu myslet na možnosti, které jsem uvedl.

## 7.1 Instrukce ke spuštění

- 1) Nainstalujte jdk-8.
- 2) Nainstalujte program Eclipse a během instalace zvolte možnost „Eclipse IDE for Java EE Developers“.
- 3) Nahrajte složku pacman z přiloženého CD k dokumentaci do složky workspace, kterou vytvořil program Eclipse při instalaci.
- 4) Vytvořte nový projekt, **File - new - Java Project**.
- 5) Nastavte jméno projektu na „pacman“ a celý projekt by se měl automaticky importovat.
- 6) Při spuštění projektu se zobrazí tabulka a zvolte možnost „applet“.

Při problémech mě kontaktujte na mail: [menclert.ita13@vsps-su.cz](mailto:menclert.ita13@vsps-su.cz)

## 8 Zdroje

- [1] [https://cs.wikipedia.org/wiki/Eclipse\\_\(v%C3%BDvojov%C3%A9\\_prost%C5%99ed%C3%AD\)](https://cs.wikipedia.org/wiki/Eclipse_(v%C3%BDvojov%C3%A9_prost%C5%99ed%C3%AD)) [2017-03-26]
- [2] <https://cs.wikipedia.org/wiki/Git> [2017-03-26]
- [3] <https://cs.wikipedia.org/wiki/Audacity> [2017-03-26]
- [4] <https://cs.wikipedia.org/wiki/GIMP> [2017-03-26]
- [5] <https://www.youtube.com/watch?v=vJaAy3jmEx8> [2017-03-26]
- [6] <https://www.youtube.com/watch?v=Rfkcl8dhfsQ> [2017-03-26]
- [7] <https://git-scm.com/images/logos/downloads/Git-Icon-1788C.png> [2017-03-26]
- [8] [https://upload.wikimedia.org/wikipedia/en/4/4c/Sublime\\_Text\\_Logo.png](https://upload.wikimedia.org/wikipedia/en/4/4c/Sublime_Text_Logo.png) [2017-03-26]
- [9] <https://eclipse.org/artwork/images/v2/eclipse-mp-built-800x274.png> [2017-03-26]
- [10] <https://www.brandeps.com/logo-download/B/Bitbucket-01.png> [2017-03-26]
- [11] <http://www.audacityteam.org/wp-content/uploads/2015/11/small-audacity.png> [2017-03-26]
- [12] <http://logonoid.com/images/gimp-logo.png> [2017-03-26]
- [13] <http://www.kilobolt.com/game-development-tutorial.html> [2017-03-26]
- [14] SCHILDT, Herbert. Mistrovství - Java. Brno: Computer Press, 2014. Mistrovství. ISBN 978-80-251-4145-8.