

Dokumentace projektu *The Rustwood Outlaw*

Jakub Endlicher

3. září 2025

Obsah

1	Úvod	2
2	Přehled projektu	2
3	Architektura a hlavní třídy	2
3.1	Třída Board (Form1.cs)	2
3.2	Třída Barricade (Barricades.cs)	3
3.3	Třída SpawnArea (Barricades.cs)	3
3.4	Třída Entity (Entity.cs)	4
3.5	Třída Player (Entity.cs)	4
3.6	Třída Enemy (Entity.cs)	5
3.7	Třída Bullet (entity.cs)	8
3.8	Třída Item (Item.cs)	8
3.9	Třída Level	8
4	Grafika a vykreslování	9
5	Načítání úrovní a práce s mapou	10
6	Nastavení hry (Settings.cs)	12
7	Uživatelské rozhraní	12
8	Zdroje a assety (Properties/Resources.resx)	14
9	Interakce uživatele	14
10	Možnosti rozšíření	14
11	Závěr	14

1 Úvod

Toto je podrobná programátorská dokumentace popisující projekt *The Rustwood Outlaw*, což je 2D akční hra vytvořená v jazyce C#. Pro uživatelskou dokumentaci si přečtěte README.md na hlavní github stránce projektu. Hra je určena pro platformu Windows Forms (.NET Framework 4.7.2). Dokumentace je zpracována dle doporučení <https://ksvi.mff.cuni.cz/~kryl/dokumentace.htm> a obsahuje podrobný popis tříd, funkcionality, algoritmů, interakce uživatele a možností rozšíření.

2 Přehled projektu

Projekt je rozdělen do několika hlavních částí:

- Herní logika a smyčka (`Form1.cs`, třída `Board`)
- Herní objekty (barikády, spawnovací oblasti, entity, předměty)
- Nastavení hry (`Settings.cs`)
- Grafické zdroje (`Properties/Resources.resx`)
- Uživatelské rozhraní (Designer soubory, panely, tlačítka)

3 Architektura a hlavní třídy

3.1 Třída Board (`Form1.cs`)

Účel: Hlavní herní plocha, která spravuje všechny objekty, zajišťuje herní smyčku, načítání úrovní, správu kolizí a uživatelské rozhraní.

Hlavní atributy:

- `List<Barricade> barricades` – seznam barikád na mapě
- `List<SpawnArea> spawnAreas` – seznam spawnovacích oblastí
- `List<Entity> entities` – seznam všech entit (hráč, nepřátelé)
- `List<Item> items` – seznam předmětů na mapě
- `Player player` – instance hráče
- `int level, score` – aktuální úroveň a skóre
- `Timer gameTime` – časovač pro herní smyčku
- `float deltaTime` – čas od posledního snímku
- UI prvky – panely, tlačítka, progress bary, atd.

Hlavní metody:

- `Startgame()` – inicializace nové hry

- `GameLoop()` – hlavní herní smyčka, volaná časovačem
- `levelTimer()` – správa časového limitu úrovně
- `LoadLevels()` – načtení úrovní ze zdrojů
- `LoadMap()` – načtení mapy a rozmístění objektů dle úrovně
- `DrawHearts()` – vykreslení životů hráče
- `YouLost()` – zpracování konce hry
- `Form1_KeyDown/KeyUp()` – zpracování vstupu z klávesnice

Popis herní smyčky:

1. Aktualizace všech entit (pohyb, kolize, útoky)
2. Spawnování nepřátel ve spawnovacích oblastech
3. Aktualizace předmětů (čas na zemi, sbírání)
4. Aktualizace UI (životy, skóre, pozadí)

3.2 Třída Barricade (Barricades.cs)

Účel: Reprezentuje barikádu na mapě, která blokuje pohyb entit.

Atributy:

- `PictureBox sprite` – grafická reprezentace
- `Point position, gridposition` – pozice v pixelech a na mřížce
- `Size size` – rozměr
- `Board board` – odkaz na herní plochu

Metody:

- `Destroy()` – odstranění barikády z plochy a paměti
- `Bounds` – obdélník pro kolize

3.3 Třída SpawnArea (Barricades.cs)

Účel: Dědí z `Barricade`, slouží jako místo pro spawnování nepřátel.

Atributy:

- `float spawnRate` – interval spawnování
- `float timeSinceLastSpawn` – čas od posledního spawnu
- `static Random random` – generátor náhodných čísel

Metody:

- `TryToSpawnEnemy(float deltatime)` – pokusí se spawnout nepřítele
- `Destroy()` – odstranění ze seznamu spawnovacích oblastí

3.4 Třída Entity (Entity.cs)

Účel: Základní třída pro všechny pohyblivé objekty (hráč, nepřátelé).

Atributy:

- `int speed, health, damage`
- `PictureBox sprite`
- `Point position`
- `Board board`
- `bool IsDestroyed` – flag označující objekty k odstranění
- `List<Barricade> obstacles`
- `List<Entity> entities`

Metody:

- `Update(float deltaTime)` – aktualizace stavu entity
- `Destroy()` – odstranění entity
- `CollidesAt(int x, int y)` – detekce kolize
- `Move(float deltaTime)` – pohyb entity
- `Draw()` – vykreslení entity
- `Bounds` – hitbox entity

3.5 Třída Player (Entity.cs)

Účel: Dědí z Entity, reprezentuje hráče.

Atributy:

- `Bitmap[] framesUp, framesDown` – pole s jednotlivými kroky animace
- `int currentFrame` – aktuální snímek animace
- `float animationTimer, animationSpeed` – animace
- `bool facingDown` – flag pro animace
- `HashSet<Keys> pressedKeys` – stisknuté klávesy
- `float shootCooldown, shootDelay` – časy střelení
- `float speedBoostTimer, fireRateTimer, damageRateTimer, multishotTimer` – časovače pro vylepšení
- `bool multishot` – flag pro multihot vylepšení

Metody:

- `Update(float deltaTime)` – pohyb, střelba, animace
- `Shoot(float deltaTime)` – logika střelby
- `UpdateItems(float deltaTime)` – efekty předmětů
- `Animate(float deltaTime)` – animace postavy

3.6 Třída `Enemy` (`Entity.cs`)

Třída `Enemy` je potomkem třídy `Entity` a reprezentuje nepřítele ve hře. Kromě základních vlastností entity (pohyb, zdraví, poškození, kolize) implementuje algoritmus pro pathfinding k hráči.

Atributy:

- `static Dictionary<(Point, Point), List<Point>> pathCache` – cache již nalezených cest mezi dvojicemi bodů (start, cíl) pro optimalizaci výpočtu.
- `Point lastStart, lastGoal` – poslední startovní a cílový bod použité cesty.
- `List<Point> lastPath` – poslední nalezená cesta.
- `int lastPathIndex` – index aktuálního kroku na poslední cestě.

Metody:

- `Pathfinding` – Hledání nejbližší cesty k hráči pomocí BFS
- `Extend` – Jeden krok BFS

Detailní popis metody `PathFinding`

1. **Zaokrouhlení pozic na mřížku:** Pozice nepřítele i cíle (hráče) jsou převedeny z pixelů na souřadnice mřížky (`RoundPointToGrid`).
2. **Využití cache:** Pokud je cesta mezi těmito body již vypočtena a uložena v `pathCache`, použije se pro další krok, což výrazně zrychluje pohyb více nepřátel.
3. **Bidirekcionální BFS:**
 - Dvě fronty (`queueStart, queueGoal`) a dvě množiny navštívených bodů (`visitedStart, visitedGoal`) rozšiřují průzkum současně od startu i cíle.
 - V každém kroku se expanduje jeden uzel z každé fronty pomocí metody `Expand`, která prochází všech 8 sousedních polí (včetně diagonál).
 - Pokud se průzkumy setkají (`meetPoint`), je nalezena průchozí cesta.
4. **Rekonstrukce cesty:**
 - Pomocí slovníků `prevFromStart` a `prevFromGoal` se zpětně sestaví celá cesta od startu k cíli přes průsečík.
 - Cesta je uložena do cache.
5. **Výstup:** Metoda vrací dvojici (`dx, dy`) určující směr dalšího kroku (např. (1,0) vpravo, (0,-1) nahoru, (1,1) diagonálně vpravo dolů).
6. **Pokud cesta neexistuje:** Vrací (0,0) – nepřítel zůstává stát.

Klíčové pomocné metody

- `Expand(...)` – provádí jeden krok BFS z daného směru, kontroluje kolize a hledá průnik s druhým průzkumem.
- `CollidesAt(x, y)` – ověřuje, zda je možné vstoupit na dané pole (není zde překážka, barikáda, jiná entita).
- `GetMaxPosition(dx, dy)` – provádí pohyb po ose X a Y po jednotlivých pixelech, dokud nenarazí na překážku.
- `RoundPointToGrid(Point coords)` – převádí pixelové souřadnice na souřadnice mřížky.

Příklad implementace metod PathFinding a Expand

```
1 private (int, int) PathFinding(Point goal)
2 {
3     Point gridStart = RoundPointToGrid(position);
4     Point gridGoal = RoundPointToGrid(goal);
5
6     if (cestaJeStejna())
7     {
8         return (nextStep);
9     }
10
11     var cacheKey = (gridStart, gridGoal);
12     if (cestaJizBylaNalezena)
13     {
14         return (nalezenaCesta);
15     }
16
17     var prevFromStart = new Dictionary<Point, Point>();
18     ...
19     visitedGoal.Add(gridGoal);
20
21     Point? meetPoint = null;
22
23     while (queueStart.Count > 0 && queueGoal.Count > 0)
24     {
25         if (Expand(queueStart, visitedStart, visitedGoal,
26             prevFromStart, prevFromGoal, out meetPoint)) break;
27         if (Expand(queueGoal, visitedGoal, visitedStart,
28             prevFromGoal, prevFromStart, out meetPoint)) break;
29     }
30
31     if (meetPoint != null)
32     {
33         zrekonstruujCestu();
34         return firstStep;
35     }
36 }
```

```

35     lastPath = null;
36     return (0, 0);
37 }

```

```

1 private bool Expand(Queue<Point> queue, ..., out Point? meetPoint
2 )
3 {
4     meetPoint = null;
5     if (queue.Count == 0) return false;
6     Point current = queue.Dequeue();
7
8     int[] dxx = { 0, 0, -1, 1, -1, -1, 1, 1 };
9     int[] dyy = { -1, 1, 0, 0, -1, 1, -1, 1 };
10
11     for (int dir = 0; dir < 8; dir++)
12     {
13         int nx = current.X + dxx[dir];
14         int ny = current.Y + dyy[dir];
15         Point next = new Point(nx, ny);
16         if (CollidesAt(nx, ny)) continue;
17         if (!visitedThis.Contains(next))
18         {
19             visitedThis.Add(next);
20             prevThis[next] = current;
21             queue.Enqueue(next);
22             if (visitedOther.Contains(next))
23             {
24                 meetPoint = next;
25                 return true;
26             }
27         }
28     }
29     return false;
30 }

```

Optimalizace a vlastnosti

- **Cache cest:** Výrazně snižuje výpočetní náročnost při pohybu více nepřátel ke stejnému cíli.
- **Bidirekcionální BFS:** Oproti klasickému BFS je rychlejší, protože průzkum probíhá z obou stran a setkání v polovině cesty je rychlejší.

Další metody třídy Enemy

- `Move(float deltaTime)` – volá `PathFinding`, provádí pohyb, kontroluje kolizi s hráčem.
- `TryDropItem()` – s určitou pravděpodobností upustí předmět po smrti.
- `Animate(float deltaTime)` – zajišťuje animaci pohybu.

3.7 Třída Bullet (entity.cs)

Účel: Reprezentuje vystřelenou kulku.

Atributy:

- float fx, fy – Aktuální pozice kulky
- float fdx, fdy – Vektor rychlosti kulky

Metody:

- Move() – pohyb kulky po mapě a řešení její kolize s číkoliv

3.8 Třída Item (Item.cs)

Účel: Reprezentuje předmět na mapě, který může hráč sebrat.

Atributy:

- ItemType Type – druh předmětu
- int Value
- Point Position
- Board Board
- PictureBox sprite
- float onGroundTimer

Metody:

- Draw() – vykreslení předmětu
- Destroy() – odstranění předmětu
- Update(float deltaTime) – čas na zemi, sbírání
- GetPickedUp() – logika při sebrání
- Bounds – hitbox předmětu

3.9 Třída Level

Účel: Uchovává informace o úrovni.

Atributy:

- string Name – Název úrovně
- int Time – Čas na úroveň
- float SpawnRate – Rychlost spawnování nepřátel
- int BossCount – Počet bossů na každé úrovni (bossové bohužel zatím nejsou implementováni)
- List<string> MapLines – Mapa jako taková

4 Grafika a vykreslování

Celá grafika je realizována pomocí knihovny Windows Forms, konkrétně prostřednictvím ovládacích prvků `PictureBox`. Každý herní objekt, který má být vizuálně zobrazen (hráč, nepřítel, barikáda, předmět atd.), je reprezentován vlastním `PictureBoxem`, který je přidán do příslušné kolekce `Entity`, `Barricades`, atd..

Princip vykreslování

- **Inicializace:** Při vytvoření každého objektu (např. v konstruktorech tříd `Player`, `Enemy`, `Barricade`, `Item`) je vytvořen nový `PictureBox`, nastaven jeho obrázek (`Image`), velikost (`Size`), pozice (`Location`) a případně další vlastnosti (např. průhledné pozadí).
- **Přidání na plochu:** `PictureBox` je přidán do kolekce `Controls` instance `Board` (`board.Controls.Add(sprite)`).
- **Aktualizace pozice:** Při každém snímku (každé iteraci `GameLoop`) je aktualizována pozice objektů změnou vlastnosti `Location` jejich `PictureBoxu`.
- **Animace:** U animovaných objektů (hráč, nepřítel) se v pravidelných intervalech mění obrázek (`Image`) podle aktuálního snímku animace.
- **Zničení objektu:** Pokud je objekt odstraněn (např. zabít nepřítel, sebrat předmět), je jeho `PictureBox` odstraněn z `Controls` a uvolněn z paměti (`Dispose`).
- **Zobrazení UI prvků:** UI prvky (životy, skóre, časovač, menu) jsou také realizovány jako `PictureBoxy`, `Labely`, `Panely` apod. a jsou dynamicky aktualizovány podle stavu hry.

Zdroje obrázků

- Všechny grafické assety jsou uloženy v souboru `Properties/Resources.resx`.
- Při inicializaci objektu je obrázek načten z `Resources` a případně upraven (např. změna velikosti, rotace).

Výhody a omezení zvoleného přístupu

- **Výhody:**
 - Jednoduchá správa objektů a jejich pozic díky `PictureBoxům`.
 - Snadná práce se změnou obrázků (animace).
 - Možnost využití standardních UI prvků Windows Forms pro menu, skóre, životy apod.
- **Omezení:**
 - Při větším počtu objektů může být správa `Controls` méně efektivní.
 - Omezené možnosti pro pokročilé grafické efekty.

Příklad vykreslení předmětu (Item)

```
1 public void Draw()
2 {
3     if (!Board.Controls.Contains(sprite))
4     {
5         Board.Controls.Add(sprite);
6         sprite.BringToFront();
7     }
8     sprite.Location = Position;
9     sprite.Visible = true;
10 }
```

Tato metoda zajistí, že je PictureBox s obrázkem předmětu přidán na plochu a nastaven na správnou pozici.

5 Načítání úrovní a práce s mapou

Načítání úrovní je řešeno pomocí textového popisu úrovní uloženého ve zdrojích projektu (Properties/Resources.resx), který je zpracován při spuštění hry.

Formát úrovní

- Všechny úrovně jsou uloženy v textovém souboru (resource) s názvem `levels`.
- Každá úroveň začíná hlavičkou s parametry (např. `Name:1 Time:60 SpawnRate:2000 Boss:0`), za kterou následuje blok řádků popisujících mapu.
- Každý řádek mapy je tvořen znaky, kde každý znak reprezentuje jeden typ objektu na dané pozici (např. `x` = barikáda, `p` = hráč, `e` = nepřítel, `s` = spawnovací oblast, `.` = volné pole).

Načtení úrovní do paměti

- Při spuštění hry metoda `LoadLevels()` načte všechny úrovně do statického seznamu.
- Metoda `LoadLevels()` rozdělí textový obsah podle řádků, detekuje začátek nové úrovně podle klíčového slova `Name:` a načte všechny parametry a mapové řádky do instance třídy `Level`.
- Každá instance `Level` obsahuje název, časový limit, rychlost spawnování, počet bossů a seznam řetězců s popisem mapy.

Vytvoření mapy a objektů

- Při startu nové úrovně je volána metoda `LoadMap(Board board, int levelIndex)`, která podle zvoleného indexu úrovně načte příslušný objekt `Level`.

- Pro každý řádek a každý znak v mapě je vytvořen odpovídající herní objekt (barikáda, hráč, nepřítel, spawnovací oblast) a přidán do příslušných kolekcí na herní ploše.
- Pozice každého objektu je vypočtena podle jeho souřadnic v mřížce a offsetu mapy.

Příklad načtení úrovně (zjednodušený výňatek)

```

1 public static void LoadLevels()
2 {
3     var lines = Properties.Resources.levels.Split(new[] { "\r\n",
4         "\n" }, StringSplitOptions.None);
5     int i = 0;
6     while (i < lines.Length)
7     {
8         if (lines[i].StartsWith("Name:"))
9         {
10             Level level = new Level();
11             levels.Add(level);
12         }
13         else
14         {
15             i++;
16         }
17     }
18 }

```

```

1 public void LoadMap(Board board, int levelIndex)
2 {
3     ....
4     switch (tile)
5     {
6         case 'x':
7             barricades.Add(new Barricade(board, new Point(x,
8                 y)));
9             break;
10        case 'p':
11            player = new Player(...);
12            entities.Add(player);
13            break;
14        case 'e':
15            entities.Add(new Enemy(...));
16            break;
17        case 's':
18            spawnAreas.Add(new SpawnArea(board, new Point(x,
19                y), level.SpawnRate));
20            break;
21    }
22    ....
23 }

```

Příklad úrovně

```
Name:level3 Time:50000 SpawnRate:3000 Boss:0
xxxxxxxxxxssssxxxxxxxxxx
X.....X
X.....X
X..X.....X..X
X.....X
X....S.XX...XX.S...X
X.....X
X....X.....X...X
X....X.....X...X
S.....b.....S
S.....p.....S
S.....S
X....X.....X...X
X....X.....X...X
X.....X
X....S.XX...XX.S...X
X.....X
X..X.....X..X
X.....X
X.....X
xxxxxxxxxxssssxxxxxxxxxx
```

6 Nastavení hry (Settings.cs)

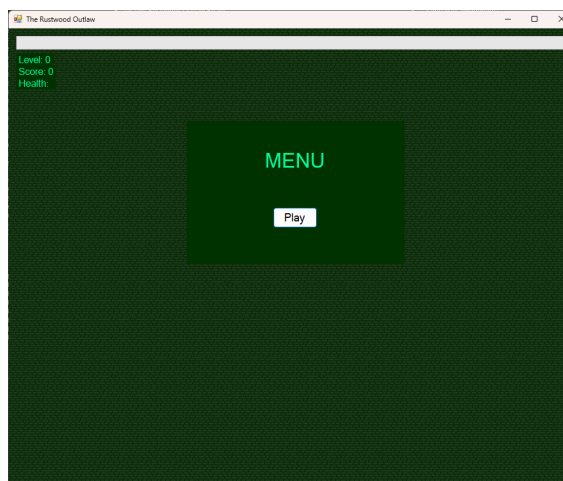
Třída `GameSettings` obsahuje globální konstanty a proměnné:

- `CellSize`, `MapSize`, `SpriteSize`
- `EnemySpeed`, `EnemyHealth`, `EnemyDamage`
- `PlayerSpeed`, `PlayerHealth`, `PlayerDamage`, `PlayerShootingSpeed`
- `enemySpawnChance`
- `itemOnGroundTime`
- `animationSpeed`
- `RefreshRate`
- `difficulty`

7 Uživatelské rozhraní

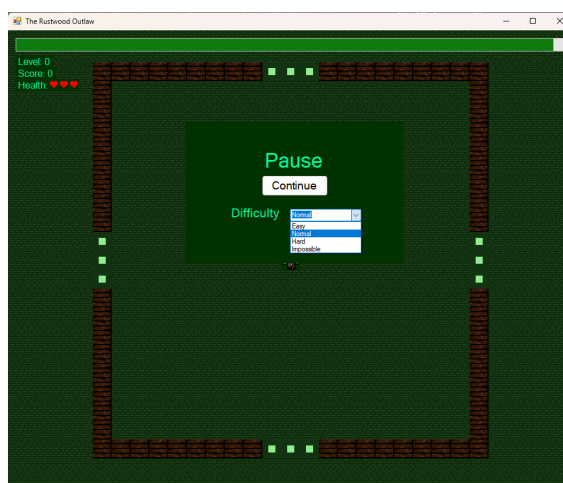
Hlavní prvky:

- `MainMenu` – hlavní menu. Slouží pouze na zapnutí hry.



- **Pause** – panel pauzy s výběrem obtížnosti.

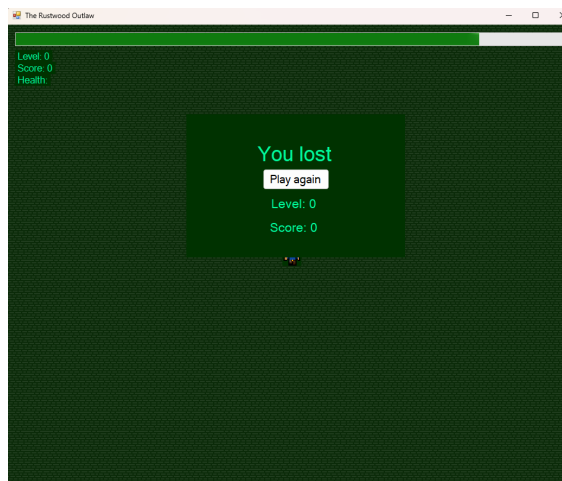
Stisknutím klávesy 'P' může hráč hru přerušit a případně změnit obtížnost, což mění šanci na spawnování nepřítele.



- **progressBar1** – časovač úrovně. Určuje čas do konce úrovně. Jakmile časovač dojde do konce, hra se automaticky přepne na další úroveň.
- **panelHearts** – zobrazení životů hráče.
- **lScore**, **lLevel**, **lLostScreenScore**, **lLostScreenLevel** – popisky skóre a úrovně.



- **pYouLost** – obrazovka prohry. Jakmile hráč zemře, zobrazí se tato obrazovka. Na ní se ukáže dosažená úroveň a počet zabitých nepřátel. Tlačítko 'Play again' pak spustí novou hru.



8 Zdroje a assety (Properties/Resources.resx)

- barricade, spawn_area, heart, hráč, nepřátelé, předměty – obrázky pro jednotlivé objekty
- levels – textový popis úrovní

9 Interakce uživatele

- Pohyb hráče: WASD
- Střelba: šipky
- Pauza: P
- Výběr obtížnosti: ComboBox v hlavním menu
- Stavba barikád: automaticky dle mapy

10 Možnosti rozšíření

- Nové typy nepřátel a bossů
- Vylepšování barikád a předmětů
- Více úrovní a map
- Ukládání postupu a žebříčky
- Zvukové efekty a hudba

11 Závěr

Projekt bych rád rozšířil o další nepřátele a případně i bosse. Také bych rád zlepšil vykreslování tak, aby hra běžela rychleji a hlavně plynuleji.