Online 2D versus hra

Dokumentace maturitní práce

Bohuslav Dušek

školní rok: 2025/2026 třída: 4.B

Zadání maturitní práce:

Cílem práce je vytvořit jednoduchou 2D hru s možností online hraní pomocí herního enginu Monogame (programovací jazyk C#). Práce zahrnuje vytvoření clientu i serveru. Ve hře budou proti sobě soupeřit dva hráči, kteří budou umístěni na stacionárních pozicích na herní mapě. Hráči na sebe budou moci navzájem útočit poštváváním počítačem-ovládáných nepřátel pohybujících se po mapě. Hra bude pokračovat nekonečně a stupňovat svou obtížnost, dokud se jeden z hráčů již nedokáže ubránit, a tím pádem prohraje.

OS: Linux

Teoretická část – Síťová komunikace v kontextu online her

Prohlašuji, že jsem na práci pracoval samostatně pouze za pomoci použitých zdrojů a že v práci i v dokumentaci jasně vymezuji, které části kódů jsou mým originálním dílem, které jsou upravenou verzí a které jsou převzaty v plném rozsahu.

----

podpis žáka

OBSAH

1. TEORETICKÁ ČÁST 5

1.1. Obvyklé uspořádání softwaru online hry 5
1.2. Update smyčka 5
1.3. Zpoždění 5
1.4. Konflikty 6
1.5. Zneužití 6

2. CÍLE PRÁCE 6

3. ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY 6

3.1. Prostředí 6
3.2. Energie 7
3.3. Monstra 7
3.4. Synchronizace 10
3.5. Grafické rozhraní 10
3.6. Struktura programu 11

4. PROGRAMOVACÍ PROSTŘEDKY 11

4.1. Programovací jazyk 11
4.2. Herní engine 11
4.3. Vývojové prostředí 11
4.4. Použité knihovny 12
4.5. Ostatní 12

5. ZHODNOCENÍ DOSAŽENÝCH VÝSLEDKŮ 12

6. INSTALACE 12

6.1. Stažení před-kompilované aplikace 12
6.2. Vlastní kompilace 12
6.3. Spuštění 12

7. OVLÁDÁNÍ 13

7.1. Hlavní menu 13
7.2. Hra 13

8. SEZNAM ZDROJŮ PRVKŮ POUŽITÝCH K VYTVOŘENÍ HRY 14

9. SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ 15

1. TEORETICKÁ ČÁST

1.1. Obvyklé uspořádání softwaru online hry

Podobně jako u většiny aplikací využívajících propojení přes síť, je u online her zapotřebí server a client. Clientem je v tomto případě samotná hra, kterou hráč spouští na svém zařízení. Herní server je buď provozován centrálně vývojářem hry, nebo je poskytnut hráči společně se hrou, dovolujíc mu spouštět ho na vlastním zařízení s vlastní konfigurací.

Veškerá data ohledně stavu a průběhu hry jsou uložena na serveru a zrcadlí se do clientu.

Hry se od ostatních síťových aplikací odlišují mnohem vyššími nároky na rychlost a přesnost. U mnoha her, zejména u žánru tzv. „střílečekˮ, často o výhře či prohře rozhodují milisekundy. Aby se zachovala férovost hry, měly by právě tyto milisekundy být v načasování reakcí hráčů, nikoli v rozdílu rychlosti jejich spojení. [1]

1.2. Update smyčka

V serveru běží konstantně během hry update smyčka, která aktualizuje stav hry podle dat získaných od jejího posledního průběhu. Jeden tento průběh se nazývá *tick*. Server by měl udržovat stabilní počet ticků za sekundu (TPS – *ticks per second*, neboli *tick rate*). Čím vyšší je TPS serveru, tím přesnější mohou být jeho výpočty. Podobná smyčka běží také uvnitř clientu, ta však slouží ke čtení vstupů hráče a zobrazování uživatelského rozhraní. V update smyčce na obou stranách také probíhá přijímání a odesílání dat přes síť. [3]

1.3. Zpoždění

Patrně největší technologickou limitací online her je fakt, že každá akce hráče i reakce hry na tuto akci se z pohledu hráče uskuteční se zpožděním daným rychlostí spojení clientu a serveru. I když tuto limitaci nelze zcela odstranit, existuje několik metod pro snížení jejího dopadu. [4]

1.3.1. Interpolace

Interpolace urychluje spojení snižováním množství dat, které je nutné mezi clientem a serverem vyměnit. V případě nutnosti zrcadlení plynulého jevu (např. pohybu objektu) není potřeba odesílat v každém ticku aktuální stav, data je možné posílat pouze v klíčových momentech a u příjemce chybějící části dopočítat. [3]

1.3.2. Předpověď jistých jevů

U mnoha akcí iniciovaných hráčem (tedy vycházejících z clientu) není nutně potřeba potvrzení serveru a tudíž je možné předpovědět jejich výsledek. Například pohyb postavy hráče v prostoru – pokud client zná parametry nutné pro výpočet pohybu, který hráč svými vstupy zadal, může hráčovi zobrazit pohybující se postavu dříve, než dostane odpověď ze serveru. Ve velké většině případů dojde server ke stejným výsledkům jako client a tudíž nebude potřeba předpovězená data nahrazovat. [2]

1.3.3. Předpověď na základě skrytých dat

Předpovědi lze dopomoci zrcadlením dat, která client hráčovi neukáže. Například může client přijímat i pozice nepřátel, které hráč aktuálně nevidí, a způsobit tím, že se při splnění správných podmínek hráči zobrazí okamžitě. V opačném případě by se nepřítel v hráčově zorném poli objevil až ve chvíli, kdy by server vyhodnotil že se tak má stát, tedy s malým, ale v zápalu hry znatelným zpožděním.

1.3.4. Nevýhody předpovědi

Nežádoucím vedlejším efektem použití této metody je fenomén tzv. *Peeker's advantage*, nejvíce znatelného ve hrách typu FPS (*first person shooter* – střílečka z pohledu první osoby). Ten poskytuje výhodu hráči vcházejícímu do nového prostoru, jelikož, díky přítomnosti skrytých dat, okamžitě při vstupu uvidí druhého hráče nacházejícího se uvnitř, zatímco

druhý hráč uvidí prvního teprve až obdrží příslušná data od serveru, který je nejdříve musí obdržet od clientu prvního hráče.

Dále předpověď samozřejmě nemůže zhodnotit všechny faktory. Pokud se na straně serveru změní podmínky tak, že zadanou akci provézt nelze (např. se před pohybujícím se hráčem objeví překážka), předpověď v clientu již nebude shodná se stavem na serveru, dojde tedy ke konfliktu, neboli desynchronizaci. [2]

1.4. Konflikty

Konflikt vznikne, když data na serveru neodpovídají datům na clientu, což může nastat z mnoha různých důvodů (zmíněný příklad se špatnou předpovědí, krátkodobý výpadek spojení, špatné parametry clientu pro předpověď atd.) a je nevyhnutelné, zejména u složitějších her. Ve většině případů je bezpečnější nahradit sporná data na clientu daty ze serveru. To však může být v některých případech pro hráče frustrující, jelikož se tak může vymazat akce, která byla z jeho pohledu perfektně provedena. Při vyhodnocování konfliktů ohledně akcí vyžadujících větší přesnosti, např. střelby, tedy případu, že client předpověděl zásah ale server ho nepotvrdil, se využívají spíše data z clientu – server rozhoduje podle rozmístění pozic hráče, jeho mířidel a jeho cíle, které sám hráč vidí, nikoli toho „reálného“ nacházejícího se na serveru. [1] [4]

Hráči tedy mohou mířit podle vlastního zraku s menším ohledem na rychlost spojení, čímž se pro ně hra stává férovější, protože se tak snižuje výhoda nabytá rychlejším spojením.

1.5. Zneužití

U většiny těchto optimalizací je též třeba pomýšlet na možné zneužití. Nejčastěji k němu dochází za pomoci úprav herního clientu, které mají hráči poskytovat nezamýšlenou výhodu. [1]

Nejspíš nejjednodušeji zneužitelným konceptem jsou skrytá data. Jelikož jsou tato data uložena v paměti stejně jako všechna ostatní, upravený client je schopen je bez velkých potíží hráči zpřístupnit. Na tomto principu funguje např. tzv. *wallhack*, který umožňuje hráči vidět nepřátele skrze překážky.

Dále lze zneužít situace, kdy se pro vyřešení konfliktu využívají data z clientu, a to tím způsobem, že client odešle serveru upravenou verzi své perspektivy, např. aby přesvědčil server o zásahu. Pokud není úprava příliš drastická, server ji nerozpozná a zásah započítá. Podobným způsobem lze v některých hrách manipulovat s pozicí hráče, což umožňuje pohyb vyšší rychlostí nebo na nezamýšlená místa. Tato manipulaci je však mnohem jednodušeji detekovatelná.

Při vyhodnocování konfliktů je tudíž třeba stanovit hranici maximální závažnosti desynchronizace, za kterou již server nebude používat data z clientu, ale použije svá.

2. CÍLE PRÁCE

Cílem práce bylo vytvořit hru pro dva hráče hratelnou propojením přes síť. Hra by měla pokračovat nekonečně a stupňovat svou obtížnost, dokud jeden z hráčů nevyhraje. Ve hře by měli být počítačem-ovládaní nepřátelé, kteří budou útočit na hráče. Hráči by měli být schopni pohyb nepřátel do jisté míry ovlivňovat, aby mohli znevýhodnit svého protivníka.

Při vytváření konceptu hry i některých prvků samotné implementace jsem čerpal inspiraci ze hry *Five nights at Freddy's*. [5]

3. ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

3.1. Prostředí

Hra se odehrává v prostředí znázorněném čtvercovou mřížkou 5 krát 5. Každé pole na mřížce symbolizuje jednu místnost. Skrze místnosti se směrem k hráčům pohybují monstra. Dvě místnosti jsou definovány jako „kanceláře“ – v každé z nich se nachází jeden hráč. Pokud se monstru podaří dosáhnout kanceláře, hráč v ní umístěný prohrává.

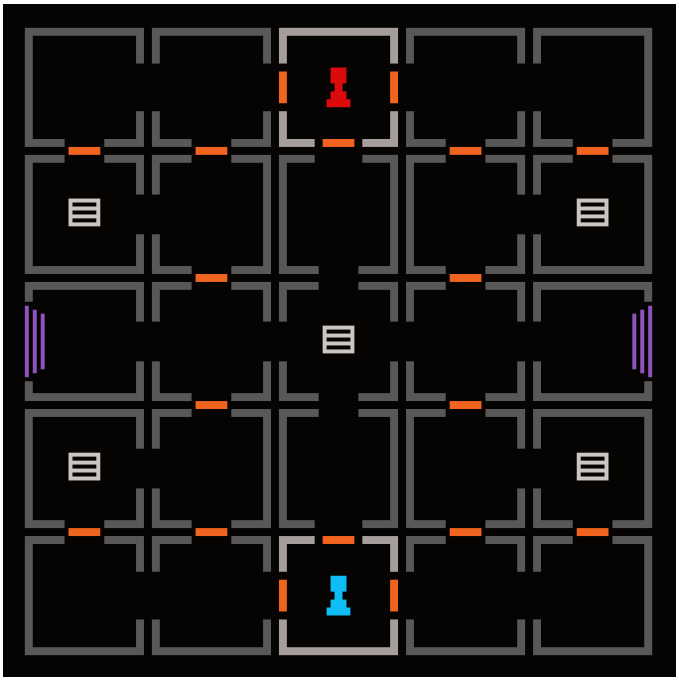
Hráč může přepínat mezi pohledem na kamery, kde vidí přibližující se monstra, a pohledem z první osoby, kde vidí svou kancelář.

Mezi některými místnostmi se nachází dveře. Hráči mohou zavírat dveře na své polovině hrací plochy a omezovat tím pohyb monster.

Hráči mohou dále na své polovině v každé místnosti vypínat a zapínat světlo. Když je světlo zhasnuté, některá monstra nelze vidět. Zároveň některá monstra na světlo reagují různými způsoby.

Část plochy, která je od obou hráčů vzdálená stejně nepatří nikomu. Místnosti v ní se v clientu vždy zobrazují jako rozsvícené, ale na serveru tak označeny nejsou. Tuto část budu nazývat „centrální chodba“.

Některé místnosti jsou propojené ventilačními šachtami. Monstra, která mohou šachty využívat, se mohou mezi těmito místnostmi přemístit jedním pohybem. Tyto místnosti jsou na mapě označeny šedou ikonou mřížky ventilace.



(Hrací plocha: modrá a červená figurka znázorňují umístění hráčů, červené linky dveře, šedé mřížky ventilaci a fialové linky pole pro zjevování monster)

3.2. Energie

Každý hráč má omezeonu energii, která se spotřebovává, když hráč rozsvítí světlo nebo zavře dveře. Na začátku hry je energie nastavena na maximální hodnotu a nikdy se nedoplňuje. Tato mechanika existuje, aby hráči nemohli jednoduše zavřít všechny dveře a znemožnit tím vlastní prohru, a zároveň aby hra nemohla nikdy trvat věčně.

Hodnotu energie server zpracovává každou jednu sekundu.

3.3. Monstra

Každé mosntrum má definované tři základní akce: Pohyb, útok a reset. Pohybem se monstrum přesune do jiné místnosti, která není kanceláří. Když se mosntrum pokusí o pohyb do kanceláře, pokud průchod není zablokován (neboli pokud nejsou zavřené příslušné dveře), nastane akce útok a daný hráč prohrává, v opačném případě nastane reset, který monstrum, ve většině případů, vrátí na jeho výchozí pozici.

Když monstrum na hráče zaútočí, hráč je vyděšen nečekaným hlasitým zvukem a zvětšeným obrázkem příslušného monstra.

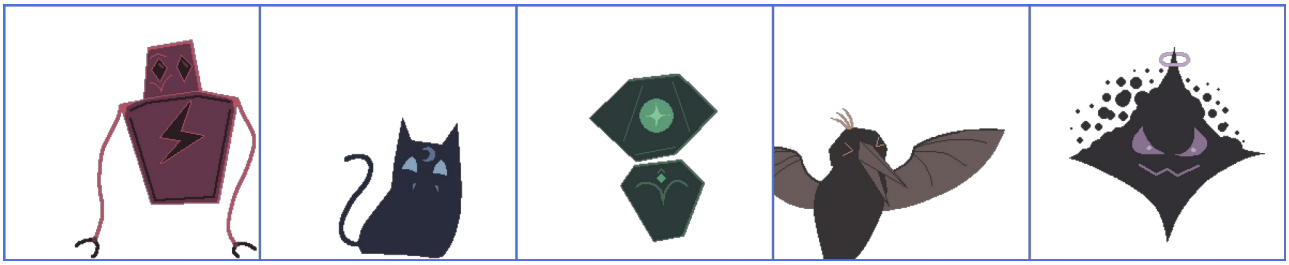
Dále má každé mosntrum hodnotu obtížnosti od jedné do deseti, která definuje rychlost jeho pohybu.

Monster je ve hře celkem pět, pojmenovaných *Lurk*, *Neko*, *Spot*, *Mare* a *Dash*.

1. **Lurk**: Postupně se přibližuje vždy k tomu hráči, který má více energie (pokud mají oba hráči stejně, nehýbe se). Při akci reset ubere energii hráči, na kterého se pokusil zaútočit, a vrátí se na jedno ze tří prostředních polí centrální chodby. Může se pohybovat ventilačními šachtami.

2. **Neko**: Jakmile poprvé nastane situace, kdy má jeden hráč méně energie než druhý, je hráč s menší hodnotou energie označen jako cíl. Při resetu se Neko vrací na jedno z prostředních tří polí centrální chodby a mění svůj cíl na druhého hráče. Pokud uběhne první fáze hry (definována níže) aniž by byl cíl zvolen, je vybrán náhodně a Nekova obtížnost je zvýšena z výchozích tří na osm.

   Nemůže se pohybovat do místností, kde je rozsvícené světlo. Pokud je světlo rozsvíceno v místnosti, kde se nachází, rychlostí jeden pohyb za sekundu se začne pohybovat ke svému cíli. Aby hráč mohl jeho přítomnost odhalit ve zhasnuté místnosti, ozývá při vybrání příslušné kamery kočičí předení. Při pohybu vydá zvuk mňoukání. Může se pohybovat ventilačními šachtami.
3. **Spot**: Začíná ve středu plochy. Po uběhnutí náhodného času se aktivuje a vydá zvukový signál. V aktivovaném stavu přetrvá 12 sekund, během kterých je každému hráči měřen čas strávený na této kameře. Po uběhnutí tohoto času se opět deaktivuje a učiní pohyb směrem k hráči, který má nižší naměřený čas. Pohybuje se pouze v rámci prostředního sloupce plochy. Při resetu se vrátí na střed a ubere hráči energii.
4. **Mare**: Při přidání do hry označí jako svůj cíl hráče s menší hodnotou energie. Pokud mají oba hráči stejně, vybere náhodně. Pokud by se měl pohnout z místnosti, kde je rozsvícené světlo, zaútočí místo toho na hráče, na jehož půlce se aktuálně nachází. Při resetu změní svůj cíl na druhého hráče a učiní jeden pohyb směrem k němu.
5. **Dash**: Vždy se nachází v jedné ze tří prostředních místností centrální chodby. Oba hráči musí zavřít dveře do své kanceláře korespondující s jeho pozicí (pokud z hráčovi perspektivy stojí Dash na levo od středu, musí zavřít levé dveře). Pokud oba hráči zavřou správné dveře nebo pokud je oba hráči nechají otevřené, nastane reset a Dash se přesune na jinou ze svých tří pozic. Pokud hráč dveře nezavře, ale jeho protivník ano, zaútočí Dash na hráče s otevřenými dveřmi.



(Textury monster Lurk, Neko, Spot, Dash a Mare – zobrazené v tomto pořadí zleva)

3.3.1. Pravděpodobnost pohybu

Většina akcí monster se může dít pouze v konstantních intervalech. Po každém uběhnutí intervalu monstrum učiní akci s pravděpodobností určenou jeho obtížností. Tyto způsob randomizace načasování pohybů je známý jako *„movement opportunity“*. Stejný přístup je použit v jak v původní hře *Five nights at Freddy‘s*, tak v mnoha dalších hrách vycházejících z ní. [8]

Pravděpodobnost úspěchu na konci intervalu se u všech monster řídí stejným vzorcem:

$$P = \frac{5 + 1.5^{d}}{5 + 1.5^{10}}$$

d - obtížnost monstra

Intervaly:

- Lurk - 5s
- Neko - 4s
- Spot - 6s (možnost aktivovat se), 10s (délka aktivovaného stavu, na konci stoprocentní šance na deaktivaci a případný pohyb)
- Mare - 6s
- Dash - 5s

K přístupu *movement opportunity* je třeba podotknout, že hráč může tuto mechaniku potenciálně využívat ve svůj prospěch. Pokud hráč zná intervaly jednotlivých monster, může své vlastní akce načasovat podle nich. Například pokud stojí jedno z monster přede dveřmi, stačí, aby byly dveře zavřené na konci jeho intervalu a bude dosaženo stejného efektu, jako kdyby byly zavřené celou dobu, ale za cenu menšího množství energie. Přestože je tento vedlejší efekt jednoduše odstranitelný randomizací intervalů, rozhodl jsem se ho ve hře ponechat, jelikož, dle mého názoru, se díky němu zvedá maximální úroveň dovedností hráče. Za tímto účelem jsem také do clientu přidal vestavěnou časomíru, aby hráči ochotní tuto mechaniku využívat nemuseli používat časomíru vlastní.

3.3.2. Pathfinding

Lurk, *Neko* a *Mare* využívají stejný algoritmus pro hledání cesty ke svému cíli.

Pro účely hledání cesty je herní plocha reprezentována ohodnoceným grafem, kde vertikální cesty z prostřední místnosti a cesty ventilací mají hodnotu 2 a všechny ostatní cesty mají hodnotu 1.

Graf je prohledáván od kanceláře cílového hráče metodou BFS (breadth-first search)[6] a ke všem dosaženým místnostem je přiřazena hodnota vzdálenosti. Nejdříve jsou tedy prohledány všechny místnosti sousedící s kanceláří, poté všechny místnosti sousedící s nimi atd. Pokud algoritmus při prohledávání narazí na místnost, která již má přiřazenou nižší nebo stejnou vzdálenost oproti té, která by ji byla přiřazena v tomto kroku, ignoruje ji.

```
01 protected Dictionary<MapTile, int> Search(MapTile startingTile) {
02   Dictionary<MapTile, int> distances = new(){ [startingTile] = 0 };
03   Queue<MapTile> tilesToSearch = new();
04   tilesToSearch.Enqueue(startingTile);
05
06   while (tilesToSearch.Count > 0){
07     MapTile currentTile = tilesToSearch.Dequeue();
08
09     List<MapTile> neighbours = GetNeighbours(currentTile,
10       c =>
11         AdditionalConnectorFilter(c) &&
12         (!c.Blocked || c.Type == ConnectorType.DOOR_OFFICE) &&
13         (!distances.ContainsKey(c.OtherTile(currentTile)) ||
14         distances[c.OtherTile(currentTile)] > distances[currentTile] + c.Value),
15       t =>
16         AdditionalTileFilter(t) &&
17         (!Enemy.BlocksTile || EnemyManager.GetByLocation(t).All
18         (e => !e.BlocksTile || e == Enemy)) &&
19         Server.Players.All(p => p.Value.state.officeTileId != t.Id));
20
21     neighbours.ForEach(t => {
22       distances[t] = distances[currentTile] + currentTile.GetConnector(t)!.Value;
23       tilesToSearch.Enqueue(t);
24     });
25   }
26
27   return distances;
28 }
```

(kód algoritmu v ONDServer.Enemies.RoamingPathfinder)

Dvě monstra využívající tento algoritmus nemohou existovat ve stejné místnosti, proto se prohledávání vyhýbá místnostem, kde takové monstrum je.

Dále algoritmus přeskakuje cesty zablokované dveřmi, kromě těch vedoucích do kanceláře. Tato výjimka existuje, protože by jinak bylo příliš jednoduché zmrazit monstrum v určité sekci plochy opakovaným zavíráním a otevíráním dveří kanceláře.

Uvedený kód dále pracuje s proměnnými *AdditionalConnectorFilter* a *AdditionalTileFilter*. Ty lze nastavit při vytvoření instance třídy v třídách ovládajících monstra a slouží k definici speciálního chování jednotlivých monster. Například *Neko* těchto proměnných využívá pro vyloučení rozsvícených místností z validních.

Po prohledání grafu dostaneme vzdálenost každé místnosti od cíle. Všechny místnosti sousedící s monstrem (jejich vzdálenost počítáme zvýšenou o hodnotu cesty k místnosti s monstrem) jsou poté porovnány s pozicí monstra a všechny, jejichž vzdálenost je vyšší než vzdálenost monstra o více než 1, jsou vyřazeny. Dále jsou vyřazeny všechny, ve kterých již monstrum bylo od posledního resetu. Každé zbývající místnosti je přiřazena hodnota rovná její vzdálenosti umocněné na -1. Poměr těchto hodnot je poměrem pravděpodobností, že se monstrum pohne do příslušné místnosti. Místnosti s nižší vzdáleností mají tedy vyšší pravděpodobnost být zvoleny.

Pokud monstrum sousedí s kanceláří, vždy se pokusí pohnout právě tam.

Pokud monstrum nemá žádné validní místnosti, kam se pohnout, výběr je opakován s inkluzí těch, kudy se již na aktuální cestě pohybovalo. Pokud ani poté není nalezena validní destinace, pohyb se neuskuteční.

Pro prohledávání grafu byla v dřívějších fázích vývoje využívána rekurzivní implementace metody DFS (depth-first search). [7] Stará verze algoritmu vždy nejdříve následovala jednu z cest aktuální místnosti a vracela se k předchozím místnostem v cestě až ve chvíli, kdy aktuální místnost neměla žádné validní sousedy. Toto vedlo k tomu, že algoritmus prozkoumával mnoho ne-ideálních cest, což ho činilo velmi neefektivním. Aktualizovaná verze již tento problém nemá, jelikož jsou všechny cesty prozkoumávány souběžně, a tudíž je možné včas vyřadit cesty, u kterých prozkoumání není třeba.

3.3.3. Fáze

Každou jednu minutu hry se přestoupí do další herní fáze. Fáze slouží k postupnému stupňování obtížnosti hry. Hra začíná fází 1, na jejímž začátku se zjeví *Lurk* a *Neko* s obtížností 4 na koncích centrální chodby. Na začátku každé další fáze nastane jedna ze tří věcí:

1. Do hry je přidáno nové monstrum. Monstra jsou přidávána ve specifickém pořadí:
    1. *Mare* nebo *Spot* (obtížnost 3)
    2. Ten z předcházející dvojice, kdo ještě ve hře není
    3. *Dash* (obtížnost 6)

    Lurk, *Neko* a *Mare* se zjevují na polích označených fialovými pruhy, *Spot* a *Dash* se zjevují ve středu plochy.
2. Dvakrát je vybráno náhodné monstrum a jeho obtížnost je zvýšena o 2 (může být vybráno dvakrát stejné).
3. Obtížnost všech monster je zvýšena o 1.

3.4. Synchronizace

Server a client komunikují pomocí jednoduchých datových struktur GameEvent a PlayerCommand.

PlayerCommand posílá client serveru ve chvíli, kdy hráč učiní akci, která má dopad na hru (zavření dveří, rozsvícení světla). Je v něm uloženo ID dané akce a její parametry (které dveře byly zavřeny...).

GameEvent posílá server oběma clientům najednou kdykoli, kdy se stav hry změní. GameEvent je často pouze konvertovaná forma PlayerCommandu, která má zpravit druhého hráče o činnosti prvního. Je v něm též uloženo ID jevu, který reprezentuje, parametry a případně ID hráče, který ho svou akcí vyvolal.

Dopady veškerých akcí hráče jsou clientem předpovídány a zobrazovány před odpovědí serveru.

3.5. Grafické rozhraní

Všechny vizuální prvky jsou v clientu definovány jako objekty třídy *UIElement* nebo tříd dědících z ní. Tyto objekty jsou uloženy v příslušné instanci třídy *Screen*, která reprezentuje jeden pohled hráče.

U objektu *UIElement* lze definovat funkci při kliknutí na element.

Pozice na obrazovce jsou dány celočíselným vektorem $x \in [0; 640]$, $y \in [0; 360]$ (objekt *Point*). Při vyšším rozlišení než 640x360 je pozice elementu vynásobena koeficientem, aby při vykreslení odpovídala reálné pozici v pixelech. Hra podporuje jakékoli rozlišení, které je celočíselným násobkem základního rozlišení 640x360 (1280x720, 1920x1080...).

3.6. Struktura programu

Program je rozdělen do pěti projektů, které jsou dále členěny takto:

1. Client (*ONDClient*)
    1. *Enemies* – Správa zrcadlených monster
    2. *GUI* – Grafické rozhraní
    3. *Map* – Správa zrcadlené hrací plochy
    4. *Net* – Komunikace se serverem
    5. *Sound* – Správa zvuků
2. Server (*ONDServer*)
    1. *Enemies* – Definice monster a řízení jejich akcí
    2. *Map* – Generování a uložení hrací plochy
    3. *Net* – Komunikace s clienty
3. *PacketLib* – Knihovna s definicemi packetů pro síťovou komunikaci importovaná do clientu i serveru.
4. *GlobalClassLib* – Knihovna s ostatními prvky sdílenými mezi clientem a serverem.
5. MonoGame library (*MonoGameLibrary*)

Celkový průběh hry je řízen třídou *ONDServer.GameLogic*.

Hlavní update smyčka serveru je umístěna v *ONDServer.Net.Server*, její protějšek v clientu se nachází v *ONDClient.Net.Client*. Cílové TPS serverové smyčky je 20, TPS clientové smyčky je 60 (je vyšší kvůli potřebě responsivnější aktualizace grafického rozhraní)

4. PROGRAMOVACÍ PROSTŘEDKY

Veškerý vývoj byl uskutečněn na operačním systému Linux.

4.1. Programovací jazyk

Hra byla vytvořena v programovacím jazyce C#. Tento jazyk jsem si zvolil, jednak protože je jedním z nejčastěji používaných jazyků ve vývoji her, jednak protože s ním mám více zkušeností než s jakýmkoli jiným jazykem.

4.2. Herní engine

Z enginů podporujících C# jsem si zvolil minimalistický engine MonoGame. V základu poskytuje vývojáři pouze nejzákladnější funkce: vykreslení obrázku na danou pozici na obrazovce, načtení stavu myši a klávesnice, apod. Není s ním spjat ani dedikovaný editor, veškeré dění se definuje v kódu.

Důvodem, proč jsem použil tento místo větších a komplexnějších enginů, jakými jsou například Unity nebo Godot, byla touha vytvořit hru s nízkými nároky na hardware a vytvořit ji sám od základů, s minimální pomocí od funkcí enginu a externích knihoven

4.3. Vývojové prostředí

Jako editor kódu jsem používal JetBrains Rider. Tento editor preferuji před editory Visual Studio a Visual Studio Code, které jsou vyvíjeny firmou Microsoft společně s jazykem C#, protože mi jeho funkce celkově lépe vyhovují a mám s nástroji firmy JetBrains dlouholeté zkušenosti, dále kvůli obavám o narušování soukromí spojeným s jakoukoli aplikací firmy Microsoft.

4.4. Použité knihovny

Externí knihovny jsem použil celkem dvě:

1. *MonoGame library*: spíše sbírka vzorků kódu poskytovaná vývojáři enginu MonoGame jako možný základ pro vývoj jakékoli hry. Ze vzorků jsem použil pouze část pro načítání textur (kterou jsem si sám lehce upravil podle svých potřeb) a přidal vlastní nadstavbu pro čtení vstupů z klávesnice a myši. Kód této knihovny lze najít v submodulu *MonoGameLibrary* hlavního repozitáře.
2. *LiteNetLib*: knihovna pro síťovou komunikaci, známá pro svou rychlost a jednoduchost. Umožňuje vývojáři definovat packety a data, která obsahují, a pomocí nich zajistit komunikaci mezi zařízeními. Vývojář dále pouze definuje, co se má stát při obdržení určitého packetu, většina ostatní správy se děje interně.

4.5. Ostatní

Veškeré vizuální prvky kromě použitého fontu jsou vytvořeny mnou v grafickém programu Aseprite.

Zvukové efekty pochází z webu Pixabay a jsou upravené programem Audacity.

Odkazy na zvukové efekty i na font naleznete v seznamu zdrojů použitých prvků.

5. ZHODNOCENÍ DOSAŽENÝCH VÝSLEDKŮ

S výsledky jsem celkově spokojen. Všechna svá základní předsevzetí jsem splnil, z nichž hlavní bylo vytvořit zábavnou a férovou hru, což se mi, dle mého názoru, povedlo.

Jediné, co bych sám sobě vyčetl, je, že se mi nepodařilo ve hře implementovat prostorový zvuk a možnost změnit rozlišení při běhu hry, což však považuji pouze za menší nedostatky.

6. INSTALACE

6.1. Stažení před-kompilované aplikace

Nevyžaduje předchozí instalaci žádných knihoven

V sekci Releases (Vydání) na stránce repozitáře lze nalézt zkompilované verze programu pro Linux a Windows. Příslušný archiv je třeba pouze stáhnout a extrahovat.

6.2. Vlastní kompilace

Pro případ, že by uživatel chtěl provézt kompilaci na vlastním zařízení. Vyžaduje .Net verze alespoň 9.0. Veškeré kompilační scripty jsou vytvořené pouze pro systém Linux.

Kompilaci lze provézt stažením celého repozitáře a spuštěním scriptu build-linux.sh – ten program zkompiluje a vytvoří odkaz na spouštěcí soubor. Scripty *build-self-contained-linux.sh* a *build-self-contained-win.sh* slouží pro vytvoření verzí aplikace spustitelných bez předchozí instalace jakýchkoli knihoven (těch verzí, které jsou ke stažení v Releases).

6.3. Spuštění

Po absolvování jednoho ze zmíněných postupů naleznete v nejvyšším adresáři spouštěcí soubor pro client nebo odkaz na něj (*OneNightDuel* ve verzi pro Linux, *OneNightDuel.bat* ve verzi pro Windows) a script pro spuštění serveru (*launch-standalone-server.sh*, respiktive *.bat*). Client se v základu spouští v rozlišení 1280x720. Pokud ho chcete spustit v jiném rozlišení, je třeba upravit soubor *resolution* ve složce *bin*, na jehož prvním řádku se nachází koeficient, kterým bude vynásobeno nejnižší možné rozlišení 640x360. Pokud bych tedy chtěl spustit aplikaci v rozlišení 1920x1080, zapsal bych do souboru číslo 3.

Server lze spustit buď v rámci clientu, nebo jako samostatný program provedením zmíněného scriptu v terminálu. Pokud je server spuštěn z clientu, automaticky se vypne po ukončení hry. Pokud je server spuštěn scriptem, spuštěný zůstane dokud není manuálně vypnut.

Pro komunikaci se serverem je na jeho zařízení nutné ve firewallu povolit přístup na port 9012.

Pokud spojení zařízení nefunguje, je možné, že byl server po předchozím spuštění nesprávně vypnut, zůstal v pasivním stavu běžet na pozadí a příslušný port zůstal zablokován. Zkontrolujte, zda na zařízení neběží proces *ONDServer* nebo *launch-standalone-server* a v případě nalezení ho vypněte.

7. OVLÁDÁNÍ

Hru lze kdykoli vypnout klávesou Escape.

7.1. Hlavní menu

V hlavním menu může hráč zadat své uživatelské jméno. Pokud není zadané žádné jméno, hráč se bude zobrazovat jako „Anonymous“.

Pokud se hráč připojuje na server, musí zde zadat IP adresu tohoto serveru. Server vždy běží na portu 9012, port proto není potřeba zadávat.

Alternativně může hráč stisknout tlačítko *Host*, což spustí server na tomto zařízení a automaticky k němu hráče připojí.

7.2. Hra

Při pohledu do kanceláře lze zavírat její troje dveře klávesami A, W a D. Tyto dveře jsou ovládatelné pouze v tomto pohledu.

Přepínat mezi pohledy do kanceláře a na kamery lze klávesou mezerník.

Při pohledu na kamery lze přepínat mezi kamerami klikáním levým tlačítkem myši na jednotlivé místnosti. Aktuálně vybraná kamera je označena ikonou modrého oka. Červené oko označuje vybranou kameru protivníka. Klávesami W a S lze ovládat dveře nad a pod vybranou kamerou. Červeně označené dveře jsou otevřené, zeleně označené jsou zavřené.

Klávesou F lze přepínat světlo na vybrané kameře. Místnost, kde je rozsvícené světlo je označena žlutým proužkem.

8. SEZNAM ZDROJŮ PRVKŮ POUŽITÝCH K VYTVOŘENÍ HRY

Vzorky kódu pro MonoGame library: https://github.com/MonoGame/MonoGame.Samples/tree/3.8.4/Tutorials/learn-monogame-2d/src/07-Optimize-Texture-Rendering/MonoGameLibrary

Font (Ponderosa): https://www.1001fonts.com/ponderosa-font.html

Původní zvukové efekty:

- Dunění v pozadí: https://pixabay.com/sound-effects/musical-creepy-industrial-sounds-ambience-482892/
- Neko:
  - Mňoukání: https://pixabay.com/sound-effects/nature-meow-sfx-405456/
  - Předení: https://pixabay.com/sound-effects/film-special-effects-cat-purr-sfx-482870/
  - Syčení: https://pixabay.com/sound-effects/film-special-effects-hiss3-103123/
- Spot (aktivace a pohyb): https://pixabay.com/sound-effects/film-special-effects-robot-noises-70217/
- Dash (pohyb): https://pixabay.com/sound-effects/nature-loon-call-335487/
- Mare (pohyb): https://pixabay.com/sound-effects/horror-beast-growl-sound-376873/
- Zavření dveří: https://pixabay.com/sound-effects/film-special-effects-metallic-door-shut-98740/
- Otevření dveří: https://pixabay.com/sound-effects/opening-metal-door-199581/
- Zapnutí / vypnutí monitoru: https://pixabay.com/sound-effects/film-special-effects-keyboard-click-327728/
- Přepnutí kamery: https://pixabay.com/sound-effects/film-special-effects-computer-mouse-click-352734/
- Zapnutí světel: https://pixabay.com/sound-effects/household-light-switch-382712/
- Vypnutí světel: https://pixabay.com/sound-effects/household-light-switch-on-382714/
- Vyděšení: https://pixabay.com/sound-effects/film-special-effects-angle-grinder-313298/
- Vypotřebování energie: https://pixabay.com/sound-effects/power-outage-451574/
- Nová fáze: https://pixabay.com/sound-effects/musical-sfx-12pm-distant-chiming-tower-clock-sound-effect-463258/

9. SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] GAMBETTA, Gabriel. *Fast-Paced Multiplayer (Part I): Client-Server Game Architecture* [online]. ©2026 [cit. 2026-03-30]. Dostupné z URL: https://gabrielgambetta.com/client-server-game-architecture.html

[2] GAMBETTA, Gabriel. *Fast-Paced Multiplayer (Part II): Client-Side Prediction and Server Reconciliation* [online]. ©2026 [cit. 2026-3-30]. Dostupné z URL: https://gabrielgambetta.com/client-side-prediction-server-reconciliation.html

[3] GAMBETTA, Gabriel. *Fast-Paced Multiplayer (Part III): Entity Interpolation* [online]. ©2026 [cit. 2026-03-30]. Dostupné z URL: https://gabrielgambetta.com/entity-interpolation.html

[4] GAMBETTA, Gabriel. *Fast-Paced Multiplayer (Part IV): Lag Compensation* [online]. ©2026 [cit. 2026-03-30]. Dostupné z URL: https://gabrielgambetta.com/lag-compensation.html

[5] *Wikipedia, the free encyclopedia* [online]. 2026-03-23 [cit. 2026-03-30]. Dostupné z URL: https://en.wikipedia.org/wiki/Five_Nights_at_Freddy's_(video_game)

[6] *Wikipedia, the free encyclopedia* [online]. 2025-10-21 [cit. 2026-03-30]. Dostupné z URL: https://en.wikipedia.org/wiki/Breadth-first_search

[7] *Wikipedia, the free encyclopedia* [online]. 2026-02-18 [cit. 2026-03-30]. Dostupné z URL: https://en.wikipedia.org/wiki/Depth-first_search

[8] CERIW. *Fnaf1-ai-simulator* [online]. 2024-1-12 [cit. 2026-03-30]. Dostupné z URL: https://github.com/CeriW/fnaf1-ai-simulator/blob/master/research/how-the-game-works.md