



# **Desktopová hra v Unity - Snake Splix**

**VOJTĚCH SVOBODA**

**V4C**

**PROFILOVÁ ČÁST MATURITNÍ ZKOUŠKY  
MATURITNÍ PRÁCE**

**BRNO 2019**

## **ZADÁNÍ MATURITNÍ PRÁCE**

obhajované před zkušební komisí

Školní rok: 2018/2019

Předmět: Soubor odborných předmětů (zaměření na informační technologie)

Studijní obor: Informační technologie (18-20-M/01)

ŠVP: Informační technologie

Žák: **Vojtěch Svoboda**

Třída: **V4C**

Vedoucí práce: Ing. Petr Pernes

Odborný konzultant:

Termín zadání MP: 7. prosince 2018

Termín odevzdání MP: 16. dubna 2019

Číslo zadání, verze zadání a název práce:

### **PROG2b: desktopová hra u Unity – Snake Splix**

#### **Zadání**

Vytvořte desktopovou hru pro dva hráče v Unity. Hra bude ve stylu Snake Splix. Implementujte několik herních režimů, např. klasika, na body, na čas, překážky apod. Získané body (score) se jménem a bezpečně uložte, pokuste se zabránit neoprávněné manipulaci s daty. Ve hře užíjte animací pro zlepšení hratelnosti, popřípadě 3D.

#### **Výstupem práce bude**

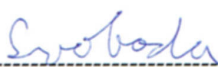
- Desktopová aplikace
- Postup instalace SW
- Uživatelská příručka
- Dokumentace a zdrojové kódy v elektronické podobě
- Dokumentace bude obsahovat záznam z testování aplikace (jednotkové testy)
- Dokumentace bude obsahovat odkazy a přístupové údaje k aplikaci, DB, Git-u, apod.
- Vývoj aplikace bude možné sledovat pomocí verzovacího systému

#### **Doporučené prostředky pro řešení, technické požadavky**


- Java, Git, C#, Python, Unity, Adobe Photoshop

Součástí zadání je Příloha zadání maturitní práce.

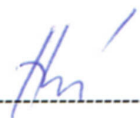
V Brně dne 7. 12. 2018



Podpis žáka



Podpis vedoucího práce



Podpis předsedy PK

# Prohlášení

Prohlašuji, že jsem maturitní práci na téma Desktopová hra v Unity - Snake Splex vypracoval samostatně a použil jen zdroje uvedené v seznamu literatury.

Prohlašuji, že:

- Prohlašuji, že беру на vědomí, že zpráva o řešení maturitní práce a základní dokumentace k aplikaci bude uložena v elektronické podobě na intranetu Střední průmyslové školy Brno, Purkyňova, příspěvkové organizace.
- Prohlašuji, že беру на vědomí, že bude má maturitní práce včetně zdrojových kódů uložena v knihovně SPŠ Brno, Purkyňova, p. o., dostupná k prezenčnímu nahlédnutí. Škola zajistí, že nebude pro nikoho možné pořizovat kopie jakékoliv části práce.
- Prohlašuji, že беру на vědomí, že SPŠ Brno, Purkyňova, p. o., má právo celou moji práci použít k výukovým účelům a po mém souhlasu nevýdělečně moji práci užít ke své vnitřní potřebě.
- Prohlašuji, že беру на vědomí, že pokud je součástí mojí práce jakýkoliv softwarový produkt, považují se za součást práce i zdrojové kódy, které jsou předmětem maturitní práce, případně soubory, ze kterých se práce skládá. Součástí práce není cizí ani vlastní software, který je pouze využíván za přesně definovaných podmínek, a není podstatou maturitní práce.

## Anotace

Tato maturitní práce se zabývá vytvořením počítačové hry pro 2 místní hráče. Hráči se snaží zabírat území a případně soupeřit s protihráčem v zabírání území. Hráči mají životy, které ztrácí při naražení do překážky nebo při kolizi s jiným hráčem. Hra obsahuje i režim pro jednoho hráče. Ve výsledné práci jsou zvukové efekty, které vylepšují celkový prožitek ze hry. Hra je naprogramována v C# a jako herní engine bylo použito Unity.

## Klíčová slova

*unity, hráč, herní engine, C#, zabraná oblast*

# Obsah

Prohlášení . . . . .	III
Anotace . . . . .	IV
Seznam použitých zkratek . . . . .	8
<b>1 Teoretický úvod</b>	<b>9</b>
1.1 Výběr tématu . . . . .	9
1.2 Cíl práce . . . . .	9
<b>2 Rozbor řešení</b>	<b>10</b>
2.1 Rozbor problémů . . . . .	10
2.1.1 Vyplnění pole . . . . .	10
2.1.2 Vytváření herních objektů . . . . .	10
2.1.3 Balance a zábava hry . . . . .	10
2.1.4 Zabíjení mezi hráči . . . . .	10
2.2 Použitá technologie . . . . .	10
2.2.1 Unity . . . . .	10
2.2.2 Bosca Ceoil . . . . .	11
<b>3 Assety</b>	<b>12</b>
3.1 Skripty . . . . .	12
3.1.1 AudioManager.cs . . . . .	12
3.1.2 GameManager.cs + GameManager2P.cs . . . . .	12
3.1.3 MainMenu.cs . . . . .	12
3.1.4 Node.cs . . . . .	13
3.1.5 Sound.cs . . . . .	13
3.2 Models . . . . .	13
3.2.1 Model hráče, mapa . . . . .	13
3.2.2 Slider . . . . .	14
3.3 Audio . . . . .	14
3.4 Fonts . . . . .	14
3.5 Scenes . . . . .	15
3.5.1 MainMenu.unity . . . . .	15
3.5.2 GameP1.unity + GameP2.unity . . . . .	15
<b>4 GameManager.cs</b>	<b>16</b>
4.1 Update() . . . . .	16
4.2 Pohyb hráče . . . . .	17
4.3 Problematika vybarvení hráčova pole . . . . .	17
4.4 Leaderboard . . . . .	19

4.5	Kdy hráč umře . . . . .	21
4.6	Ztracení životů . . . . .	22
<b>5</b>	<b>Závěr</b>	<b>24</b>
<b>6</b>	<b>Příloha - Uživatelská příručka</b>	<b>26</b>
<b>7</b>	<b>Úvod</b>	<b>27</b>
<b>8</b>	<b>Uživatelské rozhraní</b>	<b>27</b>
<b>9</b>	<b>Hra</b>	<b>28</b>
9.1	Hra jednoho hráče . . . . .	28
9.2	Hra pro dva hráče . . . . .	29
<b>10</b>	<b>Nastavení</b>	<b>30</b>

# Seznam obrázků

2.1	Prostředí programu Bosca Ceoil . . . . .	11
3.1	Slider Cube . . . . .	14
3.2	PressStart2P.ttf . . . . .	14
3.3	Main menu . . . . .	15
3.4	Hra pro 2 hráče . . . . .	15
4.1	Možnost hráče jak vybarvit pole . . . . .	18
8.1	Hlavní menu . . . . .	27
9.1	Hra pro jednoho hráče . . . . .	28
9.2	Hra pro dva hráče . . . . .	29
10.1	Nastavení . . . . .	30
10.2	Přizpůsobení . . . . .	31

# Seznam použitých zkratek

<b>C#</b>	C Sharp; programovací jazyk
<b>GameObject</b>	Objekt v herním prostředí, má atributy a svoji polohu v herním prostředí
<b>AudioMixer</b>	V audio mixeru se přidávají skupiny zvuků, kterým lze měnit hlasitost
<b>UI</b>	User Interface; uživatelské rozhraní
<b>Alpha</b>	Parametr průhlednosti barvy
<b>Vector3</b>	Odkazuje na přesné místo v herním prostředí; vypočítá se z x,y,z souřadnic
<b>Herní engine</b>	Sada komponent, které zjednodušují práci při výrobě počítačových her
<b>LaTeX</b>	Lamport TeX; program pro počítačovou sazbu; za pomoci něj jdou psát dokumenty
<b>GitHub</b>	Verzovací online systém
<b>Tick</b>	Jeden snímek v herním prostředí
<b>Ocas</b>	Cesta, kterou za sebou zanechává hráč; Pokud přes ni přejede oponent ztratí hráč život



# 1 Teoretický úvod

Celá maturitní práce je dostupná na: [github.com/vojtas131/Snake-Splix-Public](https://github.com/vojtas131/Snake-Splix-Public) a uživatelská aplikace je pod záložkou release.

## 1.1 Výběr tématu

Téma jsem si vybral, protože mě baví hry a baví mě je programovat. S herním enginem Unity už jsem věděl základy, takže jsem nemusel řešit orientaci v prostředí a učit se základy C#.

## 1.2 Cíl práce

Cílem práce je vytvořit hru, ve které se mohou zabavit 2 hráči a soutěžit spolu. Hra by měla být zábavná a správně vybalancovaná. Vedlejším cílem bylo vytvořit mód pro 1 hráče, kde hráč zápasí s časem a vyhýbá se překážkám.

## 2 Rozbor řešení

### 2.1 Rozbor problémů

#### 2.1.1 Vyplnění pole

Jako základ funkčnosti bylo navrhnout algoritmus, který by vyplňoval pole, které hráč zabere. Problém je, že hráč může obsadit pole různých tvarů a proto bylo potřeba, aby algoritmus pokryl všechny možnosti.

#### 2.1.2 Vytváření herních objektů

Bylo potřeba zachovat ve všem sounáležitost. Tedy mapa, hráčova oblast a hráči museli spolupracovat spolu a být na sobě nezávislé.

#### 2.1.3 Balance a zábava hry

Tento problém byl spíš po stránce game designu než kódu. Bylo potřeba vyřešit, aby zbývající životy měli určitý vliv na skóre.

#### 2.1.4 Zabíjení mezi hráči

Problém se skládal ze 2 částí. Kdy má hráč umřít a kdo umře pokud se hráči srazí.

### 2.2 Použitá technologie

#### 2.2.1 Unity

Jako herní engine jsem použil Unity. I když jsem měl na výběr i z jiných herních engineů (jako například Unreal engine nebo Cryengine), Unity mi přišlo nejlepší, protože jsem v něm už pracoval. Programovací jazyk pro scripty je v C#. Ačkoliv Unity podporuje i JavaScript, je už na ústupu a nedoporučuje se používat.

## 2.2.2 Bosca Ceoil

Potřeboval jsem program, ve kterém mohu jednoduše vytvořit hudbu. Bosca Ceoil splnil všechny tyto podmínky a je navíc zdarma.



Obr. 2.1: Prostředí programu Bosca Ceoil

## 3 Assets

### 3.1 Skripty

#### 3.1.1 AudioManager.cs

Skript slouží k ovládání audia skrz scény. Je dán na GameObject, který se přenáší skrz scény. Pracuje s atributy Sound.cs.

#### 3.1.2 GameManager.cs + GameManager2P.cs

Slouží k funkčnosti hry, je rozdělen na mód pro jednoho hráče a na mód pro dva hráče. Tento skript je pro jeho rozsáhlost vysvětlen v kapitole 4.GameManager na stránce 16.

#### 3.1.3 MainMenu.cs

Jsou v něm metody pro funkčnost a chod Hlavního menu. Skript dělím na regiony, ve kterých definuji funkce, pro různé části menu.

##### Customization

Neboli přizpůsobení herních prvků.

```
public void ApplyChanges(){
    warning.SetActive(false);
    if(Int32.Parse(inputObstacles.text)
        >=((Int32.Parse(inputHeight.text)-3)
        *(Int32.Parse(inputWidth.text)-3))-8){
        warning.SetActive(true);
        return;
    }
    StartCoroutine(ApplyChangesCoroutine());
}
```

Tato metoda se volá při stisku tlačítka Apply changes. Testuji aby počet překážek nebyl větší než maximální počet překážek na mapu.

Vzorec je  $((\text{výška celé mapy})-3)*((\text{šířka celé mapy})-3) - 8$  (což je počet startovních polí pro hráče). Poté se volá ApplyChangesCoroutine(), která obsahuje metody, abych vytvořil zpoždění.

```

public IEnumerator ApplyChangesCoroutine(){
    saved.GetComponent<Text>().color = new Color(0,0,0,255);
    saved.SetActive(true);
    ...
    saved.GetComponent<Text>().CrossFadeAlpha(0,1.5f,false);
    yield return new WaitForSeconds(1.5f);
    saved.SetActive(false);
    yield return null;
}

```

Pro vytvoření textu, který postupně mizí, jsem použil alpha parametr v barvě textu. Použil jsem metodu `CrossFadeAlpha`, která postupně mění alfu v barvě textu. Nastavil jsem dobu trvání na 1.5 s. Poté celá funkce čeká stejnou dobu a pak nastaví text na neviditelný.

### 3.1.4 Node.cs

Odkazuje na jedno pole na mapě. Je v něm poloha x, y, jeho `worldPosition` (místo v herním prostředí), komu políčko patří a jestli na něm právě je nebo není ocas hráče.

```

public class Node{
    public int x;
    public int y;
    public Vector3 worldPosition;
    public int land;
    public int tail;
}

```

Čerpá z něho:

- `HelpNode.cs` - Pomocná třída při barvení oblasti, kterou hráč zabral.
- `LandNode.cs` - Je v něm herní objekt, pole, a komu patří pole.
- `TailNode.cs` - Je v něm herní objekt a pole.

### 3.1.5 Sound.cs

Třída která určuje parametry zvuků ve hře. Má zdrojový zvuk, hlasitost, výšku, opakování a ke kterému `AudioMixeru` patří.

## 3.2 Models

### 3.2.1 Model hráče, mapa

Protože modely hráče a mapy dělám jenom z barvy, nebylo potřeba vytvořit modely pro tyto objekty v externím programu.

### 3.2.2 Slider

Pro posuvník hudby jsem potřeboval model. Vyrobil jsem tento jednoduchý obrázek 32x32 pixelů. Tento model jsem vytvořil v online programu Piskel.



Obr. 3.1: Slider Cube

## 3.3 Audio

Jako hudbu jsem se inspiroval 8-bitovou.

Vytvořil jsem zvuky pro:

- Smrt - Death.wav
- Výhru - Victory2.wav
- Vybarvení pole - Fill.wav
- Hudba do pozadí - MainMenuMusicTheme4.wav

## 3.4 Fonts

Jako font jsem použil font s nádechem 8-bitových her. Použil jsem font PressStart2P.ttf.

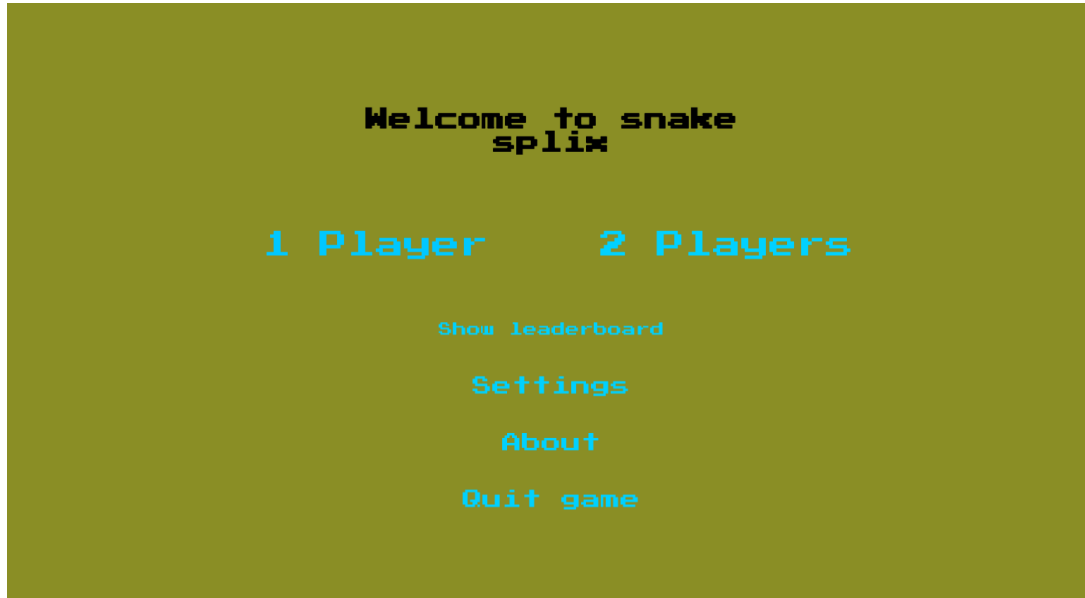
ABCDEFGHIJKLMNOPQRSTUVWXYZ  
NOPQRSTUVWXYZ

Obr. 3.2: PressStart2P.ttf

## 3.5 Scenes

### 3.5.1 MainMenu.unity

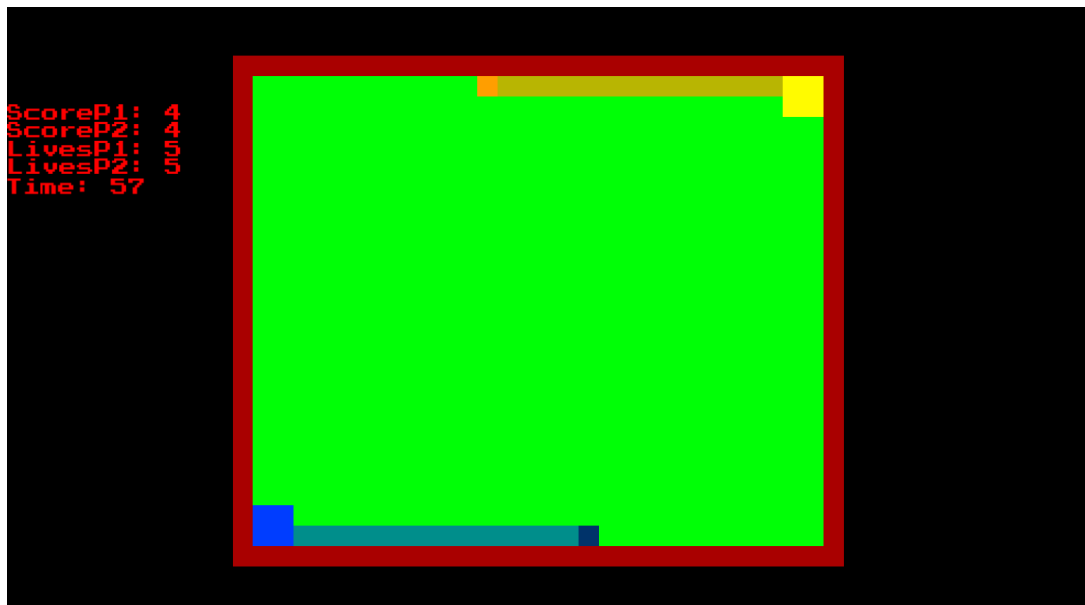
Jsou na ní tlačítka na zvolení herního módu, nastavení, o autorovi a na odejití ze hry.



Obr. 3.3: Main menu

### 3.5.2 GameP1.unity + GameP2.unity

Scéna ve které probíhá hra. Má hráče, mapu a UI prvky které ukazují životy, skóre a čas.



Obr. 3.4: Hra pro 2 hráče

## 4 GameManager.cs

### 4.1 Update()

Update metoda se volá každý tick.

```
void Update(){
    if(!gameover){
        if(Input.GetButtonDown("Esc")){
            pause ^= true;
            Cursor.visible ^= true;
        }
        if(!pause){
            pauseMenu.SetActive(false);
            GetInput();
            SetPlayerDirection();
            timer += Time.deltaTime;
            time -= Time.deltaTime;
            if (timer > moveRate){
                timer = 0;
                curDirectionP1 = targetDirectionP1;
                curDirectionP2 = targetDirectionP2;
                MovePlayer();
            }
            curTimeText.text = "Time: " + Mathf.Round(time).ToString();
            if(time<=0) ResetPlayer();
        }
        else if(pause){
            pauseMenu.SetActive(true);
        }
    }
}
```

Hra se zastaví při stisku klávesy Esc. "pause" změní svoji hodnotu a kurzor myši je viditelný.

Update metoda funguje při každém ticku (snímku). Volá se GetInput() která zjišťuje které klávesy byli stisknuty. V každém ticku se přenáší hodnota reálného času do proměnné "timer". Jakmile "timer" získá hodnotu větší než je nastaven v "moveRate", tak zavolá metodu na pohyb hráče a "timer" se vynuluje. SetPlayerDirection() nastaví směr každého hráče. Získávám dvě "curDirection" pokud se jedná o hru pro dva.



## 4.2 Pohyb hráče

Nejdříve se zjistí, kterým směrem hráč jede. Pak se nastaví hodnota x nebo y podle směru.

```
Node targetNodeP1 = GetNode(player1Node.x + P1x, player1Node.y + P1y);
Node targetNodeP2 = GetNode(player2Node.x + P2x, player2Node.y + P2y);
player1.transform.position = targetNodeP1.worldPosition;
player1Node = targetNodeP1;
if (player1PrevNode.land != 1) {
    tailP1.Add(CreateTailNodeP1(player1PrevNode.x, player1PrevNode.y));
    player1PrevNode.tail = 1;
}
```

Zjistí se následující pole, na kterém se hráč bude nacházet. Hráčova pozice se nastaví na následující pole pomocí ".transform.position". Jestliže hráčovo předchozí pole bylo jeho vlastní, ocas se nevytvoří. To stejné se provádí i pro druhého hráče.

```
if (targetNodeP1.land == 1 && tailP1.Count > 0) {
    int countTail = tailP1.Count;
    FindObjectOfType<AudioManager>().PlayOneShot("Fill");
    for (int i = 0; i < countTail; i++) {
        TailNode curr = tailP1[0];
        curr.node.tail=0;
        CreateLandNodeP1(curr.node.x, curr.node.y);
        RemoveTail(tailP1);
    }
    FillP1();
    UpdateScore();
}
```

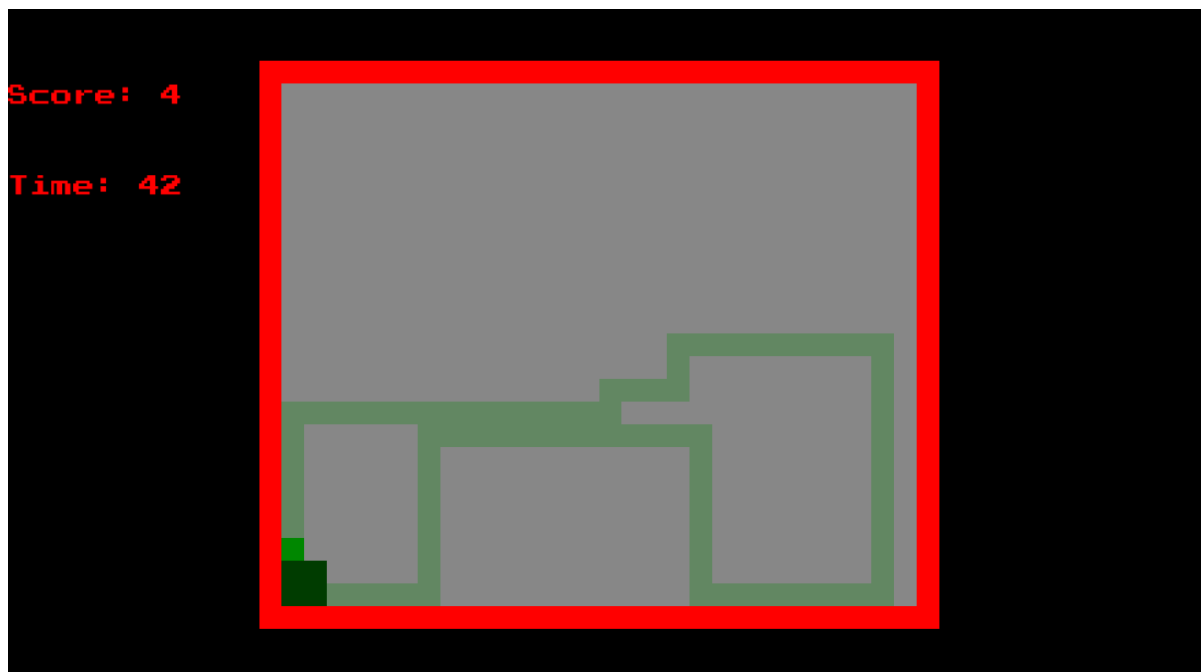
Pokud hráč skončí ve svém poli tak se začne vybarvovat jeho pole. Nejdříve vybarvím ocas a pak vybarvím oblast uvnitř.

## 4.3 Problematika vybarvení hráčova pole

Při řešení problému jak vybarvit část, kterou hráč zabral, jsem musel vytvořit vhodný algoritmus.

```
void FillRest(Node node){
    if(node.land==2) return;
    if(node.land==1) return;
    landP1.Add(CreateLandNode(node.x, node.y));
    FillRest(GetNode(node.x,node.y+1));
    FillRest(GetNode(node.x,node.y-1));
    FillRest(GetNode(node.x-1,node.y));
    FillRest(GetNode(node.x+1,node.y));
}
```

Algoritmus pro vybarvení jsem měl hned. Stačilo zvolit pole uvnitř oblasti a metoda vytvářela hráčova pole, dokud nenarazila na ocas nebo na hráčovo už zabrané pole. Při každém projití se metoda zavolá znovu se sousedními poly. Problém byl ten, že bylo zapotřebí zjistit jedno z polí, které se nachází uvnitř hráčovi oblasti. Další problém byl ten, že pokud by hráč vytvořil podobnou oblast jako je na Obr. 4.1, tak by bylo potřeba najít pole v obou oblastech.



Obr. 4.1: Možnost hráče jak vybarvit pole

V první verzi hry tedy hráč musel specificky ukončit oblast, aby se správně vybarvila.

Použil jsem tedy jiný způsob na vybarvení pole a to inverzním způsobem. Vytvořil jsem novou mapu, která se tentokrát skládala z HelpNode. HelpNode třída má Node a pomocný parametr own (ownership = vlastnictví).

```
void HelpFill(HelpNode node){
    if (node == null) return;
    if (node.node.land == 1) {
        node.own = 1;
        return;
    }
    if (node.own == -1) return;
    node.own = -1;
    HelpFill(GetHelpNode(node.node.x, node.node.y + 1));
    HelpFill(GetHelpNode(node.node.x, node.node.y - 1));
    HelpFill(GetHelpNode(node.node.x - 1, node.node.y));
    HelpFill(GetHelpNode(node.node.x + 1, node.node.y));
}
```

Procházel jsem novou mapu stejným způsobem, jako když jsem vytvářel vytvářel hráčovo pole, ale tentokrát místo vytvoření pole, jsem změnil parametr own na -1 (mimo), nebo na 1 pokud narazil na hráčovo území.

```
void FillRest() {
    for (int i = 0; i < maxWidth; i++) {
        for (int j = 0; j < maxHeight; j++) {
            if ((GetHelpNode(i, j)).own == 0) {
                CreateLandNode(i, j);
            }
        }
    }
}
```

Pak stačí projít mapou a na všech pozicích, kde se parametr nezměnil, vytvořit hráčovo pole. Metoda CreateLandNode() je ještě opatřena testem, jestli není hráčovo pole již na tomto poli vytvořeno (aby nevznikli duplikáty).

## 4.4 Leaderboard

Leaderboard je žebříček nejlepších hráčů. Hráč musí získat větší skóre než nejhorší hráč, aby se dostal na žebříček.

```
string input = inputName.text;
if(input.Contains(",") || input.Contains(";")) return;
string scores = PlayerPrefs.GetString("highscores", "");
string[] scoresAndNames = new string[2];
scoresAndNames = scores.Split(';');
string[] scoresList = new string[11];
scoresList = scoresAndNames[0].Split(',');
string[] namesList = new string[11];
namesList = scoresAndNames[1].Split(',');
```

Highscores jsou uloženy jako řetězec v registrech. Pokud neexistuje vytvoří se s hodnotou ";". Načítá se pod klíčem "highscores". V první části jsou uloženy skóre, v druhé jména (odděleno ";" a hodnoty v řetězci ","). Pokud by jméno obsahovalo jeden z těchto znaků, tak se vrátí a nenechá hráče vytvořit takové jméno.

```

for(int i=0;i<scoresList.Length;i++){
    bool emptyField = Int32.TryParse(scoresList[i],out int x);
    if(!emptyField){
        namesList[i]=input;
        scoresList[i]=score.ToString();
        break;
    }
    if(score>=x){
        for(int n=scoresList.Length-1;n>i;n--){
            scoresList[n]=scoresList[n-1];
            namesList[n]=namesList[n-1];
        }
        scoresList[i]=score.ToString();
        namesList[i]=input;
        break;
    }
}
}

```

Převádím skóre ze stringu na int. Pokud nejde převést (pole obsahuje "") metoda TryParse() vyhodí false. Nejvyšší skóre je uloženo na začátku. Jakmile je hráčovo aktuální skóre větší než uložená hodnota, tak se všechny hodnoty posunou o 1 dolů.

```

string listOut = "";
int length;
if(scoresList.Length>=10)    length = 10;
else length = scoresList.Length;
for(int i=0;i<length;i++){
    listOut += scoresList[i]+",";
}
listOut += ";";
for(int i=0;i<length;i++){
    listOut += namesList[i]+",";
}
PlayerPrefs.SetString("highscores",listOut);

```

Následně převedu všechny hodnoty zpět do jednoho stringu ve stejném formátu jako jsem string načítal. Uložím do registru pod klíčem "highscores".

## 4.5 Kdy hráč umře

Aby byla hra více zábavná ve dvou hráčích, přidal jsem možnost zabíjení se mezi sebou. S tím však přišly problémy typu, kdo má umřít pokud se hráči srazí přímo.

```
if(player1Node == player2Node){
    if(player2Node.land==3){
        killP1=true;
    }
    else if(player1Node.land==1){
        killP2=true;
    }
    else{
        killP1=true;
        killP2=true;
    }
}
```

Pokud se hráči budou nacházet oba na stejném poli, tak by měli oba umřít. Dal jsem ale tomu hráči, který se bude nacházet uvnitř svého pole, výhodu. Díky tomu přežije ten hráč, který je uvnitř své vybarvené oblasti, pokud se oba hráči srazí. Pokud se oba hráči srazí mimo své pole, tak umřou oba.

```
if(player2Node.tail == 1)    killP1=true;
if(player1Node.tail == 2)    killP2=true;

if(player1Node.tail == 1 || player1Node.land == -1
|| player1Node.land == 2 || player1Node.land == 4){
    scoreP1-=5;
    ResetPlayer(false);
}

if(player2Node.tail == 2 || player2Node.land == -1
|| player2Node.land == 2 || player2Node.land == 4){
    scoreP2-=5;
    ResetPlayer(true);
}
```

V první části kódu zjišťuji, jestli se některý z hráčů nenachází na ocasu druhého hráče. Pokud tomu tak je, tak druhý hráč zemře. V druhé části kódu zjišťuji, jestli se hráč nezabil sám. K tomu může dojít, pokud hráč trefí překážku, nebo když trefí svůj vlastní ocas. Ještě ke všemu ztratí 5 bodů ze skóre. Pokud umřel druhý hráč díky prvnímu hráči, přičte se mu ke skóre 5 bodů.

## 4.6 Ztracení životů

Pro 2 hráče jsou 2 metody, jedna při úmrtí a druhá při skončení hry. V případě jednoho hráče stačila jedna metoda na prohru, protože hráč má jen jeden život.

```
private void ResetPlayer(bool player){
    if(!player){
        livesP1--;
        int countTailP1 = tailP1.Count;
        for(int i=0;i<countTailP1;i++){
            tailP1[0].node.tail=0;
            RemoveTail(tailP1);
        }
        int listInt = rnd.Next(0,landP1.Count-1);
        player1Node = landP1[listInt].node;
        player1.transform.position = player1Node.worldPosition;
        Node player1NodeRight = GetNode(player1Node.x+1,player1Node.y);
        if(player1NodeRight.land== -1
            || player1NodeRight.land==2
            || player1NodeRight.land==4){
            targetDirectionP1 = Direction.left;
        }
        else targetDirectionP1=Direction.right;
    }
}
```

Proměnná "player" říká, který z hráčů umřel (false = Hráč 1;true = Hráč 2). Hráči se odečtou životy a poté se na každé pole, kde měl hráč ocas, zavolá metoda RemoveTail(), kde se odebere část ocasu a změní v poli hodnotu "tail" na 0. Následně se vybere jedno náhodné pole, které hráč vlastní a oživí ho zde. Jestli by bylo pole hned vedle překážky, bude směr hráčova pohybu doleva.

```
void GameOver(){
    FindObjectOfType<AudioManager>().Play("PlayerWin");
    finalScoreP1+=livesP1*10;
    finalScoreP2+=livesP2*10;
    gameoverMenu.SetActive(true);
    if(finalScoreP1==finalScoreP2){
        gameoverMessage.text = "TIE";
    }
    else if(finalScoreP1<finalScoreP2){
        gameoverMessage.text ="P2 WIN";
        matchesWonP2++;
    }
    else if(finalScoreP1>finalScoreP2){
        gameoverMessage.text ="P1 WIN";
        matchesWonP1++;
    }
    matchesScoreP1.text=matchesWonP1.ToString();
    matchesScoreP2.text=matchesWonP2.ToString();
}
```

Metoda `GameOver()` se volá pokud jednomu z hráčů dojdou životy, nebo vyprší čas. Přehraje se zvuk vítězství, pak se převedou zbývající hráčovi životy na skóre. Následně se porovnají skóre a vítězovi se přidá vyhraný zápas. Pokud hráči hrají víckrát, pak se číslo vyhraných zápasů zvětšuje. V případě remízy se nepřičte vyhraný zápas ani jednomu.

## 5 Závěr

V maturitní práci se mi bohužel nepovedlo začlenit animace, protože jsem nevěděl, jak je zakomponovat do takto graficky jednoduché hry. Zabezpečení highscore je pouze v registru a tak se k tomu někteří uživatelé můžou dostat. Ostatní požadavky zadání jsem splnil.

Díky maturitní práci jsem si vylepšil znalosti s herním enginem unity a jazyka C#. Pokud bych maturitní práci dělal teď, napsal bych kód určitě jinak (například bych raději udělal jeden skript pro mód 1 hráče i pro 2). Bohužel na kompletní přepsání nemám dostatek času.

Pokud bych se k této maturitní práci v budoucnu vrátil, určitě bych dodělal lepší zabezpečení dat a přidal různé drobnosti. Mezi drobnosti bych uvedl například herní měnu, různé hráčovi barvy, levely a s nimi zvětšující se obtížnost, nepřátelé. I přes tyto drobnosti dopadla maturitní práce podle mých představ a jsem s výsledkem spokojený.

Výslednou dokumentaci jsem vytvořil v LaTeXu a kód je dostupný na githubu ve složce Documentation.



# Reference

- [1] Press Start 2P Font | dafont.com. DaFont - Download fonts [online]. Dostupné z: <https://www.dafont.com/press-start-2p.font>
- [2] Unity - Manual: Unity User Manual (2018.3). [online]. Copyright © 2019 Unity Technologies. Publication [cit. 06.04.2019]. Dostupné z: <https://docs.unity3d.com/Manual/index.html>
- [3] Průvodce jazykem C# | Microsoft Docs. [online]. Dostupné z: <https://docs.microsoft.com/cs-cz/dotnet/csharp/>
- [4] Bosca ceoil. Home [online]. Dostupné z: <https://boscaceoil.net/>
- [5] Piskel - Free online sprite editor. Piskel - Free online sprite editor [online]. Dostupné z: <https://www.piskelapp.com/>
- [6] LaTeX - A document preparation system. LaTeX - A document preparation system [online]. Dostupné z: <https://www.latex-project.org/>

## **6 Příloha - Uživatelská příručka**

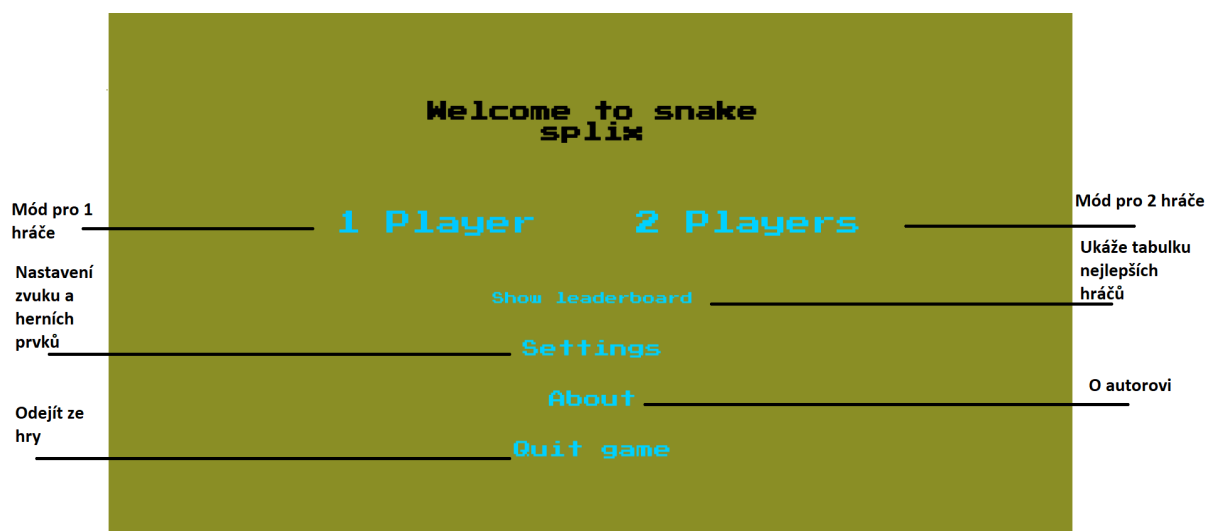
### **UŽIVATELSKÁ PŘÍRUČKA**

## 7 Úvod

Snake Splex je 2D hra pro jednoho nebo dva hráče. Jako čtvereček jedné určité barvy se budete snažit zabírat území, vyhýbat se překážkám a případně vyhrát nad protihráčem. Pokud se vám bude dařit, můžete své skóre uložit do seznamu nejlepších hráčů. Jakmile se začnete nudit, můžete změnit velikost mapy, rychlost hry a jiné herní prvky.

## 8 Uživatelské rozhraní

Při zapnutí hry vás přivítá toto menu.



Obr. 8.1: Hlavní menu

Pokud chcete hrát sami stiskněte 1 Player (1 hráč) a pokud chcete hrát ve 2 stiskněte 2 Players (2 hráči).

Pokud se chcete podívat na seznam nejlepších hráčů, můžete tak udělat kliknutím na tlačítko Show leaderboard (Ukázat žebříček).

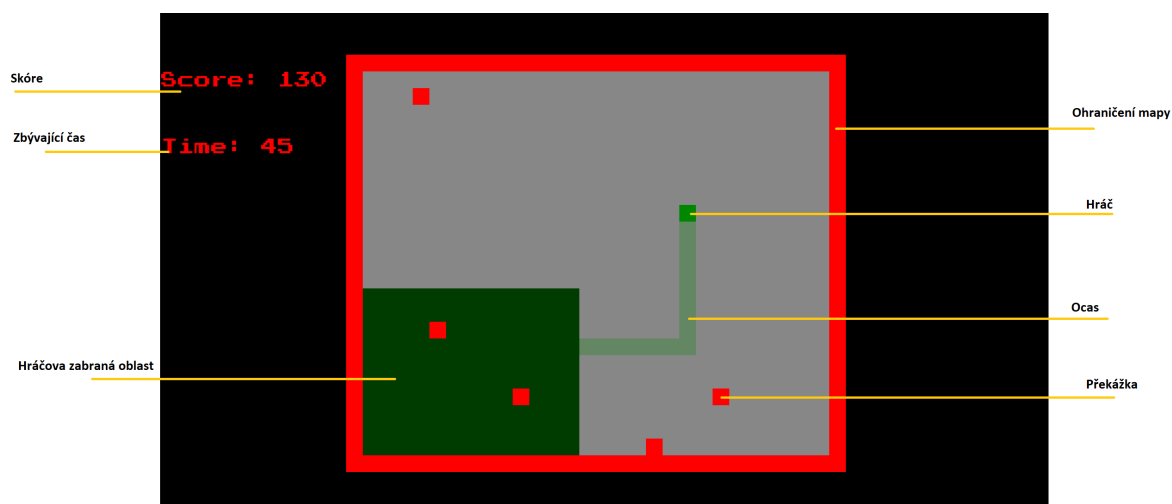
Pokud je hlasitost hudby pro vás příliš hlasitá nebo chcete změnit jeden z herních prvků jako jsou životy, můžete tak udělat po stisku tlačítka Settings (Nastavení).

Pro odejití ze hry slouží tlačítko Quit game (Odejít ze hry).

# 9 Hra

## 9.1 Hra jednoho hráče

Tato obrazovka se načte po stisku tlačítka 1 Player (1 hráč).



Obr. 9.1: Hra pro jednoho hráče

Pohyb hráče je pomocí kláves WASD.

- W = Pohyb nahoru
- A = Pohyb doleva
- S = Pohyb dolů
- D = Pohyb doprava

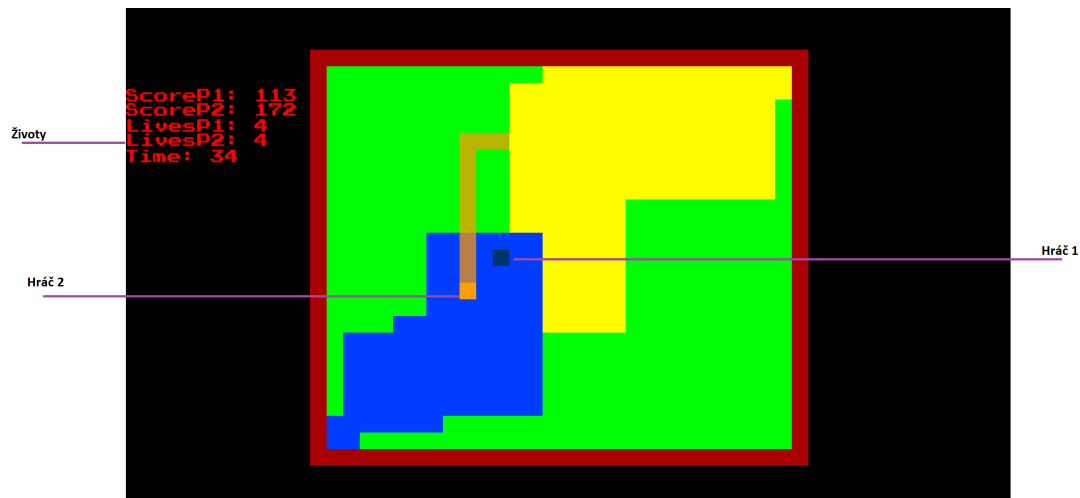
Pokud hráč dojde zpět do své zabrané oblasti, zabere se pole mezi ocasem a hráčovou oblastí.

Snažíte se vyhnout překážkám a ani nenarazit do ohraničení mapy.

Vlevo nahoře jde vidět zbývajcí čas a vaše momentální skóre.

## 9.2 Hra pro dva hráče

Tato obrazovka se načte po stisku tlačítka 2 Player (2 hráči).



Obr. 9.2: Hra pro dva hráče

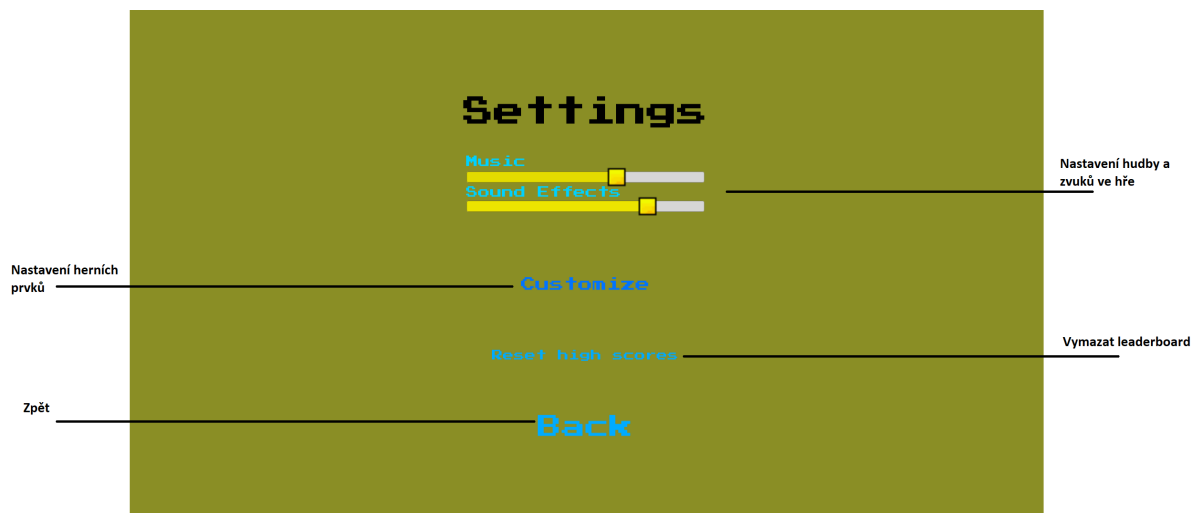
Pokud hrajete hru pro dva hráče druhý hráč se ovládá klávesy IJKL.

- I = Pohyb nahoru
- J = Pohyb doleva
- K = Pohyb dolů
- L = Pohyb doleva

V tomto režimu soupeříte s protihráčem. Každý hráč se snaží zabrat co nejvíce polí. Protihráče můžete zabít, když mu přejedete přes ocas, ale jenom pokud je mimo své pole. Hra končí pokud jednomu z hráčů dojdou životy, nebo vyprší časový limit. Vyhraje ten hráč, který má větší skóre.

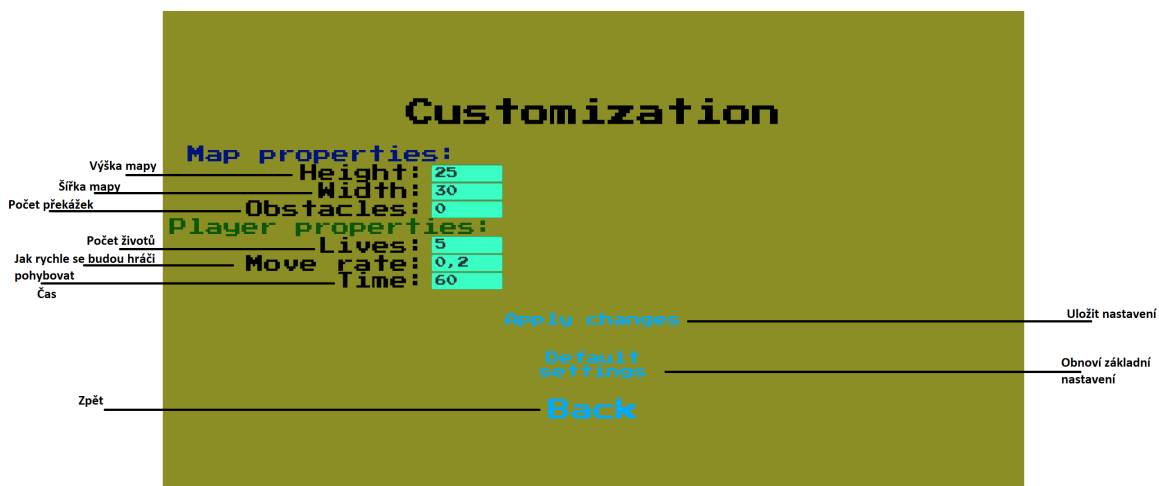
## 10 Nastavení

Tato obrazovka se načte po stisku tlačítka Settings (Nastavení).



Obr. 10.1: Nastavení

Na této obrazovce lze nastavit věci související s hrou. Můžete ztlumit zvuky, vynulovat žebříček nebo pomocí tlačítka customize se přesunout na novou obrazovku, kde lze nastavit různé herní prvky.



Obr. 10.2: Přizpůsobení

Na této obrazovce se mění herní prvky. Lze tu změnit velikost mapy, počet překážek, životy (pro mód 2 hráčů), move rate (jak rychle se budou hráči pohybovat) a čas. Pokud chcete změny uložit, klikněte na tlačítko apply changes. Pokud chcete obnovit základní nastavení, můžete tak udělat pomocí tlačítka default settings. Tlačítko back vás vrátí na předchozí obrazovku.