Pac-Man

Gymnázium, Praha 6, Arabská 14

Předmět Programování, vyučující Jan Lána

Marek Daňa 4. E duben 2020

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 Praze dne 30. dubna 2020 *Podpis: Marek Daňa*

Anotace

Tato práce se zabývá naprogramováním 2D hry Pac-Man. Ve hře hráč ovládá postavu Pac-Man a má za úkol posbírat všechny mince v bludišti. V plnění úkolu se mu pokouší zabránit duchové, kteří se pohybují v bludišti. Hráč může sebrat bonus, aby duchy zastavil. Hra se mírně odlišuje od původní hry a obsahuje 3 různé úrovně. Duchové jsou celkem 4 a 3 z nich se pohybují po určené dráze. Pronásledují hráče, pouze když se přiblíží. Poslední duch hráče následuje neustále.

Annotation

This work deals with the programming of 2D game Pac-Man. In the game player controls character of Pac-Man and his job is to collect coins in the maze. The ghosts, which move in the maze try to prevent him from achieving this goal. Player can use bonus to stop the ghosts. The game has 3 levels and is slightly different from original game. Overall there are 4 ghosts – 3 of them follow their trace and follow player only when he is nearby. Last of the ghosts follows player all the time.

Zadání projektu

Zadáním je hra ve stylu známé hry Pac-Man. Hráč ovládá smajlíka (Pac-Man), který se nachází v bludišti (vpravo a vlevo na hrací ploše jsou vchody do tunelu => průchod z jedné strany na druhou) a jeho úkolem je sesbírat všechny body v tomto bludišti. V bludišti se nachází duchové, kteří se snaží hráče zastavit. Na obranu proti těmto duchům může hráč nalézt bonusový předmět, který mu umožní duchy zastavit.

Upřesněné zadání:

- Aplikace poběží v prohlížeči a bude založená na JavaScriptovém frameworku Phaser 3.

Platforma:

- Windows
- JavaScript – Phaser 3

Obsah

Úvod

V této práci bych chtěl pojednat o mé ročníkové práci z programování na téma Pac-Man. Popisuji zde, jak je projekt strukturován a jak funguje. Tuto hru jsem si vybral, protože jsem ji ve svých mladších letech rád hrál. Zajímalo mě, jak obtížné by mohlo být naprogramovat si vlastní verzi této hry. Dále jsem chtěl využít toho, že jsem se minulý rok v rámci týmové ročníkové práce naučil pracovat s Phaserem 3 a tudíž jsem to chtěl dále zužitkovat. Doufám, že v práci dále prohloubím své znalosti Phaseru 3 a procvičím se ve vytváření webových her.

1) Použité technologie

Jako vývojové prostředí jsem použil **Visual Studio Code**. Hra byla naprogramována v JavaScriptovém frameworku **Phaser 3**, který je určen na vytváření 2D her v prohlížeči. Kromě toho je v projektu použita knihovna **EasyStar**, která má za úkol starat se o pohyb v bludišti. Jako kompilátor webových aplikací je použit **Parcel**. Ke spuštění projektu je nutné mít **Node.js** v prostředí **Windows**. Pro vytváření grafické části hry byl využit **GIMP** a **Malování**. Ke tvorbě jednotlivých úrovní jsem využil mapový editor **Tiled**.

2) Struktura projektu

Program se dělí na dvě hlavní části. První je HTML část, která zobrazí webové stránky s hrou a spustí script **Phaseru 3**. Druhou částí je samotný kód hry napsaný v **Phaseru 3** (JavaScriptu).

Zdrojový kód je rozložen do několika souborů s příponou „.js“. V souboru *main.js* se inicializuje třída *Phaser.Game*, která se stará o chod aplikace. Nastaví se v něm velikost okna, způsob renderování, fyzika, používané scény. Základním principem **Phaseru 3** je přepínání mezi jednotlivými scénami, přičemž každá scéna má vlastní soubor a funkci.

Každá scéna má základní metody **init**, **preload**, **create** a **update** a vlastní proměnné. Metoda **init** slouží k načtení dat z předchozí scény. Metoda **preload** načítá soubory (obrázky, mapy) ze složky *assets*. V metodě **create** se vytváří třídy, objekty, skupiny, animace atd. Metoda update je nekonečný cyklus, který se stará o běh hry.

Z datových struktur je v projektu nejvíce zastoupen primitivní var pro uložení jednoduchých informací – například jméno hráče, počet životů atd. Dále je využíváno pole, třídy, metody, či objekt pro uložení hry.

2.1) Main.js

V tomto souboru se vytváří třída hry a nastavují se její vlastnosti. Nastavuje se zde velikost okna hry, způsob renderování, počet FPS atd. Také se importují scény (exportují se z ostatních .js souborů), které má hra využívat.

```
let game = new Phaser.Game({
    type: Phaser.AUTO,
    width: 1280, // sirka okna
    height: 896, // vyska okna
    canvas: document.getElementById('myCustomCanvas'),
    // seznam scen
    scene: [
        MenuScene, PlayScene, PauseScene, LeaderBoardScene, PlayScene2, PauseScene2, PlayScene3, PauseScene3

    ],
    audio: {
        disableWebAudio: false
    },
    render: {
        pixelArt: true
    },
    // nastaveni fyziky (typ, gravitace, fps ..)
    physics: {
        default: "arcade",
        arcade: {
            fps: 60,
            gravity: { y: 0 },
        }
    }
});
```

Obrázek 1- inicializace hry

2.2) MenuScene.js

Tato část kódu vytváří hlavní nabídku a je to první scéna, která se zobrazí, když hráč hru spustí.

V metodě **preload** se načtou soubory pro celou hru. Dále se každému souboru nastaví jméno, aby se poté na něj dalo odkázat.

V metodě **create** se vytvoří a nastaví pozadí a tlačítka (Play a Leaderboard). Každé spouští rozdílnou scénu.

V metodě **update** se kontroluje, zda hráč stiskl některé z tlačítek a případně spustí další scénu. Při zmáčknutí tlačítka Play se hráči objeví možnost zadat svou přezdívku.

```
background = this.add.image(640, 448, "background")
title = this.add.image(640, 100, 'title');
playbutton = this.add.image(640, 250, 'play');
leaderboardbutton = this.add.image(640, 320, 'leaderboard');

playbutton.setInteractive();
leaderboardbutton.setInteractive();


playbutton.on('pointerdown', function (pointer) {
    Menu = 1;

    playerName = prompt("Please enter your nickname", "Pacman");
    if (playerName === null) {
        playerName = "Pacman";
    }
    else {
        while (playerName.length > 20) {
            alert('Please enter a shorter name');

            playerName = prompt("Please enter your nickname", "Pac
        }
        localStorage.setItem("playerName", playerName);
    }
});
```

Obrázek 2- zadáni jména

2.3) LeaderboardScene.js

Do této scény se lze dostat z hlavního menu a zobrazí se v ní seznam deseti nejlepších hráčů. Ze souboru se načte pole s deseti nejlepšími hráči a jejich skóre. Poté se vytvoří 20 textových polí, která jsou ve dvou sloupcích, do prvního sloupce se z pole načtou jména a do druhého sloupce se načte skóre.

```
    scoreField = JSON.parse(localStorage.getItem("score"));

    for (var i = 0; i < 10; i++) {
        scoreText.push(this.add.text(800, 198 + i * 65));
        nameText.push(this.add.text(400, 198 + i * 65));

    }
    if (scoreField !== null && scoreField !== undefined) {
        x = scoreField.length;
        if (x > 10) {
            x = 10;
        }
        for (var i = 0; i < 10; i++) {
            if (i < x) {
                scoreText[i].setText(" " + scoreField[i].score);
                nameText[i].setText(i + 1 + ": " + scoreField[i].playerN
            }
            else {
                nameText[i].setText(i + 1 + ": ");
            }
        }
    }
    else {
        for (var i = 0; i < 10; i++) {
            scoreText[i].setText(i + 1 + ": ");
        }
    }
```

Obrázek 3- načtení žebříčku

2.4) PauseScene.js

Tato scéna (a jejich varianty pro danou PlayScene) se spustí v úrovních, když hráč zmáčkne klávesu escape. Hráč může hru opětovně rozběhnout, nebo se vrátit do hlavního menu, přičemž přijde o veškerý dosavadní postup a neuloží se mu počet bodů.

2.5) PlayScene.js

Všechny *PlayScene[n].js* soubory mají obdobnou strukturu, liší se pouze odlišnými hodnotami a rozdílnou mapou.

V této scéně se nejdříve vytvoří proměnné. Poté následují metody: **distance** (dostane 4 souřadnice a vrátí vzdálenost mezi nimi), **getTileID** (vrací ID na daných souřadnicích), **collectCoin** (odstraní minci a přičte hráči body), **collectBonus** (aktivuje bonusový efekt a odstraní bonus z herní plochy), **damageR** (a 3 obdobné, které obsluhují událost kolize mezi duchem a Pac-Manem, kolizi detekuje Phaser a v případě, že nastane, spustí metody **damage[X]**).

```
// metoda, ktera vypocita a vrati vzdalenost mezi 2 body
function distance(x1, y1, x2, y2) {
    var dx = x1 - x2;
    var dy = y1 - y2;

    return Math.sqrt(dx * dx + dy * dy);
}

// metoda, ktera vrati ID tile na dane souradnici
function getTileID(x, y) {
    var tile = map.getTileAt(x, y);
    return tile.index;
};

// metoda, ktera se spusti pri kolizi PacMana s coinem, odstrani dany coin z herni plochy a pricte score, ktere zaroven aktualizuje
function collectCoin(PacMan, coin) {
    coin.destroy(coin.x, coin.y);
    coinScore++;
    totalScore++;
    text.setText(`Nickname: ${localStorage.getItem("playerName")}   Score: ${totalScore}`)

    return false;
}
```

Obrázek 4- některé z metod

V metodě **create** se vytváří Pac-Man, duchové, herní plocha, mince, bonusy a různé ukazatele. Dále se přiřadí hodnoty proměnným a nastaví se vstup z klávesnice.

```
// vytvoreni mapy podle .json mapy
map = this.add.tilemap("map1");
terrain = map.addTilesetImage("Bloky", "terrain")
topLayer = map.createStaticLayer("top", [terrain], 0, 0).setDepth(-1)

// vytvoreni jednotlivych vrstev mapy podle .json souboru
CoinLayer = map.getObjectLayer('points')['objects'];
coins = this.physics.add.staticGroup()
CoinLayer.forEach(object => {
    let obj = coins.create(object.x + 16, object.y - 16, "coin");
    obj.setScale(0.5)
    obj.body.width = object.width;
    obj.body.height = object.height;
});
BonusLayer = map.getObjectLayer('bonus')['objects'];
bonus = this.physics.add.staticGroup()
BonusLayer.forEach(object => {
    let obj = bonus.create(object.x + 16, object.y - 16, "bonus");
    obj.setScale(0.5)
    obj.body.width = object.width;
    obj.body.height = object.height;
});
// pokud ma tile vlastnost collide, nastavi se jeho kolize s objekty hry (PacMan a duchove)
topLayer.setCollisionByProperty({ collide: true });

// vytvori PacMan a duchy a nastavi jejich kolize s hranicemi a tily
PacMan = this.physics.add.sprite(PacManRespawnX, PacManRespawnY, "PacMan");
PacMan.setSize(30, 30);
RedGhost = this.physics.add.image(RedGhostRespawnX, RedGhostRespawnY, 'RedGhost');
GreenGhost = this.physics.add.image(GreenGhostDeadX, GreenGhostDeadY, 'GreenGhost');
PurpleGhost = this.physics.add.image(PurpleGhostDeadX, PurpleGhostDeadY, 'PurpleGhost');
GreyGhost = this.physics.add.image(GreyGhostDeadX, GreyGhostDeadY, 'GreyGhost');

this.physics.add.collider(PacMan, topLayer);
this.physics.add.collider(RedGhost, topLayer);
this.physics.add.collider(GreenGhost, topLayer);
this.physics.add.collider(GreyGhost, topLayer);
this.physics.add.collider(PurpleGhost, topLayer);
```

Obrázek 5- Vytvoření bludiště, duchů a nastavení kolizí

V metodě **update** se kontroluje, zda hráč nesebral všechny mince a zda mu neklesly životy na 0. V případě, že hráč dokončí úroveň, uloží se skóre a začne další úroveň. V případě, že mu klesnou životy na 0, uloží se doposud dosažené skóre a spustí se hlavní nabídka.

```
    // casovana udalost dokonceni urovne
    timedEvent = this.time.delayedCall(3000, winEvent, [], this);
    function winEvent() {

        // nacteni skore a pridani nove serazeneho skore do pameti prohlizece
        var testObject = JSON.parse(localStorage.getItem("score"));
        if (testObject !== null) {
            scoreField = JSON.parse(localStorage.getItem("score"));
        }
        var scoreObject = {
            playerName: localStorage.getItem("playerName"),
            score: totalScore,
        };
        if (scoreField === null || scoreField === undefined) {
            scoreField[0] = scoreObject;
        }
        else {
            scoreField.push(scoreObject);


            var length = scoreField.length;
            for (var i = (length - 1); i >= 0; i--) {
                for (var j = (length - i); j > 0; j--) {
                    if (scoreField[j] === undefined) {
                        break;
                    }
                    if (scoreField[j].score > scoreField[j - 1].score) {

                        var tmp = scoreField[j];
                        scoreField[j] = scoreField[j - 1];
                        scoreField[j - 1] = tmp;

                    }
                }
            }
        }
        // uloze pole skore do pameti prohlizece
        localStorage.setItem("score", JSON.stringify(scoreField));
        coinScore = 0;

        // ukonci scenu a spusti dalsi uroven
        this.scene.stop();
        this.scene.launch("PlayScene2", {totalScore: totalScore });

    }
}
```

Obrázek 6- událost v případě sebrání všech mincí

Dále se ve 3 vteřinových intervalech objevují duchové, v případě, že jsou poraženi, se obnoví po uplynutí daného času. V každé úrovni se hráč může přesunout z jedné strany na druhou pomocí průchodů, což je zařízeno jednoduchou podmínkou. Také se zaokrouhluje pozice duchů a hráče z důvodu snažšího pohybu. Dále se hráči nastavuje směr pohybu podle toho, jakou klávesu drží.

```
    // podminky zajistujici pohyb, pouzivaji nadefinovany vstup z klavesnice
    if ((keys.W.isDown || keys.S.isDown || keys.A.isDown || keys.D.isDown)) {

        // pohyb hrace
        if (keys.W.isDown) {
            PacMan.setVelocityY(-1 * speed);
        }
        else if (keys.S.isDown) {
            PacMan.setVelocityY(speed);
        }
        if (keys.A.isDown) {
            PacMan.setVelocityX(-1 * speed);
        }
        else if (keys.D.isDown) {
            PacMan.setVelocityX(speed);
        }

    }
```

Obrázek 7- pohyb hráče

2.6) Knihovna EasyStar

Důležitou součástí projektu je knihovna **EasyStar**, která je určená k hledání cesty v HTML5 hrách. **EasyStar** dostane informace o mapě a je schopný najít cestu mezi 2 body a uložit ji v jednotlivých krocích do pole.

```
    // nastaveni pluginu, ktery se stara o hledani cesty v ramci tilu mapy
    var easystarjs = require('easystarjs');
    var easystar = new easystarjs.js();
    finder = easystar;
    finder2 = easystar;
    finder3 = easystar;
    finder4 = easystar;
    grid = [];

    // vlozi jednotlive tily do mrizky
    for (var y = 0; y < map.height; y++) {
        var col = [];
        for (var x = 0; x < map.width; x++) {
            col.push(getTileID(x, y));
        }
        grid.push(col);
    }

     var finder2: js id);
    finder2.setGrid(grid);
    finder3.setGrid(grid);
    finder4.setGrid(grid);

    tileset = map.tilesets[0];
    properties = tileset.tileProperties;
    acceptableTiles = [];

    // nastavi ktere tily jsou pruchozi
    for (var i = tileset.firstgid - 1; i < 1120; i++) {
        if (!properties.hasOwnProperty(i)) {

            acceptableTiles.push(i + 1);
            continue;
        }
        if (!properties[i].collide) { acceptableTiles.push(i + 1); }
    }

    finder.setAcceptableTiles(acceptableTiles);
    finder2.setAcceptableTiles(acceptableTiles);
    finder3.setAcceptableTiles(acceptableTiles);
    finder4.setAcceptableTiles(acceptableTiles);

}
```

Obrázek 8- nastavení EasyStar pro danou mapu, dostane informace o tom, který z bloků je průchozí a který ne.

V projektu je knihovna používána k vytvoření trajektorie pohybu duchů. Zelený, fialový a šedý duch se hýbe podél vytvořené trajektorie a pouze v případě, že se hráč přiblíží, začnou ho pronásledovat. Červený duch hráče pronásleduje neustále nejkratší možnou cestou.

```
// finder, diky pluginu easystar js vytvori pole path ve kterem jsou jednotlive tily, ktere vedou k cily nejkratsi cestou
finder.findPath(Math.floor(RedGhost.x / 32), Math.floor(RedGhost.y / 32), Math.floor(PacMan.x / 32), Math.floor(PacMan.y / 32), function (path) {
    if (path === null || path[1] === undefined) {

    } else {
        // pokud neni aktivni bonus, duchove se pohybuji po ceste vytvorene finderem, cesta k hraci se obnovuje a duch nasleduje tuto vytvorenou cestu, tudiz pronasleduje
        if (bonusF === false) {

            if (path[1].x * 32 + 16 > RedGhost.x) {
                RedGhost.setVelocityX(speed / 2);
            }
            if (path[1].x * 32 + 16 < RedGhost.x) {
                RedGhost.setVelocityX(-speed / 2);
            }
            if (path[1].y * 32 + 16 > RedGhost.y) {
                RedGhost.setVelocityY(speed / 2);
            }
            if (path[1].y * 32 + 16 < RedGhost.y) {
                RedGhost.setVelocityY(-speed / 2);
            }

            if (path[1].x * 32 + 16 === RedGhost.x) {
                RedGhost.setVelocityX(0)
            }
            if (path[1].y * 32 + 16 === RedGhost.y) {
                RedGhost.setVelocityY(0)
            }

        }
    }
});
// finder prepocita nalezenou cestu a opět se uloží do path
finder.calculate();
```

Obrázek 9- Pohyb duchů

3) Grafická část projektu

3.1) Textury

Textury jsou uloženy ve formátu PNG. Všechny textury byly vytvořeny v grafických programech **GIMP** a **Malování**. Vzhledem k tomu, že se ve hře nenachází žádné textury náročné na vytvoření, nebylo nutné používat zdroje z internetu.

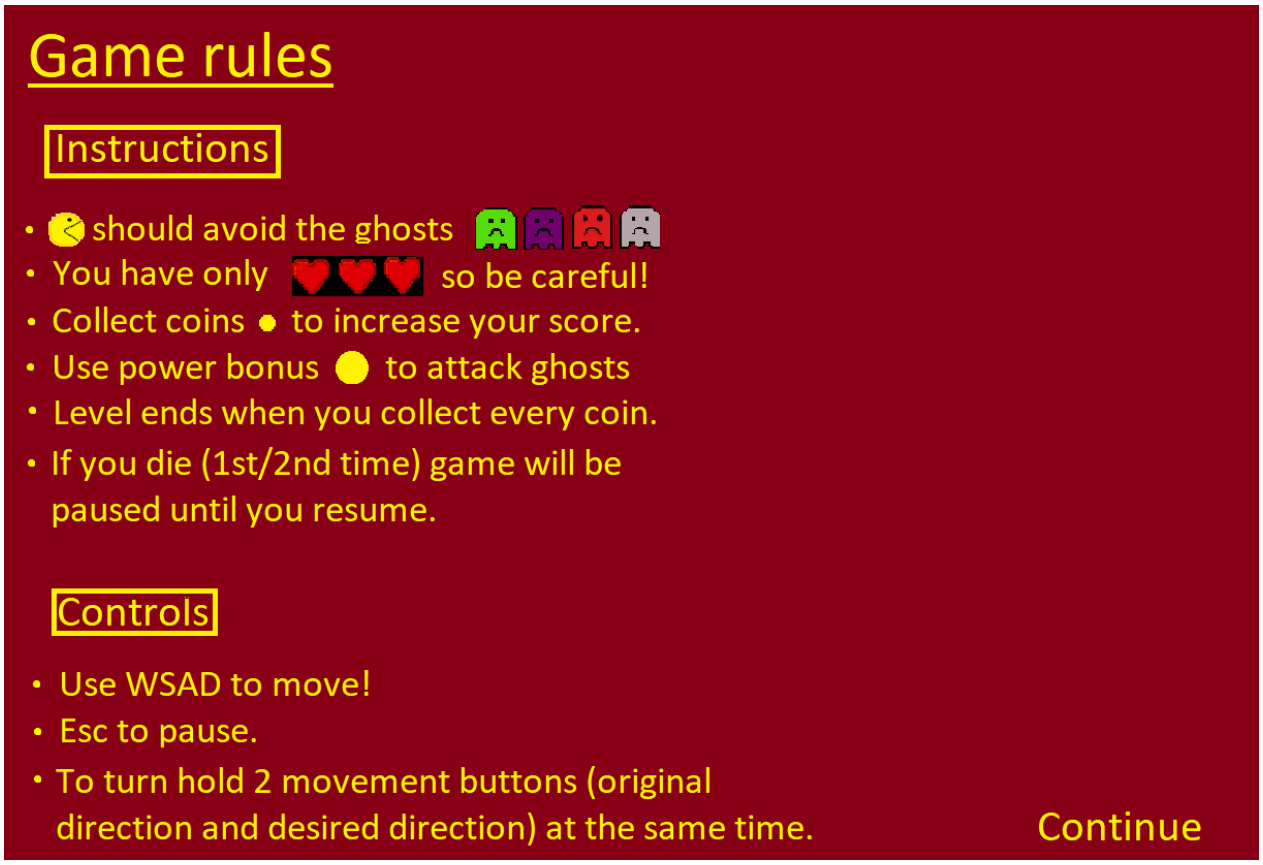
Obrázek 10- duch

Obrázek 11-ukazatel života

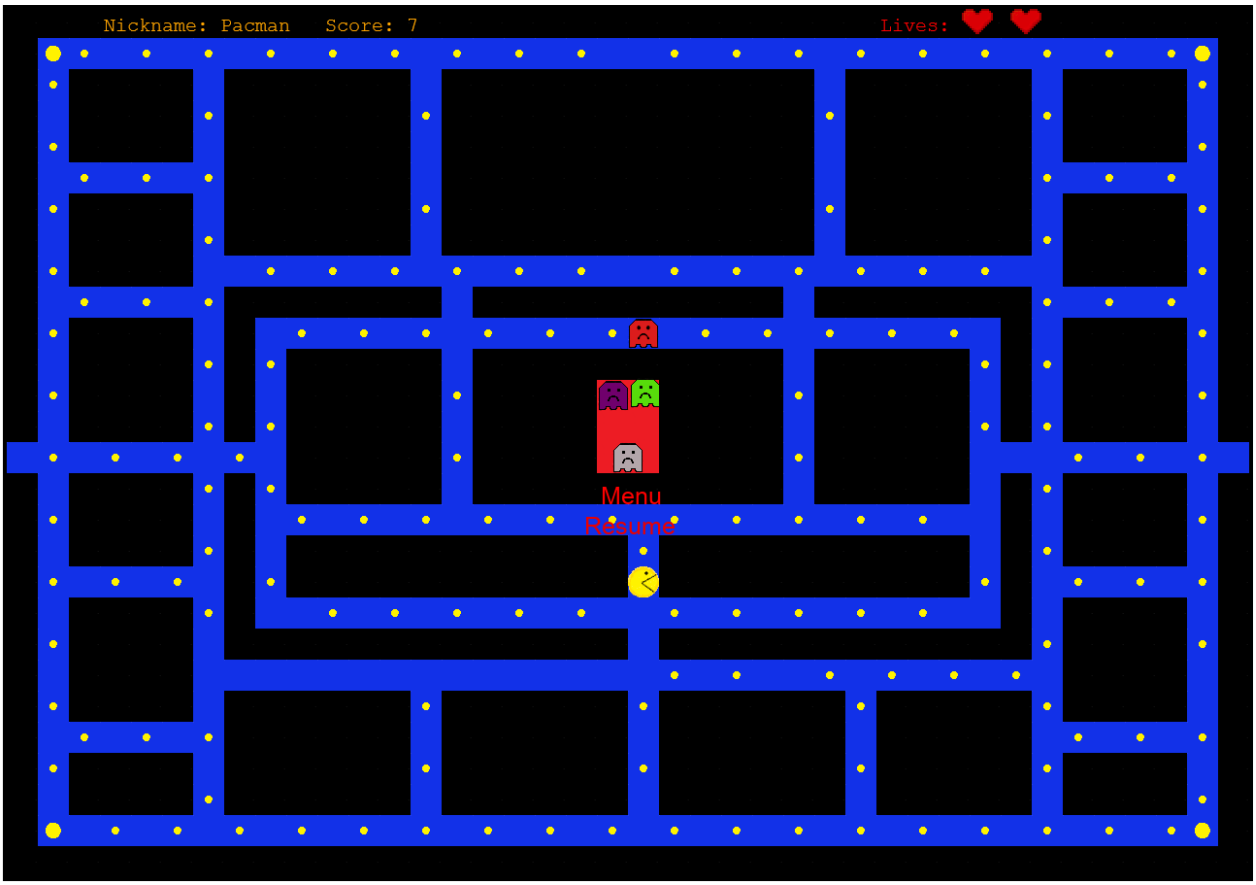
Obrázek 12-kostička

3.2) Úrovně

Jednotlivé úrovně byly vytvořeny v programu **Tiled**. Jedná se o mapový editor určený k vytváření map do her. V malování jsem vytvořil základní stavební bloky, dále jsem v grafickém prostředí editoru **Tiled** poskládal bludiště a tento program vygeneroval .**json** soubor, ve kterém jsou uloženy pozice jednotlivých bloků. **Phaser** následně s pomocí tohoto souboru a textur, které jsou uložené v *dist/assets/Tilesets*, vytvoří herní plochu.



Obrázek 13- ovládání a pravidla



Obrázek 14- 1. úroveň

4) Instalace a ovládání

Hra se spustí pomocí souboru Play.bat (ve složce src), který obsahuje příkazy k instalaci potřebných knihoven a následnému spuštění. V případě problémů uživatel může smazat soubor *deleteThisToReinstall* a při dalším spuštění dojde k přeinstalování. Následně se objeví webová stránka s oknem, kde se nachází menu, ve kterém se uživatel pohybuje pomocí tlačítek. Po zmáčknutí tlačítka Play se hráči spustí obrazovka s návodem a následně se může pustit do hraní.

Hráč ovládá postavu Pac-Man, která se objeví v bludišti. Pohybuje se pomocí kláves WSAD (pohyb nahoru, dolů, doleva, doprava). Hráč se může pohybovat po modrých kostičkách a pro zabočení musí držet tlačítko pro pohyb v původním směru a zároveň tlačítko pro pohyb ve změněném směru. Zelený, fialový a šedý duch se pohybují po dané dráze a v případě, kdy je hráč v jejich blízkosti, začnou ho pronásledovat. Červený duch se pohybuje nejkratší cestou směrem k hráči. Klávesou escapce lze hru pozastavit a následně obnovit či se vrátit do hlavní nabídky. Úroveň končí sesbíráním všech mincí, popřípadě ztrátou všech 3 životů (hráč ztrácí jeden život kolizí s duchy). Při sebrání bonusu (větší mince) se duchové zastaví a hráč je může porazit. Každý z duchů se po určitém čase obnoví. Skóre hráč dostává za porážení duchů a sbírání mincí.

Závěr

Zadání práce jsem splnil a naprogramoval hru Pac-Man. Hra funguje po vzoru Pac-Man z roku 1980. Nejedná se ovšem o přesnou kopii, ale pouze o adaptaci. Za účelem lepší hratelnosti se například Pac-Man pohybuje pouze při držení klávesy pro pohyb a na rozdíl od původní hry se může zastavit. Hra dále obsahuje žebříček nejlepších hráčů a celkem si hráč může zahrát 3 úrovně. Po stisknutí tlačítka Play se zobrazí ovládání a cíl hráče.

Grafická část práce je vytvořená v programu **GIMP** a malování. Projekt je založen na JavaScriptu a napsán v **HTML5** frameworku **Phaser 3**, který jsem používal minulý rok v rámci skupinové ročníkové práce ve 3. ročníku, tudíž pro mě byla práce s tímto frameworkem o něco jednodušší.

Co se týče možných vylepšení, dalo by se ještě přidat více úrovní, nabídnout hráči vyšší obtížnost (například různé vzorce pohybů duchů), či přidat zvukové efekty.

Nicméně s výsledkem projektu jsem spokojen. Přestože existuje mnoho verzí této hry myslím, že je má verze zábavná a přináší něco nového.

Zdroje

Framework Phaser 3 [online]. Dostupné z: https://phaser.io/phaser3

Phaser 3 API dokumentace [online]. Dostupné z: https://photonstorm.github.io/phaser3-docs/index.html

Ročníková práce 3. ročník Gymnázium Arabská Autoři: Průšek, Daňa, Mucha, Vašek z 30. 4. 2019 [online]. Dostupné z https://github.com/gyarab/Space-Pilots

Knihovna EasyStar [online]. Dostupné z: https://easystarjs.com/, https://github.com/prettymuchbryce/easystarjs

Tutoriál ke knihovně EasyStar v Phaseru 3, Autor: Renaux, J., z 6. 3. 2018 [online]. Dostupné z: https://www.dynetisgames.com/2018/03/06/pathfinding-easystar-phaser-3/

Davey, R., z 6. 2. 2015, Phaser Tutorial [online]. Dostupné z: https://phaser.io/tutorials/coding-tips-005

Aplikace Tiled [online]. Dostupné z: https://www.mapeditor.org/index.html

Phaser 3 Game Tutorial 9 Tiled, 2. 3. 2019, Youtube , kanál uživatele jest array [online]. Dostupné z: https://www.youtube.com/watch?v=2_x1dOvgF1E