Střední průmyslová škola Třebíč

Maturitní práce

SÍŤOVÁ HRA LODĚ

Profilová část maturitní zkoušky

Studijní obor: Informační technologie

Třída: ITB4

Školní rok: 2024/2025 Josef Vaněk

Zadání práce

Střední průmyslová škola Třebíč
Manželů Curieových 734, 674 01 Třebíč

Zadání ročníkové práce

Obor studia: **18-20-M/01 Informační technologie**

Celé jméno studenta:	**Josef Vaněk**		
Třída:	**ITB4**	Školní rok:	**2024/2025**
Číslo tématu:	**2**		
Název tématu:	**Síťová hra "lodě"**		
Rozsah práce:	**15 - 25 stránek textu**		

Specifické úkoly, které tato práce řeší:

Napište program, který bude nahrazovat klasickou hru lodě pro dva hráče. Hra bude realizována jako síťová na dvou PC spojených pomocí protokolu TCP/IP. Na každém PC bude hrací plocha nejméně 50x50 pozic. Navrhněte způsob interní reprezentace hry, způsob vykreslování vlastních lodí, střelby i úspěšných zásahů. Navrhněte způsob ovládání hry i rozmístění vlastních lodí na začátku hry. Pro spojení PC využijte knihovnu Sockets. Navrhněte vhodný přenosový protokol. Pro realizaci využijte vývojové prostředí Visual Studio, pro správu verzí platfomu GitHub.

Termín odevzdání:	**28. března 2025, 23.00**
Vedoucí projektu:	**Ing. Ladislav Havlát**
Oponent:	**Ing. Jana Veselá**
Schválil:	**Ing. Petra Hrbáčková, ředitelka školy**

ABSTRAKT

Tato práce se zaměřuje na vývoj síťové hry Lodě, která umožňuje dvěma hráčům soupeřit přes lokální síť. Hlavním cílem bylo vytvořit funkční aplikaci kombinující klasickou herní mechaniku s moderními technologiemi, konkrétně jazykem C# a .NET Frameworkem. V teoretické části jsou rozebrány použité nástroje, jako je Visual Studio 2022 pro vývoj, TCP/IP protokol pro síťovou komunikaci a GitHub pro správu verzí. Praktická část popisuje návrh herní logiky včetně tříd pro herní plochu a lodě, implementaci síťového propojení mezi klientem a serverem a tvorbu uživatelského rozhraní s rozšířenou mřížkou 50×50 políček. Aplikace podporuje základní herní prvky – rozmístění lodí, střídání tahů, detekci zásahů a potopení flotily – a poskytuje okamžitou zpětnou vazbu. Přestože výsledná verze splňuje základní požadavky, práce upozorňuje na omezení, jako je jednoduché grafické rozhraní nebo absence pokročilých funkcí (AI protihráč, šifrování).

KLÍČOVÁ SLOVA

síťová hra Lodě, TCP/IP protokol, architektura klient-server, uživatelské rozhraní

ABSTRACT

This thesis focuses on the development of a networked game, Ships, which allows two players to compete over a local network. The main goal was to create a functional application combining classical game mechanics with modern technologies, specifically C# and the .NET Framework. The theoretical part discusses the tools used, such as Visual Studio 2022 for development, TCP/IP protocol for network communication and GitHub for version control. The practical part describes the design of the game logic including classes for the game board and ships, the implementation of the networking between client and server, and the creation of a user interface with an extended 50×50 grid of patches. The application supports basic game elements - ship deployment, turn rotation, hit detection and fleet sinking - and provides instant feedback. Although the final version meets the basic requirements, the paper points out limitations such as the simple graphical interface or the lack of advanced features (AI adversary, encryption). The project demonstrates the practical application of programming skills and serves as a starting point for future extensions, such as the

integration of 3D graphics or cross-platform support. The work also reflects the challenges of optimizing network code and synchronizing game state between players.

KEYWORDS

network game Ships, TCP/IP protocol, client-server architecture, user interface

PODĚKOVÁNÍ

Děkuji Ing. Ladislavovi Havlátovi za cenné připomínky a rady při tvorbě maturitní práce.

V Třebíči dne 28. března 2025 podpis autora

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl v ní všechny prameny, literaturu a ostatní zdroje, které jsem použil.

V Třebíči dne 28. března 2025

podpis autora

Obsah

Úvod ... **8**
1 Teoretická část ... **9**
1.1 Použité technologie ... **9**
1.1.1 Visual Studio 2022 ... 9
1.1.2 Jazyk C# ... 9
1.1.3 .NET Framework ... 10
1.1.4 TCP/IP Protokol ... 11
1.1.5 GitHub ... 12
1.1.6 Architektura Klient – Server ... 12
1.2 Princip hry „lodě“ ... **13**
1.3 Historie hry ... **14**
1.4 Nejznámější taktiky ... **15**
1.4.1 Checkboard (Šachovnicový vzor) ... 15
1.4.2 Hunt and Target (Hledání a ničení) ... 15
1.4.3 Edge Avoidance (Vyhýbání se okrajům) ... 16
1.4.4 Paritní strategie (Sudá/lichá pole) ... 16
1.4.5 Density Tracking (Sledování hustoty) ... 16
1.4.6 Salvo Varianta ... 16
1.4.7 Psychologické hry ... 16
1.4.8 Křížová eliminace ... 16
2 Praktická část ... **17**
2.1 Implementace logiky hry ... **17**
2.1.1 Třída Gameboard ... 17
2.1.2 Třída Ship ... 19
2.2 Komunikace přes síť ... **20**
2.3 Uživatelské rozhraní ... **22**
2.3.1 Technické parametry: ... 23
2.3.2 Ovládací prvky ... 23
2.3.3 Interakce s uživatelem ... 23
Závěr ... **25**
Seznam použitých zdrojů ... **26**
Seznam použitých symbolů a zkratek ... **28**

Seznam obrázků 29

Úvod

„Síťová hra Lodě“ je přetvoření klasické deskové hry „Lodě“ do hry pro dva hráče, kterou mohou hrát na lokální síti. Jako cíl mám umožnění hraní této hry na dvou počítačích propojených TCP/IP protokolem. Hráči budou mít k dispozici hrací plochu o rozměrech 50x50 políček, kde rozmístí své lodě a zahájí souboj s protivníkem. Aplikace nabídne jednoduché ovládání rozmisťování a volbu délky lodí, střelbu na oponenta a uživatel uvidí, jestli protivníkovu loď zasáhl, nebo jestli se netrefil. Pokud potopí soupeřovi celou loď dostane o tom oznámení. Protihráč naopak na své herní ploše uvidí, kam oponent vystřelil. Hru vytvořím v kódovém editoru Microsoft Visual Studio 2022 v programovacím jazyce C# s využitím .NET Frameworku. Pro síťovou komunikaci použiji protokol TCP/IP a knihovnu System.Net.Sockets. Pro správu verzí projektu využiji platformu GitHub, s pomocí GitHub desktopové aplikace pro snadnější nahrávání souborů do repozitáře.

1 Teoretická část

Zde budou informace o použitých technologiích, principu hry a její historii.

1.1 Použité technologie

Technologie použity pro tvorbu hry.

1.1.1 Visual Studio 2022

Visual Studio je výkonné integrované vývojové prostředí (IDE) od společnosti Microsoft, které je určené pro tvorbu softwaru v široké škále programovacích jazyků, včetně C++, C#, Pythonu a JavaScriptu. Nabízí širokou paletu nástrojů pro usnadnění vývoje, jako je pokročilé ladění, automatické doplňování kódu, refaktoring a integrovaná správa verzí pomocí Gitu. [9] Díky podpoře různých frameworků, například .NET, umožňuje vývoj desktopových, webových i mobilních aplikací.

Jednou z hlavních předností Visual Studia je jeho flexibilita a rozšiřitelnost – uživatelé si mohou přizpůsobit prostředí pomocí rozsáhlé knihovny rozšíření. K dispozici jsou různé verze: Community (zdarma pro jednotlivce a malé týmy), Professional (pro zkušené a pokročilé vývojáře) a Enterprise (s pokročilejšími funkcemi pro velké firmy). Alternativou je Visual Studio Code, lehčí a open-source varianta, která je ideální pro jednodušší projekty. Celkově je Visual Studio jedno z nejkomplexnějších a nejpoužívanějších vývojových prostředí na trhu. [5]

1.1.2 Jazyk C#

C# je moderní objektově orientovaný programovací jazyk vyvinutý společností Microsoft jako součást platformy .NET. Kombinuje prvky jazykových rodin C a C++ s jasnou strukturou a syntaxí, která podporuje čitelnost kódu a snižuje riziko chyb. Jazyk je silně typovaný, což vyžaduje explicitní deklaraci datových typů proměnných. Je navržen pro tvorbu aplikací různého rozsahu – od desktopových řešení (Windows Forms, WPF) přes webové služby (ASP.NET) až po mobilní aplikace (Xamarin) a hry (Unity engine).

Základem C# je koncept tříd a objektů, kde třídy definují vlastnosti a metody, zatímco objekty reprezentují konkrétní instance. Dědičnost, zapouzdření a polymorfismus umožňují vytvářet modulární a rozšiřitelné architektury. Jazyk podporuje rozhraní

(interfaces) pro definici společných kontraktů mezi nezávislými komponentami a abstraktní třídy pro částečnou implementaci. Výjimky (try/catch bloky) slouží k řízení chybových stavů, zatímco správa paměti je automatizována prostřednictvím garbage collectoru, což minimalizuje úniky paměti.

C# se průběžně vyvíjí, s novými verzemi přidávajícími funkce jako pattern matching, records nebo top-level statements pro zjednodušení syntaxe. Integruje se s ekosystémem .NET, včetně knihoven pro práci se soubory, sítěmi nebo databázemi (Entity Framework). Díky kompilaci do mezijazyka (IL) a běhovému prostředí (CLR) je kód multiplatformní, s podporou pro Windows, Linux i macOS. Bezpečnostní mechanismy, jako jsou namespaces pro logické členění kódu nebo modifikátory přístupu (public, private), zajišťují kontrolu nad architekturou projektu. C# je vhodný jak pro začátečníky díky přehledné syntaxi, tak pro experty vyžadující vysoký výkon a flexibilitu. [7]

1.1.3 .NET Framework

.NET Framework je vývojová platforma od Microsoftu, navržená pro tvorbu a provoz aplikací primárně v prostředí Windows. Jako základ ekosystému .NET poskytuje komplexní infrastrukturu, která zahrnuje běhové prostředí (CLR – Common Language Runtime), knihovny tříd (FCL – Framework Class Library) a nástroje pro kompilaci a správu kódu. Zaměřuje se na integraci různých programovacích jazyků (C#, VB.NET, F#) pod jednotnou sadu knihoven, což umožňuje vývojářům kombinovat komponenty napsané v odlišných jazycích do jednoho projektu.

Architektura .NET Framework stojí na principu spravovaného kódu, kde CLR zajišťuje automatickou správu paměti (garbage collection), bezpečnostní kontrolu přístupu a optimalizaci výkonu během běhu aplikace. Knihovny tříd nabízejí předpřipravené funkce pro práci se soubory, sítěmi, grafikou, databázemi (např. ADO.NET) nebo webovými službami (WCF, ASP.NET), což urychluje vývoj bez nutnosti psát kód od nuly. Platforma podporuje vytváření desktopových aplikací (Windows Forms, WPF), webových řešení (ASP.NET Web Forms, MVC) i enterprise služeb.

Klíčovou vlastností je kompatibilita s legacy systémy prostřednictvím COM (Component Object Model) a P/Invoke pro volání nativních knihoven. Bezpečnostní

model zahrnuje řízení přístupu na úrovni kódu (CAS – Code Access Security) a digitální podpisy. Pro nasazení aplikací slouží nástroje jako ClickOnce nebo Global Assembly Cache (GAC).

Ačkoli .NET Framework byl postupně doplněn o multiplatformní varianty (.NET Core, Mono), jeho hlavní doménou zůstávají Windows aplikace s důrazem na stabilitu a integraci s OS. Verze 4.x přinesly paralelní programování, dynamické komponenty nebo vylepšenou podporu asynchronních operací (async/await). Platforma je vhodná pro firemní prostředí, kde dlouhodobá podpora a kompatibilita s existujícími systémy hrají klíčovou roli. I přes nástup .NET 6+ zůstává základem pro mnoho enterprise řešení, modernizovaných postupným přechodem na hybridní architektury. [8]

1.1.4 TCP/IP Protokol

TCP/IP je soubor protokolů, který tvoří základ komunikace v moderních počítačových sítích včetně internetu. Definuje pravidla pro přenos dat mezi zařízeními nezávisle na jejich hardwaru nebo operačním systému a rozděluje komunikaci do vrstev (model TCP/IP), z nichž každá řeší specifický úsek procesu. Aplikace (jako webové prohlížeče nebo e-mailové klienty) využívají aplikační vrstvu (HTTP, FTP, SMTP) k formulaci požadavků. Transportní vrstva (TCP, UDP) zajišťuje spojení mezi koncovými body – TCP garantuje spolehlivé doručení dat prostřednictvím potvrzování přijetí, řízení toku a opravy chyb, zatímco UDP je rychlejší, ale bez záruky doručení.

Internetová vrstva (IP) adresuje a směruje pakety mezi sítěmi pomocí logických adres (IPv4/IPv6), zatímco vrstva síťového rozhraní (Ethernet, Wi-Fi) spravuje fyzický přenos přes konkrétní médium. Klíčovým principem je paketové přepínání: data se rozdělí na menší jednotky, které putují sítí nezávisle na sobě a jsou na cíli opět složeny. TCP před přenosem navazuje spojení třífázovým handshake (SYN, SYN-ACK, ACK), během nějž synchronizuje parametry komunikace.

IP adresy identifikují zařízení v síti, porty pak konkrétní služby na těchto zařízeních (např. port 80 pro HTTP). Směrovací protokoly (RIP, OSPF) a ARP (převod IP na MAC adresy) zajišťují efektivní trasování. TCP/IP integruje mechanismy pro detekci zahlcení sítě (congestion avoidance) a dynamicky upravuje rychlost přenosu. Bezpečnostní vrstvy (TLS/SSL) se často přidávají nad TCP pro šifrování, ačkoli samotný TCP/IP neobsahuje nativní ochranu proti útokům. Protokol je

nepostradatelný pro veškerou internetovou infrastrukturu – od lokálních sítí po globální cloudové služby – a tvoří technický základ pro interoperabilitu heterogenních systémů. [6]

1.1.5 GitHub

GitHub je webová platforma a služba založená na distribuovaném verzovacím systému Git, určená pro správu zdrojového kódu, spolupráci vývojářů a organizaci softwarových projektů. Slouží jako centrální úložiště (repository), kde lze ukládat kód včetně historie změn, větvení (branches) a možnosti reverze k předchozím verzím. Hlavní funkcí je usnadnění týmové práce: vývojáři navrhují úpravy prostřednictvím pull requestů, které umožňují revizi kódu, diskusi o změnách a automatické testování před sloučením do hlavní větve.

Platforma integruje nástroje pro CI/CD (GitHub Actions), správu úkolů (Issues, Projects) a dokumentaci (Wiki, GitHub Pages). Podporuje otevřený přístup k open-source projektům, včetně licencování a sociálních funkcí jako „forkování“ (kopírování cizího repozitáře) nebo „hvězdičky“ pro označení oblíbených projektů. GitHub také nabízí bezpečnostní funkce: skenování zranitelností (Dependabot), kontrola přístupu (role contributorů) a šifrovanou komunikaci přes SSH/HTTPS.

Pro firmy poskytuje GitHub Enterprise s rozšířenými možnostmi auditování, SAML autentizací a hostingem v privátních cloudech. Díky integracím s externími nástroji (Jira, Slack, Azure DevOps) tvoří páteř moderního vývojového workflow. Platforma zároveň slouží jako globální komunikační hub pro vývojáře – od neformálních diskusí v sekci „Discussions“ až po hosting statických webů (GitHub Pages). S nástupem GitHub Copilot (AI asistent pro psaní kódu) expanduje i do oblasti automatizace vývoje. Jako de facto standard pro verzování a kolaboraci je GitHub klíčový pro open-source ekosystém i enterprise týmy vyžadující transparentnost a škálovatelnost. [9]

1.1.6 Architektura Klient – Server

V rámci jazyka C# a platformy .NET je pro implementaci klient-server komunikace zásadní namespace System.Net.Sockets, který poskytuje nízkoúrovňové nástroje pro práci se síťovými protokoly, zejména TCP/IP. Tento model umožňuje vytvářet stabilní, spojově orientovanou komunikaci mezi klienty a servery, kde TCP zajišťuje

spolehlivé doručení dat prostřednictvím potvrzování paketů, kontrolu toku a opravu chyb.

Základem je třída TcpListener pro serverovou stranu, která naslouchá na určeném portu a IP adrese, přijímá příchozí spojení a pro každého klienta vytváří dedikované TcpClient připojení. Na straně klienta se využívá TcpClient k navázání spojení se serverem, přičemž výměna dat probíhá přes NetworkStream – bajtový proud pro čtení a zápis. Asynchronní metody (AcceptTcpClientAsync, ReadAsync, WriteAsync) umožňují neblokující operace, což je klíčové pro škálovatelnost serverů obsluhujících desítky či stovky paralelních klientů.

TCP/IP v System.Net.Sockets garantuje řádné doručení zpráv ve správném pořadí, ačkoli vyžaduje explicitní správu spojení – například detekci výpadků klientů pomocí heartbeat zpráv nebo obnovení přerušené komunikace. Protokol také umožňuje definovat vlastní formáty zpráv (např. prefixy s délkou dat), což je užitečné pro binární protokoly v real-time aplikacích nebo IoT zařízeních.

Pro komplexnější scénáře (šifrování, autentizace) se System.Net.Sockets kombinuje s vyššími vrstvami, jako je SslStream pro TLS, nebo se integruje do frameworků jako ASP.NET Core pro HTTP/WebSocket komunikaci. TCP/IP v této architektuře tvoří páteř pro enterprise systémy, hry vyžadující nízkou latenci nebo průmyslové aplikace, kde stabilita spojení a kontrola datových toků jsou prioritou. I přes náročnější správu oproti vyšším abstrakcím (REST API) poskytuje System.Net.Sockets nezbytnou flexibilitu pro optimalizované síťové aplikace v C#. [11]

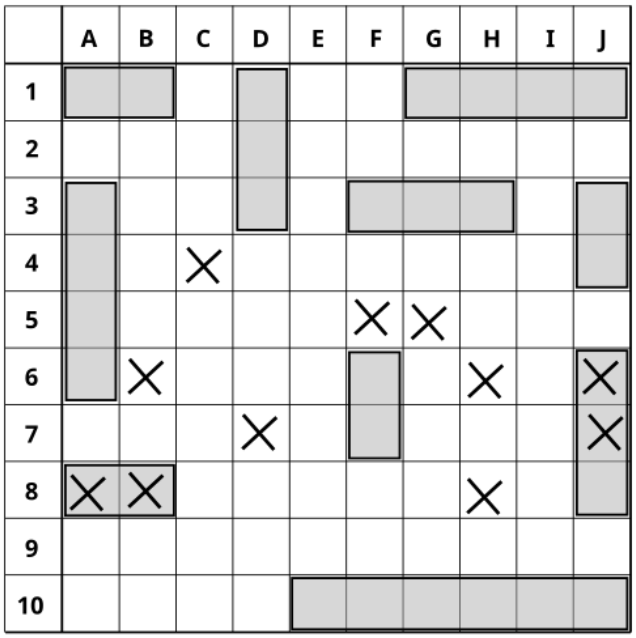
1.2 Princip hry „lodě“

Lodě je hra, kde jde hráčovi o to, aby zničil celou flotilu lodí protihráče. Na začátku hry si hráč rozloží svoji flotilu lodí na svou herní plochu a připraví si prázdnou tabulku/herní plochu pro zápis souřadnic svých střel na protihráčovu plochu. Herní plocha je mřížka, označená na ose x písmeny abecedně od A (A, B, C, D, E, F, G, H, ...), na ose y naopak číslicemi od jedničky (1, 2, 3, 4, 5, 6, 7, 8, …), někdy mohou být na obou osách číslice.

Hráči mohou každý tah jednou vystřelit na protihráče tím, že mu sdělí souřadnice políčka, kam chtějí střílet a dostanou odpověď, jestli se trefili, nebo ne. Soupeř si

zapíše na svojí herní plochu, na jaké políčko hráč zaútočil. Takto se střídají, dokud nepotopí všechny soupeřovi lodě.

V některých variantách této hry hráči mohou pokládat miny, na které když protivník zaútočí tak explodují a jako by vystřelili na vlastní herní plochu, nebo další varianty můžou mít radary které oskenují políčka okolo toho, kde byl ten radar položen, a podobné vychytávky. Variant této hry je mnoho s mnoha odlišnými taktikami, ale základní princip hry, výstřel + odpověď, zůstává. [1][10]



Obrázek 1.1 Rozehraná hra lodě [1]

1.3 Historie hry

Hra Lodě, známá globálně jako Battleship, má kořeny v počátku 20. století. Původně vznikla jako papírová hra pro dva hráče, kterou si vojáci během první světové války krátili volné chvíle. Inspiraci pravděpodobně čerpala z taktických map námořních bitev, kde hráči odhadovali pozice nepřátelských plavidel. První komerční verzi vydal v 30. letech americký výrobce her Milton Bradley pod názvem Broadsides, ale masivní popularitu získala až v 60. letech pod ikonickým jménem Battleship.

V 70. letech se hra přenesla do elektronické podoby – například do kapesních elektronických zařízení, která automatizovala kontrolu zásahů. Skutečný zlom přišel s nástupem počítačů a konzolí v 80. a 90. letech, kdy vznikly digitální adaptace s

grafickým rozhraním (např. verze pro Nintendo nebo PC). Ty už umožňovaly hru proti umělé inteligenci nebo online soupeřům.

V Československu se Lodě staly populární díky školním sešitům a knížkám s předtištěnými mřížkami, kde děti kreslily své flotily. Po roce 1989 se rozšířily komerční verze s plastovými herními plány a magnetickými figurkami. S nástupem internetu a mobilních aplikací (např. Battleship Online) se hra přizpůsobila modernímu trendu multiplayeru přes síť, doplněná o 3D vizualizace nebo tematické variace (vesmírné lodě, fantasy motivy).

Dnes je Lodě kulturním fenoménem – objevila se ve filmech (Battleship, 2012), seriálech (Přátelé) i jako vzdělávací nástroj pro výuku logiky a pravděpodobnosti. Její jednoduchá pravidla a psychologický prvek (čtení soupeřových tahů) z ní činí nadčasovou klasiku, která přežila přesun od papíru k digitálním platformám. I přes konkurenci komplexních her zůstává symbolem taktického myšlení a rodinné zábavy. [1][2][10]

1.4 Nejznámější taktiky

Hra Lodě, ač zdánlivě jednoduchá, vyžaduje kombinaci logiky, pravděpodobnosti a psychologie. Zde jsou jedny ze známějších strategií pro maximalizaci šancí na výhru. [4]

1.4.1 Checkboard (Šachovnicový vzor)

Hráč střílí pouze na každé druhé pole, vytvářející šachovnicový vzor. Tím maximalizuje šanci zasáhnout loď, protože všechny lodě (kromě délky 1) zabírají minimálně dvě sousední pole. Tato metoda rychle eliminuje velké plochy a zvyšuje pravděpodobnost brzkého zásahu. [4]

1.4.2 Hunt and Target (Hledání a ničení)

Fáze 1 – Hledání: Náhodné nebo systematické střílení po celé ploše, dokud není zasažena část lodi.

Fáze 2 – Ničení: Po zásahu se střílí na sousední pole (vodorovně/svisle), dokud není loď potopena. Tato taktika využívá faktu, že lodě nelze umístit diagonálně. [4]

1.4.3 Edge Avoidance (Vyhýbání se okrajům)

Někteří hráči se zpočátku vyhýbají okrajům herní mřížky, protože menší lodě (např. torpédoborec) se často umisťují blíže ke středu. Tato strategie ale není univerzální – zkušení protivníci mohou okraje úmyslně využívat. [4]

1.4.4 Paritní strategie (Sudá/lichá pole)

Pokročilá metoda, kde hráč využívá matematickou paritu (sudá/lichá pole). Lodě o délce 2+ musí obsadit obě parity, takže po zásahu na sudé pole se cíleně střílí na lichá, což urychluje lokalizaci. [4]

1.4.5 Density Tracking (Sledování hustoty)

Algoritmický přístup, kdy hráč počítá pravděpodobnost výskytu lodi v každém neprobraném poli na základě zbývajících lodních délek a volného místa. Tuto taktiku často používají AI nebo soutěžní hráči. [4]

1.4.6 Salvo Varianta

V některých verzích hráč dostává tolik výstřelů, kolik má zbývajících lodí. Zde se uplatňuje tzv. „rozděl a panuj“ – soustředění střelby do menších oblastí, aby se zvýšila šance na zásah více lodí naráz. [4]

1.4.7 Psychologické hry

V tradiční papírové verzi hráči používají blufování – například předstírání zmatku nebo falešné reakce na soupeřovy zásahy. V digitálních verzích tato taktika mizí, ale zůstává důraz na analýzu soupeřova tempa a vzorců. [4]

1.4.8 Křížová eliminace

Po zásahu lodi se střílí do tvaru kříže (nahoru, dolů, vlevo, vpravo), dokud není určen směr lodi. Jakmile je směr znám, systematicky se ničí zbývající části. [4]

Protiopatření: Zkušení hráči rozmisťují lodě tak, aby nepoužívali běžné užívané taktiky a vzorce – například rozdělují velké lodě na méně předvídatelná místa nebo využívají „lapače“ (malé lodě jako návnady).

Tyto taktiky činí z Lodě nejen hru náhody, ale i strategického myšlení, která zůstává výzvou i po desetiletích existence.

2 Praktická část

2.1 Implementace logiky hry

2.1.1 Třída Gameboard

Třída Gameboard představuje herní plochu o rozměrech 50×50 políček a řeší se v ní, jak akce hráčů ovlivňují hrací pole. Identifikátor každého políčka jsou souřadnice [řádek, sloupec] a může mít následující hodnoty:

- Výchozí/default (když nemá žádnou ze hodnot 1, 2 nebo 3) znamená prázdné políčko, na kterém není žádná část žádné lodě.
- 1 znamená políčko, které je obsazené částí lodě.
- 2 znamená políčko, kde byla loď zasažena.
- 3 znamená políčko, na které hráč vystřelil, ale nebyla na něm žádná část žádné z lodí.

Konstruktor nastaví do dvojrozměrného pole velikost herní plochy a vytvoří list lodí.

```
public Gameboard()
{
    board = new int[GridSize, GridSize];
    ships = new List<Ship>();
}
```

Obrázek 2.1 Konstruktor

Metoda PlaceShip slouží k umístění lodě na herní plán. Zajišťuje, aby loď nebyla umístěna mimo hrací plochu nebo na již obsazené políčko.

Má následující parametry:

- int row: Řádek počátečního políčka lodě.
- int col: Sloupec počátečního políčka lodě.
- int shipSize: Délka lodě (počet políček).
- bool isHorizontal: Orientace lodě (true = vodorovná, false = svislá).

Pokud je některé políčko neplatné (překročení hranic nebo kolize s jinou lodí), metoda ukončí svůj běh bez provedení změn.

Po validaci jsou souřadnice lodě uloženy do 2D pole board jako hodnota 1 a vytvořen nový objekt Ship, který je přidán do listu ships.

```
public void PlaceShip(int row, int col, int shipSize, bool isHorizontal)
{
    List<Point> shipCells = new List<Point>();

    for (int i = 0; i < shipSize; i++)
    {
        int r = isHorizontal ? row : row + i;
        int c = isHorizontal ? col + i : col;

        if (r < GridSize && c < GridSize && board[r, c] == 0)
        {
            shipCells.Add(new Point(r, c));
        }
        else
        {
            return;
        }
    }

    foreach (Point p in shipCells)
    {
        board[p.X, p.Y] = 1;
    }

    ships.Add(new Ship(shipCells));
}
```

Obrázek 2.2 PlaceShip

Metoda CheckHit kontroluje, zda byla loď zasažena.

Metoda ProcessAttack, obsahující parametry řádek a sloupec, vyhodnocuje důsledek útoku na zadané souřadnice a vrací následující hodnoty:

0: Útok do vody, mimo loď (miss).

1: Úspěšný zásah lodě. (hit)

2: Potopení celé lodě. (nastaví metodu isSunk na true)

Pokud je políčko již označeno jako zásah (2) nebo netrefeno (3), vrací odpovídající hodnotu.

Pokud je políčko lodí (1), změní jeho stav na 2 (zásah) a projde všechny lodě v listu ships, aby našel tu, která obsahuje toto políčko.

Pro nalezenou loď zavolá metodu RegisterHit a zkontroluje, zda je loď potopena (metodou IsSunk()).

Pokud je loď potopena, vrátí stav 2, jinak 1.

Pokud útok míří do vody (0), políčko se označí že se netrefil (3).

Metoda RenderBoard využívá knihovnu System.Drawing k vizualizaci herního pole.

Parametry: Objekt Graphics [3] pro vykreslování, offsetX, offsetY a velikost políčka (cellSize).

Barevné rozlišení:

- Modrá (Color.Blue) znamená prázdné pole vodu.
- Zelená (Color.Green) pro lodě.
- Červená (Color.Red) pro zásahy.
- Šedá (Color.Gray) pro pole, na které bylo vystřeleno ale ta střela netrefila loď.

Pro každé políčko se vykreslí obdélník příslušné barvy a černě ohraničený rámeček.

Metoda GetSunkShipsCount projde celý list lodí a zjistí, kolik jich je dohromady potopeno, tuto hodnotu vrací.

2.1.2 Třída Ship

Třída Ship je třída vnořená do třídy Gameboard a slouží k zapamatování a zjišťování pozic lodí a jejich poškození. V konstruktoru se vytvoří list políček (cells) a množina zásahů (hits).

Metoda ContainsCell(int row, int col) prochází seznam cells a kontroluje, zda loď obsahuje políčko na zadaných souřadnicích. Pokud ano, vrací true.

Metoda RegisterHit(int row, int col) přidá souřadnice zásahu do množiny hits, ale pouze pokud políčko patří lodi (ověřeno pomocí ContainsCell).

Metoda IsSunk() porovnává počet zásahů (hits.Count) s celkovým počtem políček lodě (cells.Count). Pokud se rovnají, loď je považována za potopenou a metoda vrací true.

Při umístění lodě (PlaceShip) vytvoří Gameboard nový objekt Ship a uloží jeho pozice.

Při útoku (ProcessAttack) Gameboard iteruje přes všechny lodě v ships, hledá tu, která obsahuje zasažené políčko, a aktualizuje její stav.

Detekce potopení lodě (IsSunk) porovná počet zasažených políček a celkovou délku dané lodě a pokud se rovnají, vrátí true (byla pototpena), pokud se nerovnají vrátí false (nebyla potopena).

2.2 Komunikace přes síť

Komunikace mezi klientem a serverem využívá protokol TCP/IP a třídy z namespace System.Net.Sockets. Pro komunikaci ze serverové strany se nejprve server inicializuje tím, že vytvoří socket serverSocket a naváže jej na port 5555 a čeká na připojení klienta, viz Obrázek 2.3 Inicializace serveru

```
serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
serverSocket.Bind(new IPEndPoint(IPAddress.Any, 5555));
serverSocket.Listen(1);
labelStatus.Text = "Waiting for connection...";
```

Obrázek 2.3 Inicializace serveru

Po připojení klienta se spouští vlákno receiveThread(), která se nastaví do pozadí.

Metoda RecieveData() poté v nekonečné smyčce kontroluje příchozí sockety (zprávy) od klienta.

Tyto zprávy jsou zpracovávány v metodě ProcessGameData(), která aktualizuje herní pole a uživatelské rozhraní.

Ze strany klienta pro připojení k serveru musí uživatel zadat IP adresu a port do textBoxů tbIPAddress a tbPort.

Po stisknutí tlačítka Connect se inicializuje socket a naváže spojení se serverem, viz Obrázek 2.4 Připojení klienta

```
Počet odkazů: 1
private void BtnConnect_Click(object sender, EventArgs e)
{
    string ipAddress = tbIPAddress.Text;
    if (!int.TryParse(tbPort.Text, out int port))
    {
        MessageBox.Show("Invalid port number");
        return;
    }

    clientSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    try
    {
        clientSocket.Connect(ipAddress, port);
        MessageBox.Show("Connected successfully");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Connection failed: " + ex.Message);
    }
}
```

Obrázek 2.4 Připojení klienta

Pro kontrolu odpovědí je tady Timer responseTimer, který každých 100 milisekund kontroluje, zda jsou data k dispozici. Příchozí zprávy jsou dekódovány a předány metodě ProcessGameData().

Zprávy jsou přenášeny mezi klientem a serverem jako řetězce ve formátu příkaz,souřadnice. Po přijetí se tento řetězec rozdělen pomocí metody Split(), která rozdělí string (řetězec) podle znaku ‘,’ na pole string[] parts, kde podle na pozici 0 je příkaz pro útok (Attack), nebo pro položení lodě (PlaceShip), který uživatel vybral z radio buttonů (tlačítka k zaškrtnutí, kde jde vybrat pouze jedno) a na pozicích 1 a 2 v tomto poli se nachází souřadnice (na pozici 1 je řádek a na pozici 2 je sloupec), na které se má vystřelit/položit loď. Při pokládání se v horizontální orientaci položí loď, pokud má více jak jedno políčko, od kliknutého pole směrem doprava. Při vertikální orientaci se položí od klinutého pole směrem dolů.

Příklady:

Umístění lodě: "PlaceShip,5,10,3,true" (řádek 5, sloupec 10, délka 3, horizontální orientace).

Útok: "Attack,7,12" (útok na řádek 7, sloupec 12).

Odpověď serveru: "Hit,7,12", "Miss,7,12", "ShipDestroyed,7,12".

Synchronizace tahů

Řízení pomocí isMyTurn:

Server začíná hru s isMyTurn = true a po svém tahu nastaví isMyTurn = false, čímž umožní klientovi provést tah.

Klient po odeslání útoku (SendMessageToServer("Attack,...")) nastaví isMyTurn = false a čeká na odpověď serveru.

Blokování neplatných akcí:

Pokud hráč klikne na protihráčovo pole, když není jeho tah, zobrazí se varování “It is not your turn“ (nejste na tahu), viz Obrázek 2.5 Kontrola, kdo je na tahu:

```
if (!isMyTurn)
{
    MessageBox.Show("It is not your turn!");
    return;
}
row = (boardY - opponentBoardOffsetY) / cellSize;
col = (boardX - opponentBoardOffsetX) / cellSize;

if (rbAttack.Checked)
{
    SendMessageToServer($"Attack,{row},{col}");
    isMyTurn = false;
}
```

Obrázek 2.5 Kontrola, kdo je na tahu.

Přerušení komunikace je ošetřeno bloky try-catch, které zobrazí chybové hlášení (např. SocketException).

Pro obnovení spojení může klient znovu použít tlačítko Connect pro opětovné připojení.

2.3 Uživatelské rozhraní

Vlastní plocha (Your Board) zobrazuje pozice hráčových lodí, zásahy a miny. Políčka s loděmi jsou vykreslena zeleně (Color.Green), zásahy naopak červeně (Color.Red), střely do prázdna (když netrefí loď) šedě (Color.Gray). Umístění lodí probíhá v režimu Place Ship.

Protihráčova plocha (Opponent's Board) zobrazuje pouze výsledky útoků (zásahy a miny). Lodě protihráče jsou skryty. Útoky se provádějí v režimu Attack.

2.3.1 Technické parametry:

Velikost políčka: 15×15 pixelů.

Rozměr herní mřížky: 50×50 políček (celkem 750×750 px).

Pozice ploch:

Vlastní plocha: playerBoardOffsetX = 20, playerBoardOffsetY = 200 (klient) / 220 (server).

Protihráčova plocha: opponentBoardOffsetX = 900, opponentBoardOffsetY = 200 (klient) / 220 (server).

2.3.2 Ovládací prvky

Radio buttony:

- rbPlaceShip: Umožňuje režim umisťování lodí.
- rbAttack: Aktivuje režim útoku na protihráče.

Nastavení lodí:

- NumericUpDown (rozsah 1–5): Volba délky lodě.
- ComboBox: Výběr orientace lodě (horizontální/vertikální).

Síťové připojení (pouze klient):

- tbIPAddress: Textové pole pro zadání IP adresy serveru.
- tbPort: Textové pole pro port (např. 5555, sever má jako výchozí port nastaven 5555).
- btnConnect: Tlačítko pro navázání spojení se serverem.
- labelResponse: Zobrazuje odpovědi serveru (např. Hit,7,12).
- labelStatus: Indikuje stav spojení (např. pokud je klient připojen tak vypíše "Client connected").

2.3.3 Interakce s uživatelem

Pro umisťování lodí (rbPlaceShip) hráč vybere délku lodě (NumericUpDown), orientaci (ComboBox) a klikne na své herní ploše.

Pokud je pozice platná, loď se vykreslí jako zelená políčka.

Neplatné umístění (překrytí lodí, překročení hranic) je automaticky ignorováno.

Pro útočení (rbAttack) hráč klikne na protihráčovu plochu, čímž odešle souřadnice na které zaútočí.

Server/klient zpracuje útok a vrátí výsledek (Hit, Miss, ShipDestroyed).

Uživatelské rozhraní se poté aktualizuje protihráčovu plochu na základě odpovědi.

Po zadání IP a portu hráč stiskne tlačítko Connect, které inicializuje socket.

Neplatný port nebo nedostupný server vyvolá chybové hlášení.

Vykreslování a aktualizace se řeší pomocí metody RenderBoard, která vykreslí herní plochu pomocí System.Drawing.Graphics. [3]

Metoda InvalidateCell(), optimalizuje program tím že překreslí pouze konkrétní políčko, ne celou plochu.

Úspěšný zásah nastaví barvu políčka na červenou a při potopení lodě hlášení v MessageBoxu "You destroyed an enemy ship!".

Když se hráč netrefí barva políčka se nastaví na šedou.

Závěr

Tato maturitní práce měla za cíl vytvořit síťovou verzi hry Lodě pro dva hráče s využitím technologií C#, .NET Framework a TCP/IP protokolu. Cíle byly splněny: aplikace umožňuje rozmísťování lodí na ploše 50×50 políček, střídání tahů s okamžitou zpětnou vazbou a detekci konce hry po potopení všech lodí. Síťová komunikace byla implementována prostřednictvím architektury klient-server, která zajišťuje stabilní spojení i na lokální síti. Teoretická část shrnula historii hry, její strategické varianty a popsala klíčové technologie včetně principů TCP/IP a objektově orientovaného programování v C#. Praktická část pak prokázala, že i s omezenými zdroji lze vytvořit funkční produkt, byť s určitými kompromisy v uživatelském rozhraní a výkonu.

Původní záměr – vytvořit plně funkční síťovou hru – byl naplněn. Nicméně některé aspekty zůstaly v základní podobě. Například grafické rozhraní, postavené na System.Drawing, sice splňuje účel, ale nedosahuje úrovně komerčních her s moderními vizuálními efekty. Rozměr herní mřížky (50×50) byl zvolen jako unikátní prvek oproti standardnímu formátu 10×10, což sice rozšiřuje taktické možnosti, ale zároveň zvyšuje nároky na přehlednost a optimalizaci vykreslování. Zatímco literatura (např. Wikipedia nebo didaktické zdroje) poskytla dostatek informací o pravidlech hry a formálních náležitostech práce, detailní technické postupy (jako implementace socketů v C#) vycházely spíše z experimentování a odborných fór, což ukazuje na potřebu hlubšího čerpání z technických manuálů nebo specializovaných publikací.

V porovnání s existujícími řešeními (např. komerční hra *Battleship Online*) tato práce nabízí méně funkcí, jako např. absence AI protihráče. Na druhou stranu přináší rozšířenou herní plochu, která mění dynamiku hry – větší prostor umožňuje komplexnější taktiky, ale zároveň prodlužuje průběh hry. Zatímco tradiční papírové a digitální verze Lodí spoléhají na menší mřížku a rychlejší hratelnost, zdejší řešení cílí na náročnější hráče. Síťová komunikace přes TCP/IP je spolehlivá, ale oproti moderním řešením (WebSocket, UDP pro rychlejší hry) postrádá optimalizaci pro vysokou latenci. Výsledky práce mohou sloužit jako vzdělávací materiál pro studenty informačních technologií – demonstrují práci se síťovými protokoly, objektově orientovaným designem a uživatelským rozhraním v C#.

Seznam použitých zdrojů

[1] Lodě. Online. In: WIKIMEDIA FOUNDATION, INC. Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 16.9.2024. Dostupné z: https://cs.wikipedia.org/wiki/Lod%C4%9B. [cit. 2025-03-27].

[2] SICARD, Sarah. *Where the 'Battleship' board game originated*. Online. MILITARY TIMES. 29.11.2023. Dostupné z: https://www.militarytimes.com/off-duty/military-culture/2023/11/29/where-the-battleship-board-game-originated/. [cit. 2025-03-28].

[3] *Graphics Třída*. Online. MICROSOFT. 2025. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/api/system.drawing.graphics?view=windowsdesktop-9.0. [cit. 2025-03-28].

[4] FRANCO, JC. *How to Win at Battleship: 15 Tips, Tricks & Strategies (Boost Your Win Rate)*. Online. GAMESVER. 25.1.2024. Dostupné z: https://www.gamesver.com/how-to-win-at-battleship-tips-tricks-strategies-boost-your-win-rate/. [cit. 2025-03-28].

[5] *Visual Studio 2022*. Online. MICROSOFT. 2025. Dostupné z: https://visualstudio.microsoft.com/cs/vs/. [cit. 2025-03-28].

[6] *TCP overview*. Online. MICROSOFT. 17.4.2024. Dostupné z: https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/sockets/tcp-classes. [cit. 2025-03-28].

[7] *Obecná struktura programu jazyka C#*. Online. MICROSOFT. 29.1.2025. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/fundamentals/program-structure/. [cit. 2025-03-28].

[8] *Úvod do .NET*. Online. MICROSOFT. 16.3.2025. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/core/introduction. [cit. 2025-03-28].

[9] *About GitHub and Git*. Online. GITHUB. 2025. Dostupné z: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git. [cit. 2025-03-28].

[10] FULLEYLOVE, Rebecca. *The Board Games Everyone Loved in the 20th Century*. Online. THE STRONG NATIONAL MUSEUM OF PLAY. 2025. Dostupné z: https://artsandculture.google.com/story/the-board-games-everyone-loved-in-the-20th-century/7AXBAJCg4AwRLQ. [cit. 2025-03-28].

[11] *Client-Server Model*. Online. SANCHHAJA EDUCATION PRIVATE LIMITED. GeeksforGeeks. 3.1.2025. Dostupné z: https://www.geeksforgeeks.org/client-server-model/. [cit. 2025-03-28].

Seznam použitých symbolů a zkratek

Zkratka	Význam
IDE	Integrated developer environment (Integrované vývojové prostředí)
CLR	Common language runtime (běhové prostředí)
ARP	Address resolution protokol
IoT	Internet of Things

Seznam obrázků

Obrázek 1.1 Rozehraná hra lodě [1] 14
Obrázek 2.1 Konstruktor 17
Obrázek 2.2 PlaceShip 18
Obrázek 2.3 Inicializace serveru 20
Obrázek 2.4 Připojení klienta 21
Obrázek 2.5 Kontrola, kdo je na tahu. 22