Dokumentace projektu *The Rustwood Outlaw*

Jakub Endlicher

3. července 2025

Obsah

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 havní 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 gameTimer` – 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 střílení
- `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 uložená 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`

```
private (int, int) PathFinding(Point goal)
{
    Point gridStart = RoundPointToGrid(position);
    Point gridGoal = RoundPointToGrid(goal);

    if (cestaJeStejna())
    {
        return (nextStep);
    }

    var cacheKey = (gridStart, gridGoal);
    if (cestaJizBylaNalezena)
    {
        return (nalezenaCesta);
    }

    var prevFromStart = new Dictionary<Point, Point>();
    ...
    visitedGoal.Add(gridGoal);

    Point? meetPoint = null;

    while (queueStart.Count > 0 && queueGoal.Count > 0)
    {
        if (Expand(queueStart, visitedStart, visitedGoal,
           prevFromStart, prevFromGoal, out meetPoint)) break;
        if (Expand(queueGoal, visitedGoal, visitedStart,
           prevFromGoal, prevFromStart, out meetPoint)) break;
    }

    if (meetPoint != null)
    {
        zrekonstruujCestu();
        return firstStep;
    }

```

```
    lastPath = null;
    return (0, 0);
}
```

```
private bool Expand(Queue<Point> queue, ..., out Point? meetPoint
    )
{
    meetPoint = null;
    if (queue.Count == 0) return false;
    Point current = queue.Dequeue();

    int[] dxx = { 0, 0, -1, 1, -1, -1, 1, 1 };
    int[] dyy = { -1, 1, 0, 0, -1, 1, -1, 1 };

    for (int dir = 0; dir < 8; dir++)
    {
        int nx = current.X + dxx[dir];
        int ny = current.Y + dyy[dir];
        Point next = new Point(nx, ny);
        if (CollidesAt(nx, ny)) continue;
        if (!visitedThis.Contains(next))
        {
            visitedThis.Add(next);
            prevThis[next] = current;
            queue.Enqueue(next);
            if (visitedOther.Contains(next))
            {
                meetPoint = next;
                return true;
            }
        }
    }
    return false;
}
```

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:

- `flot 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. zabitý nepřítel, sebraný 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)

```
public void Draw()
{
    if (!Board.Controls.Contains(sprite))
    {
        Board.Controls.Add(sprite);
        sprite.BringToFront();
    }
    sprite.Location = Position;
    sprite.Visible = true;
}
```

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)

```
public static void LoadLevels()
{
    var lines = Properties.Resources.levels.Split(new[] { "\r\n",
        "\n" }, StringSplitOptions.None);
    int i = 0;
    while (i < lines.Length)
    {
        if (lines[i].StartsWith("Name:"))
        {
            Level level = new Level();
            levels.Add(level);
        }
        else
        {
            i++;
        }
    }
}
```

```
public void LoadMap(Board board, int levelIndex)
{
    ....
        switch (tile)
        {
            case 'x':
                barricades.Add(new Barricade(board, new Point(x,
                    y)));
                break;
            case 'p':
                player = new Player(...);
                entities.Add(player);
                break;
            case 'e':
                entities.Add(new Enemy(...));
                break;
            case 's':
                spawnAreas.Add(new SpawnArea(board, new Point(x,
                    y), level.SpawnRate));
                 break;
        }
    ...
}
```

Příklad úrovně

```
Name:level3 Time:50000 SpawnRate:3000 Boss:0
xxxxxxxxxsssxxxxxxxxx
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
xxxxxxxxxsssxxxxxxxxx
```

6 Nastavení hry (`Settings.cs`)

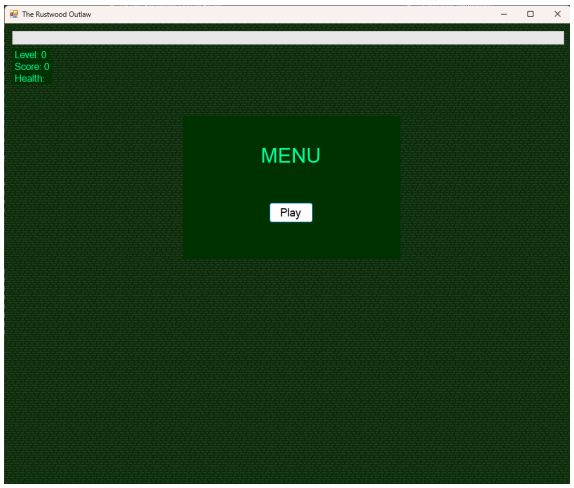
Třída `GameSettings` obsahuje globální konstanty a proměnné:

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

7 Uživatelské rozhraní

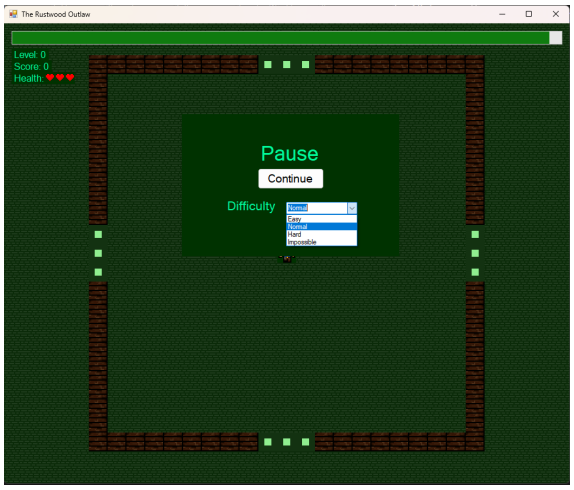
Hlavní prvky:

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



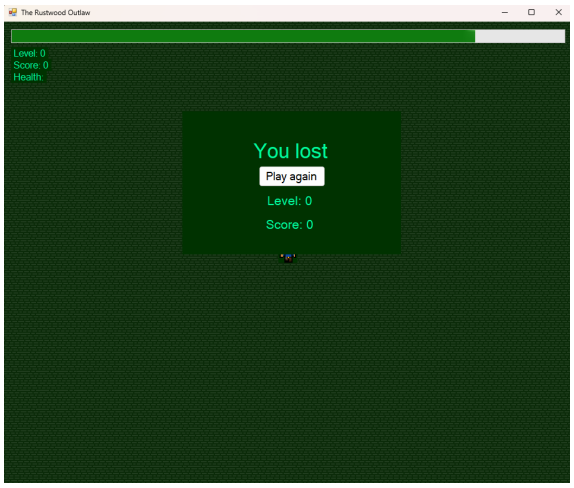
- `Pause` – panel pauzy s výběrem obtížnosti.

  Stisknutím klávesy 'P' může hráč hru přerušit a případně změnit obtížnost, což mění šanci na spawnutí 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: šipky nebo WASD
- Střelba: mezerník
- 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.