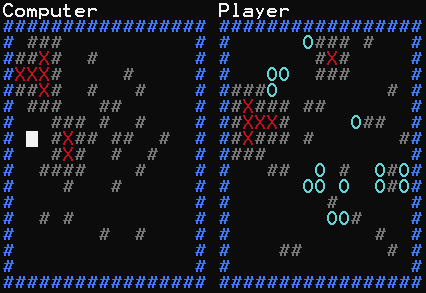
Zápočtový program

**Hra Lodě**

****Autor: Oleg Petruny  
Předmět: NPRG038  
Semestr: letní  
Jazyk: C#

**Obsah**

[Specifikace stručně 2](#_Toc145573216)

[Specifikace úplná 2](#_Toc145573217)

[Průběh hry 2](#_Toc145573218)

[Engine 3](#_Toc145573219)

[Zpracování ovládání 3](#_Toc145573220)

[Scény 3](#_Toc145573221)

[Grafika 4](#_Toc145573222)

[Síťování 4](#_Toc145573223)

[Průběh práce 5](#_Toc145573224)

[Co nebylo doděláno 5](#_Toc145573225)

[Závěr 5](#_Toc145573226)

# Specifikace stručně

Hra mořský boj na vlastním konzolovém enginu v С#. Má barevný a zvukový výstup, ovládá se pomocí klávesnice, a podporuje síťování.

# Specifikace úplná

Program je hra „mořský boj“ v příkazovém rozhraní. Hrací plochy mohou být libovolné velikosti a zároveň i objekty, taky „lodě“, mohou být libovolné a být náhodně umístěné na ploše. Cílem hry je zničit lodě všech soupeřů dřív, než dojdou vlastní.

Hra také podporuje více režimů. Singleplayer (sám se sebou), Computer (proti počítači), Multiplayer (hra více hráčů v jednom okně) a Client a Server (hra více hráčů přes internet). Hra podporuje nejvíce 5 hráčů, přičemž toto omezení je pouze designově.

Celá hra je postavená na enginu, ve kterém lze jednoduše vytvářet další libovolné konzolové hry. Engine obsahuje podsystémy pro:

* práci s výstupem konzole, tedy možnost barevného výstupu, nastavení fontů a rozlišení okna,
* zvukový výstup,
* obsluhu vstupu klávesnice,
* tvorbu jednotlivých scén,
* tvorbu síťového připojení client-server.

Hra také obsahuje nastavení buď v hlavním menu, před začátkem hry, nebo ve zdrojovém kódu (třída Shared), které jsou sdílené napříč celou aplikaci.

Samotný kód je OS independent, ale oficiálně hra podporuje pouze Windows.

# Průběh hry

Při spuštění nové hry se nejdřív zobrazí nastavení preferencí, kde lze zvolit režim hry a velikost hrací plochy. V případně že je vybrána lokální hra pro více hráčů, lze určit jejich počet. Po schválení preferencí ihned začíná buď fáze hry, nebo výběr rozložení lodí. Pro jednoduchost ovládání, lodě nelze umisťovat ručně, ale pouze náhodně přegenerovat celé rozložení. Taky při zásahu častí lodě, hráč pokračuje v útoku.

Singplayer se hraje pouze na jedné ploše, kterou hráč nemůže vybrat a nevidí lodě na ní. Cílem je zničit všechny lodě na co nejmenší počet zásahů.

Computer obsahuje již umělou inteligenci, proti které živí hráč musí soupeřit. Designově omezeno pouze na jeden počítač a jednoho hráče, ale s triviálními úpravy logiky lze povolit libovolný počet hráčů.

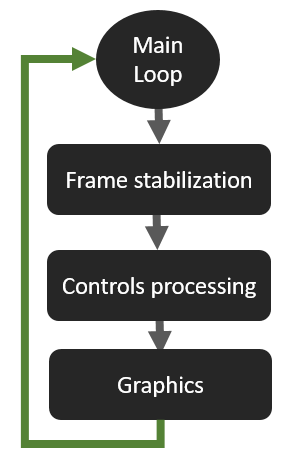
Multiplayer popořadě povolí každému hráči vybrat jeho rozložení. Je doporučeno, aby při výběru rozložení, ostatní hráče se nedívali. Po ukončení tvorby rozložení, začíná fáze útoku. Každý hráč popořadě utočí na každou plochu, která ještě obsahuje lodě. Útočit mohou také mrtvý hráče.

Server a Client funguje přibližně stejně jako Multiplayer. Pro použití vzdálené hry, je potřeba, aby byl povolen port 25555. Tyto režimy jsou provedeny tak, aby pouze Server měl kontrolu nad celou hrou. Klient pouze posílá dotazy nebo aplikuje vizuální aktualizace.

Server vytvoří lobby pro připojení jiných hráčů. Pro připojení ostatních hráčů, server musí sdělit „invite key“. Až všechny hráče budou hotovy, server může startovat hru.

Client pro připojení musí zadat ip adresu serveru a jeho „invite key“. Až se připojí, může vybírat svoje rozložení a taky dat najevo, že je připraven k začátku hry.

# Engine

Program začíná funkcí **Main()**, kde se nejdřív inicializuje grafika a poté se vstoupí do hlavního cyklu, který je de facto srdce celého enginu.

První část cyklu se stará o stabilizaci snímků, tedy vnucuje spánek hlavního vlákna. Napomáhá to libovolné logice být navázanou na počet snímku v sekundu, protože je zaručeno, že jich nebude víc, než je specifikováno v **Shared.Env.FpsLimit** a budou mít přibližně stejný interval. Doporučená největší hodnota je 60, protože odezva ovládání a grafiky s větším FPS nezlepšují se pro hráče lineárně ale logaritmicky.

Jako další v cyklu se zpracuje vstup a poté se vypočítá a vytiskne grafika.

Pro ukončení cyklu lze použít **Shared.Func.ExitApplication()**. Při havarování, engine nenastartuje znovu schválně, aby byla zobrazena chyba.

# Zpracování ovládání

Ovládání je vyřešeno celkem triviálním způsobem. Nejdřív někde ve hře (v Scéně) se naplní slovník **<Klávesa, Akce>,** a potom se projde každý klíč, zda-li je tato klávesa zmačknuta. V případě, že klávesa spouští funkci, která způsobuje silné změny (třeba přepínaní na novou Scénu), lze určit minimální delay mezi každým ohlášením zmačknutí.

V libovolný okamžik ve hře lze vymazat všechny napojené klávesy a akce. To způsobí, že se vynechá celé zpracování v aktuálním nebo příštím cyklu, což ale není hráčem citelný.

Zároveň není potřeba hlídat více vláknový přístup, protože častí hry, které zpracovávají a modifikují slovník běží pouze v jednom vlákně za sebou, tedy žádný konflikt nenastává.

# Scény

Scény umožňují rozložení hry do více kousků. Tak jsou třeba scény „Hlavní menu“, „Nastavení“ a „Hra“. Scéna dostane plnou kontrolu na celým grafickém bufferem a je pouze na ní, aby ho správně použila. Základní abstraktní třída obsahuje statické funkce pro změnu scén a zároveň aktuální scénu. Samotné přepínání každá scéna si může libovolně upravit. Taky každá scéna si může předávat data pomoci objektu **AfterchangeData**, ale zase je na ní, jak si s těmi daty poradí.

# Grafika

Hotový výstup se vykreslí pomocí statické **Update()** a samotný výstup lze zmodifikovat kdykoli a kdekoli, protože grafický buffer je statická public instance. Před vykreslením ale musí proběhnout značné množství výpočtů.

Jako první se v metodě **Bake()** popořadě slepí všechny vrstvy dohromady. Tady se hodně pracuje s ořezáváním stringů a tvorbou nových instancí **Sixel**. Bohužel pokud okno nepřesahuje aspoň 128x128 znaků (což zatím ve hře není), paralelní výpočty jednotlivých řádků pouze zpomalují proces.

Poté jestli je nastaven **Shared.Env.PrintSameBuffer = false**, se vyzkouší, jestli náhodou minulý buffer a nový nejsou stejný. Pokud ano, další zpracování se zahodí a tisknout se do konzole nebude. Pokud ne, nový buffer se uchová a výpočty pokračují.

Následovně pokud **Shared.Env.SixelCompression = true,** se sloučí Sixely stejné barvy vedle sebe.

A až po tom všem se začne tisknout do konzolového okna. Tady je potřeba si vzít na vědomi, že se nebude tisknout úplně celý nový konzolový buffer, ale pouze upravovat stávající posouváním kurzoru.

Celkově tedy nikde nelze aplikovat paralelní zpracování, protože data jsou relativně malá. Reálnou překážkou je pomalý konzolový výpis, který je potřeba co nejvíc omezovat (přibližně 80ms jeden výpis).

Pokud aplikace je builděná pro windows, grafika má také možností i pro ovládání okna. Lze tak nastavovat rozlišení nebo fonty. Změna rozlišení za běhu není vždy bezpečná a tajemství se kryje někde uvnitř samotné windows konzole. Zároveň není vše tak jednoduché s fonty. Grafika sice umí nahrát libovolný font do paměti, ale aby nahraný font se mohl použít v konzoli, musí být navíc zapsaný v registrech systému. Proto lze pouze nastavovat fonty, které již existují v konzoli uživatele. A taky občas se může stát, že každý uživatel má jiné předinstalované fonty, proto grafika u každého se může lišit a častěji tím špatným směrem.

# Síťování

Pro síťování jsou k dispozici dvě třidy Server a Client. Logicky je server pouze hub pro sdílení předdefinované informace mezi klienty. Client se stará pouze o připojení k serveru, posílání požadavků na server nebo obdržení zpráv od serveru. Požadavky a odpovědi většinou na sebe nenavazují, tedy provoz je neřízený oboustranný, a pouze ověřovaný na straně serveru.

Před použitím se musí vytvořit určitá instance pomocí statické metody **Create()**. Je potřeba si dávat pozor, že je možné vytvořit pouze jednu instanci každého typu v celém programu. Pokud vytvoření proběhlo úspěšně, instanci lze uvést do provozu pomoci **Start()**. Pro zastavení instance lze použit **Stop()**. A před vytvořením nové, je potřeba použít **Dispose()** na staré instanci, případně počkat než instance bude odstraněná GC.

Client začíná přímím připojením na server, které pokud skončí úspěšně, pokračuje nezávislým čekáním na zprávy od serveru a čekáním na data pro odesílání.

Server začíná posloucháním nových připojení, kde po úspěšné autorizaci klienta, se vytvoří vlákna pro nezávislé poslouchání/posílání zprav od/pro klienta.

Je potřeba brát v úvahu, že v síťování většina funkcí má vlastní vlákno a je potřeba obstarat bezpečný průtok dat. Pro funkce očekávající data k odesílání, je očekáváno použiti kolekce příbuzné k **BlockingCollection**. Pro představu možné implementace viz. **NetManager.cs**.

Připojení je navíc zkomplikované ověřováním. I server i klient musí při startu dostat stejný „invite key“, který se použije k autorizaci. Po autorizaci klient dostane nový klíč pro komunikaci a jeho aktuální UID. Celé síťování je připravené pro běh v šifrovaném provozu jak pro vnější síťovou komunikaci, tak vnitřní herní. Bohužel nebyl zprovozněn derivovácí AES algoritmus pro šifrování, proto jsou posílané pouze čisté JSON objekty.

Každá funkce je připravená k havarování na libovolnou událost. Ovšem pokud se jedná o síťovou chybu, ta je potom dále předzpracovaná v jednoduchou k pochopení uživatelem chybu.

# Průběh práce

Dohromady celá práce zabrala přibližně 120h v průběhu 3 měsíců. Hodně často problémy vznikaly v nejvíce nečekaném místě. Lze ale zvýraznit 3 nejtěžší.

První se týká nastavovaní fontů, které jsou zprovozněny, přes staré Céčkové funkce v kernel32.dll. Je netriviální najít dokumentaci pro tyto funkce, nebo jak používat Céčkové struktury v C#. Taky hodně funkcí během vývoje Windows zanikly nebo byli modifikované, proto to byl velký pokus-omyl.

Druhý souvisel s implementováním online logiky do hry. Předtím hra vůbec nebyla na to připravena, proto se musela přepsat veškerá logika znova.

Třetí byl zprovoznění šifrování, kde mi jednoduše chyběla spousta dovednosti v samotném šifrování.

# Co nebylo doděláno

Jediná domluvená a nesplněná věc, je přehrávaní audia z konzole. Bylo rozhodnuto použit veškerý čas na zdokonalení a odladění online hry, protože má mnohem značnější vliv na herní zážitek.

Taky v průběhu byl ponechán šifrovaný síťový provoz, z důvodu uvedených výše.

Někde okrajově se taky plánovalo run-time ošetření rozlišení okna. To by ale vyžadovalo úpravy grafiky, které by způsobily značné zpomalení vykreslování. Celé vykreslování bohužel je velmi omezené konzoli.

# Závěr

Samotné téma/kostra programu je celkem jednoduché. Ale díky záměru na univerzalitu se značně a komplexně rozšiřuje. Byla potřeba mít zkušeností práce s <windows.h>, optimalizacemi, vlákny a sítě. To vše ale umožnilo vytvořit nejen hezkou a malou hru, ale taky engine umožňující tvorbu mnoha dalších konzolových her, které jíž budou mít značně kratší dobu vývoje.