Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Vývoj aplikace v jazyce C#
-
Desková hra Lovci noci

Bc. Jakub Riedl

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jakub Riedl

Informatika

Název práce

Vývoj aplikace v jazyce C# – Desková hra Lovci Noci

Název anglicky

Application Development in C # – Board game Shadow hunters

Cíle práce

Cílem práce je převedení pravidel a algoritmů deskové hry Lovci noci do podoby počítačové aplikace, aby bylo umožněno jednomu lidskému hráči hrát proti třem počítačem řízeným protihráčům. Protihráči se budou chovat podle předem připravených algoritmů daných pravidly hry.

Metodika

Práce sestává ze dvou částí – teoretické a praktické. Metodika zpracování teoretické části práce vychází ze studia odborné literatury. Na základě zjištěných poznatků budou popsána teoretická východiska práce.

Praktická část práce bude spočívat v analýze, návrhu a implementaci aplikace umožňující hrát počítačovou dobu deskové hry Lovci noci v provedení pro jednoho lidského hráče a 3 počítačové protivníky. Proces implementace bude popsán, výsledná aplikace bude otestována a budou navrženy možnosti jejího dalšího možného rozvoje.

Doporučený rozsah práce

60-80 stran

Klíčová slova

C#, desková hra, Lovci noci, skriptovaná AI, Visual Studio 2017

Doporučené zdroje informací

Běhálek, Marek. Programovací jazyk C#. Programovací jazyk C#. [online].

KOLÁŘOVÁ, Marie a Jiří ELIŠKA. Desková hra. 2010.

KRAČMAR, S. – VOGEL, J. – ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE. STROJNÍ FAKULTA. *Programovací jazyk C*. Praha: Vydavatelství ČVUT, 1995. ISBN 80-01-01375-8.

NOWAK, Stanislav. Formální popis deskových her. Praha, 2011. Bakalářská práce. Univerzita Karlova v Praze. Vedoucí práce Prof. RNDr. Petr Štěpánek, DrSc.

Petzold, Charle. Programování Microsoft Windows Forms v jazyce C#, Computer Press a.s., 2006, ISBN: 80-251-1058-3

Předběžný termín obhajoby

2018/19 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 26. 3. 2019

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 26. 3. 2019

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 27. 03. 2019

Čestné prohlášení

Prohlašuji, že svou diplomovou práci " Vývoj aplikace v jazyce C# - Desková hra Lovci noci" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob. Veškeré citace jsou zpracovány dle normy ISO 690:2011.

V Praze dne 25.3.2019 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu diplomové práce panu Ing. Jiřímu Brožkovi, Ph. D., za vedení a cenné rady při vypracování diplomové práce.

Taktéž děkuji dalším osobám: Mgr. Aleně Vávrové a firmě Mindok s.r.o.

Vývoj aplikace v jazyce C# - Desková hra Lovci noci

Souhrn

Diplomová práce vznikla za účelem vytvoření aplikace, která by umožnila uživateli zahrát si deskovou hru Lovci noci podle pravidel hry. Cílem bylo vytvořit algoritmus jak pro uživatele tak i pro počítačem řízené hráče (AI).

Pro vlastní tvorbu aplikace bylo použito prostředí Visual Studio 2017 umožňující programování v jazyce C#.

V teoretické části diplomové práce je přiblížen pojem deskové hry, jejich stručná historie, současný stav a užití různých deskových her. Dále je vysvětlen pojem teorie her, strategie a jejich volba, hra s inteligentním hráčem a sedlový bod. Je přiblíženo vývojové prostředí Visual Studia 2017 a pojem umělá inteligence.

Praktická část diplomové práce se týká pravidel hry Lovci noci použitých v aplikaci, strategií jednotlivých ras vystupujících ve hře, herních postav a jejich osobních herních strategiích a herních karet. Práce dále obsahuje komentovaný kód aplikace samotné.

Klíčová slova:C#, desková hra, Lovci noci, skriptovaná AI, Visual Studio 2017

Application Development in C
-
Board game Shadow hunters

Summary

The thesis was created in order to create an application that would allow the user to play the board game Shadow hunters according to the rules of the game. The aim was to create an algorithm for both users and computer controlled players (AI).

The Visual Studio 2017 environment, which enables C # programming, was used to create the application.

The theoretical part of the thesis describes the concept of board games, their brief history and the current state and use of various board games. Next is explained the concept of game theory, strategy and their choice, game with intelligent player and saddle point. The development environment of Visual Studio 2017 and the concept of artificial intelligence are presented.

In the practical part of the thesis is concerned with the rules of the game Night Hunters used in the application, strategies of individual races performing in the game, game characters and their personal game strategies and game cards. The work also contains an annotated application code.

Keywords: board game, Shadow hunters, C#, script AI, Visual Studio 2017

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Hry provázejí člověka od samotných počátků. Fungovaly jako proces výuky, náhražka reálného boje, důkaz strategického myšlení, převahy nad soupeřem, i pouze jen jako možnost ukrácení dlouhé chvíle.

V dnešní době se termín „hry“ často pojí, hlavně u mladé generace, pouze s hrami počítačovými, a to i přes to, že aktuálně je opravdu velké množství her deskových. I proto jsem se rozhodl vypracovat v diplomové práci převod právě jedné takovéto nové deskové hry do počítačové podoby. Práce počítá s tím, že uživatel je s deskovou hrou Lovci noci předem seznámem, důvěrně zná její pravidla a mechaniky a tudíž výsledná aplikace pro něj bude sloužit jako tréninková.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem diplomové je převedení pravidel a algoritmů deskové hry Lovci noci do podoby počítačové aplikace, aby bylo umožněno jednomu lidskému hráči hrát proti třem počítačem řízeným protihráčům. Protihráči se budou chovat podle předem nadefinovaných algoritmů daných pravidly hry.

Výsledná aplikace bude okení aplikací pro OS Windows, která bude vytvořená v prostředi Visual Studia 2017 za použití jazyka C#.

2.2 Metodika

Práce sestává ze dvou částí – teoretické a praktické. Metodika zpracování teoretické části práce vychází ze studia odborné literatury. Na základě zjištěných poznatků budou popsána teoretická východiska práce.

Praktická část práce bude spočívat v analýze, návrhu a implementaci aplikace umožňující hrát počítačovou dobu deskové hry Lovci noci v provedení pro jednoho lidského hráče a 3 počítačové protivníky. Proces implementace bude popsán, výsledná aplikace bude otestována a budou navrženy možnosti jejího dalšího možného rozvoje.

3 Teoretická východiska

3.1 Deskové hry

Hry provázejí lidstvo do nepaměti a jsou součástí snad všech civilizací. Samotná historie deskových her se datuje přibližně do doby 3500 let před naším letopočtem. Nejstarší deskové hry, které byly objeveny, pocházejí z archeologických vykopávek v údolí Nilu. I z dalších epoch je známo mnoho vyobrazení zachycujících hráče sedící u herní desky. Jedná se například o etruskou vázu, nejspíše s hrdiny krátícími si čas před dobytím Troje něčím na desce – dáma to prý nebyla, která je uložena ve vatikánském muzeu. Obrázek z této vázy používá jako svůj symbol pražské Deskohraní, dříve nazývané Olympiáda duševních sportů.

Počátek 19. století znamená pro deskové hry velký úpadek. A to nejen mezi dětmi, ale hlavně mezi dospělými hráči. Ve společnosti převládá myšlenka, že lidstvo konečně dospělo, a tak nemůže ztrácet čas hraním her. Kariéra a vydělávání peněz se stává pro lidi důležitější. V této době jediné hry, které slaví úspěch, jsou hry, které přinášejí zisk. A jediní, kdo mají na hry čas, jsou lidé, kteří mají peněz dostatek. Díky tomu se do popředí dostávají hry hazardní a s nimi i velký rozmach kasin. A deskové hry? To jsou přeci hry pro děti.

Na konci 20. století naštěstí zažívají deskové hry svou renesanci a nový rozkvět. Jsou označovány jako moderní společenské hry, a to i přes to, že některé vycházejí z velmi starých kořenů, což jim ale neubírá nic na atraktivitě. Opět k hrací desce usedají i dospělí, leckde i celé rodiny. Bohužel skoro dvě stě let útlumu deskových her v lidech zanechalo své negativní stopy. Ať už je hra sebe víc chytlavá a zábavná, vždy je znevážená ve srovnání se skutečnou prací. Z doby útlumu se uchytilo i několik nepěkných výrazů. Vždyť jak pohrdavě zní „ty jsi hračička“ nebo napomenutí „nech toho hraní!“. Stejně tak děti jdoucí do puberty mají k hrám často odpor. Cítí se povzneseny nad dětskou zábavu, kterou v jejich očích hraní deskové hry je.

V dnešní době se objevuje stále více rozličných deskových her. Ať již hry, kde hráči hrají proti sobě, to je například i hra, která je tématem této práce, Lovci noci, tak hry, kde hráči hrají spolu proti hře samotné, příkladem může být THE GAME: Hraj, dokud můžeš! od firmy Mindok.

Kromě výše zmíněných her existují i hry s postupně se zvyšující se obtížností, které hráče učí jak hrou postupovat. Tento typ deskových her se nazývá Smart Game či Chytré hry. Mimo to se některé deskové hry začaly používat i jako jedna z pomůcek učitelů při výuce. Příkladem zde může být seminář Škola hrou pořádaný na ZŠ Karla Čapka či přednášky Mgr. Aleny Vávrové, které organizuje Národní institut dalšího vzdělávání.

Deskové hry se v dnešní době využívají tedy nejen pro relaxaci, ale i jako vhodná a zábavná pomůcka pro výuku různých předmětů, ať již na základních a středních školách nebo třeba v kybernetice, respektive při vývoji umělé inteligence. (1) (2) (3) (4)

3.1.1 **GO**

Vznik této čínské deskové hry se odhaduje 2 až 3 tisíce let před naším letopočtem a nejstarší písemní zmínka o ní pochází z 6. století před naším letopočtem. Během 7. a 8. století se hra Go spolu s čínským uměním postupně dostává do Japonska, kde si ji osvojují převážně buddhističtí mniši a samurajové. Později se Go dostává do přízně japonských vládců a na začátku 17. století vznikají čtyři školy Go, které jsou podporované samotnou vládou. Toto období je pro hru Go označováno za jeho zlatou éru.

Na konci 19. století ale školy postupně zanikají, hra samotná ovšem neupadá v zapomnění. Na začátku 20. století (r. 1924) vzniká Japonská asociace Go, která organizuje první profesionální zápasy. Druhá polovina 20. století se nese ve znamení šíření profesionální hry Go do Koreje a zpět do domovské Číny, stejně tak, ale v menší míře, i do dalších východoasijských zemí. Právě korejští hráči dominují profesionální scéně, a to od druhé poloviny 90. let 20. století.

Mimo Asii se tato hra rozšířila až na konci 19. století. Kolem roku 1850 ji přivezli čínští přistěhovalci do Spojených států amerických (USA), ale více méně neopustila jejich komunitu. V Německu se, díky článku a později knize Hra go od pana Korschelta, dostává Go do povědomí na konci 19. století. A na našem území byli hráči Go již před první světovou válkou. Dokládají to záznamy o předplatitelích časopisu o hře Go, který se vydával v Rakousku v letech 1909 – 1910. A v českém jazyce vyšla pravidla Go údajně již ve 30. letech 20. století.

Hra Go se považuje za nejsložitější deskovou hru, a to díky počtu variant, které dosahují až 10^{800}(na rozdíl od šachů, které mají "pouze" 10^{120} možností), což je více, než je

atomů ve vesmíru. Na druhou stranu, i přes tento ohromný počet variant, samotná pravidla hry jsou velmi jednoduchá.

Ke hře se používá hrací deska zvaná goban, která má 19x19 průsečíků, při rychlých či výukových hrách se používají hrací desky menší (13x13, 9x9). Každý hráč má své hrací kameny (černé a bílé) a v každém svém tahu může na desku položit pouze jeden kámen. Cílem hry je zajmout co nejvíce nepřátelských kamenů.

Hráči mají v rámci své zkušenosti s hrou různé hodnosti, pokud hraje silnější hráč se slabším v rámci vyrovnané hry, dostane silnější hráč takzvané hendikepy. Tedy slabší hráč začíná již s předem určeným počtem kamenů na herní ploše.

Samotná hra, díky neskutečnému množství variant jak a kam hrát, byla po dlouhý čas poslední klasickou strategickou hrou, u které počítače nedokázaly porazit nejlepší hráče (šachy podlehly počítači Deep Blue roku 1997). To již není pravdou. Program AlphaGo společnosti GooogleDeepMind porazil na jaře roku 2017 tehdy nejlepšího hráče Go na světě, čínského profesionálního hráče KcheŤie. Střetli se již po druhé a sám KcheŤie chválil vývoj AlphaGo, a to v tom směru, že se od jejich prvního střetnutí značně zlepšil. Tento druhý a hraniční zápas skončil 3:0 pro AlphaGo, hrálo se na dvě vítězství ze tří her a velikost herní desky byla klasická 19x19. Program AlphaGo využívá ke své práci algoritmus Monte Carlo, rozhoduje se na základě umělé neuronové sítě (strojové učení) a to pozorováním her jak hráčů, tak i jiných programů (5).

3.2 Teorie her

Teorie her se zabývá rozhodováním v konfliktních situacích.

Konfliktní situací rozumíme takovou situaci, kdy naše okolí (protihráč) může na naše kroky nějak zareagovat. Převážně je to nějakým protiopatřením, aby zmírnilo dopad námi vytvořené situace, či otočilo onu situaci ve svůj prospěch.

Dobrým příkladem konfliktní situace, kterou řeší teorie her, je oligopol. Pokud jedna firma z oligopolu rozhodne, že upraví svou výrobu pro maximalizaci zisku (bude vyrábět více), ale bez ohledu na ostatní podniky, může se snadno stát, že se cena výrobku na trhu sníží. A to z důvodu, že i ostatní podniky mohou též navýšit svou výrobu a trh bude přesycen daným výrobkem. Tím pádem zisky "našeho" podniku nebudou optimální v rámci předem nastavené výroby.

Základní pojmy:

- Hra – každá konfliktní situace, kterou může řešit teorie her.
- Hráč – každý, kdo se účastní hry a může její výsledek svými kroky (tahy) ovlivnit.
    - Racionální – snaží se o co nejlepší výsledek hry.
    - Iracionální – výsledek hry ho nezajímá.
- Strategie – chování hráče dle vybrané alternativy
- Výplata – výsledek hry vyjádřený kvantitativně, je posuzovaný z hlediska uvažování hráče.
    - Kladná hodnota – užitek pro hráče.
    - Záporná hodnota – prohra hráče.
- Výplatní funkce – předpis pro výplatu hráče v závislosti na vybrané strategii.

Rozdělení her dle kritérií:

- Počet hráčů – hry pro dva a *n* hráčů, kde *n* je více jak dva hráči.
- Součet výplat hráčů
    - Hry s konstantním součtem – hráč A může získat pouze tolik, kolik hráč B ztratí.
        - Hry s nulovým součtem – zvláštní případ hry s konečným součtem.
    - Hry s nekonstantním součtem – co hráč A získá, nemusí nutně hráč B ztratit.
- Velikost prostoru strategií
    - Hra konečná – pokud je množina strategií hráčů konečná, je i sama hra hrou konečnou.
    - Hra nekonečná – pokud množina strategií je alespoň pro jednoho hráče nekonečná, jedná se o hru nekonečnou.
- Informovanost hráčů
    - Hry s úplnou informovaností – hráči znají aktuální stav hry, její dosavadní průběh.
    - Hry s neúplnou informovaností – hráči nejsou nijak informováni o průběhu hry.
- Zájem hráčů
    - Hry antagonistické – hráč A může získat pouze tolik, kolik hráč B ztratí.

    - Hry neantagonistické – co hráč A získá, nemusí nutně hráč B ztratit.
        - Kooperativní – hráči mohou mezi sebou vytvářet koalice.
        - Nekooperativní – hráči mezi sebou nemohou vytvářet koalice.
- Důsledky voleb
    - Hry deterministické – při zvolené strategii hráč přesně ví, co získá.
    - Hry stochastické – do hry vstupuje náhoda, při zvolené strategii má výplata pravděpodobnostní rozdělení.
- Racionalita hráčů – hra hraná proti inteligentnímu hráči (hra dle minimaxu[1]) a hra hraná proti neinteligentnímu hráči (přírodě).

Krom výše zmíněného, můžeme dělit hry i podle hráčem předem zvoleným oborem strategie na strategie čisté a strategie smíšené. Toto dělení je možné pouze u her v maticovém stavu, kdy se jedná o antagonistický konflikt s konečným počtem strategií.

Obor čistých strategií znamená, že hráč dosáhne svého cíle pouze za použití jedné z možných strategií. Optimální chování hráče, tedy to, co mu přinese nejlepší možnou výplatu, je dáno právě touto jednou strategií.

V oboru čistých strategií též používáme pojem sedlový bod. Sedlový bod je optimální řešení pro oba hráče při řešení antagonistického konfliktu. Ani jednomu hráči se nevyplatí zvolit jinou strategii. Tedy pokud hráč A volí strategii vedoucí k sedlovému bodu, ale hráč B volí jinou strategii, na konci hry na tom bude hráč B hůře než hráč A.

Řešení hry v oboru čistých strategií je tedy definováno tak, že maticová hra má řešení v oboru čistých strategií právě tehdy, když má sedlový bod.

Pokud maticová hra sedlový bod nemá, musíme řešení hledat v oboru smíšených strategií. V tomto oboru má každá strategie hodnotu pravděpodobnosti jejího použití. Například při hře kámen – nůžky – papír je pravděpodobnost pro každou strategii dána 1/3, čili 33,33%. (6)

[1] Principem je minimalizace maximálních možných ztrát při hře dvou a více hráčů (např. dáma, šachy)

3.3 Vývojové prostředí -Microsoft Visual Studio 2017

Microsoft Visual Studio 2017 je poslední verze vývojového prostředí (IDE) od firmy Microsoft. Tato poslední verze byla vydána 7. března 2017 a její poslední aktualizace na verzi 15.9.9 proběhla 12. března 2019.

Toto vývojové prostředí slouží k vývoji jak konzolových aplikací, tak aplikací s grafickým rozhraním. Podporuje vývoj aplikací (Windows Forms, webové aplikace, webové služby, a to ve strojovém nebo řízeném kódu) např. na platformách Microsoft Windows, Windows Mobile, .NET a Microsoft Silverlight.

3.3.1 Microsoft Visual C#

Tento programovací jazyk byl vyvinut firmou Microsoft dohromady s platformou .NET Framework a jedná se o vysokoúrovňový objektově orientovaný jazyk.

3.4 Umělá inteligence

Umělá inteligence (v angličtině Artificial intelligence, tedy AI) patří do oboru informatiky, zabývající se tvorbou strojů, počítačových programů, prokazujících známky inteligentního chovaní. Příkladem takového chování může být schopnost učit se, plánovat.

AI rozlišujeme na slabé (hloupé) a silné (chytré). Slabé AI projdou přes Turingův test, ale neobstojí proti argumentu čínského pokoje. Silné umělé inteligence by měli nejen překonat Turingův test, ale i prokázat, že rozumí otázkám, které dostávají a i svým vlastním odpovědím. Neb schopnost porozumění je to nejdůležitější co od takovéto AI očekáváme.

S pojmem umělá inteligence se setkáváme i v počítačových hrách, kde se jedná o předem nascriptované chování protivníků, kteří ve větší či menší míře reagují na činy, chování a způsob jakým hráč samotnou hrou prochází. I z tohoto důvodu je později zmíněné algoritmy a strategie počítačem ovládaných protihráčů označováno jako. (7)

3.4.1 Turingův test

Test pojmenovaný po Alanovi Turingovi, který ho prezentoval roku 1950, má za cíl zjistit, zdali se systém AI skutečně chová inteligentně či nikoliv. Z důvodu, že samotný pojem inteligence je těžce definovatelný, používá tento test porovnání s člověkem.

Test probíhá tak, že do navzájem oddělených místností umístíme člověka, počítač se SW vybavením, o kterém předpokládáme, že je AI a pozorovatele. Pozorovatel pokládá otázky jak člověku tak AI bez znalosti, kdo je kdo. Pokud dle jejich odpovědí není schopen určit, s kým komunikuje, tedy kdo je AI a kdo je skutečný člověk, AI prošlo Turingovým testem.

Turingův test byl dlouho považován za základní měřítko toho, co by měla AI zvládat. Již od roku 1966 (program ELIZA Josepha Weizenbauma) jsou známy programy, které jsou schopny částečně splnit podmínky Turingova testu, ale rozhodně nejsou AI, tzv. chatterboti.

Na některé nedostatky Turingova testu poukazuje i argument čínského pokoje. (8)

3.4.2 **Argument čínského pokoje**

Tento argument Johna Searla byl představen v časopise The Behavioral and Brain Sciences v roce 1980. Je brán jako odmítnutí Turingova testu jakožto průkazu zda daný program je AI či nikoliv.

Teoreticky máme pokoj, ve kterém jsou v tištěné formě všechny věty, které mohou v čínštině vzniknout (proto čínský pokoj). V pokoji se také nachází operátor, kterému jsou podávány otázky v čínštině, kterou operátor vůbec neovládá. Úkolem operátora, je na tyto otázky smysluplně odpovídat.

Samotným argumentem je to, že operátor, který čínštinu, jak bylo řešeno, nezná, je postupem času schopen na otázky adekvátně odpovídat. A to díky tomu, že se naučí porovnávat znaky v pokoji se znaky otázek. Tento postup porovnání by programátoři mohli rovnou zahrnout do svého algoritmu a právě proto se Searl domnívá, že by se nejednalo o AI, ale o pouhý "automat". (9) (10)

4 Vlastní práce

4.1 Desková hra Lovci noci

"Společenská hra se skrytými rolemi. Temný je svět stínu! S vražednou nenávistí připraveni k boji stojí proti sobě upíři a vlkodlaci. Upíři loví vlkodlaky, vlkodlaci loví upíry. Lidé se ocitají uprostřed mezi nimi, někdy bojují na jedné straně a jindy zase na straně druhé. Která skupina bude vašim osudem? Pomocí potřebného vybavení a mocných kouzel zničte své protivníky a jednou provždy ukončete krvavý spor. Zvítězí upíři, vlkodlaci, nebo lidé? Rozhodnutí je na vás. " (11)

Desková hra Lovci noci je společenskou hrou pracující se skrytými rolemi hráčů a jejich různými cíli. Hráči hrají proti sobě v souladu s rolí (postavou), kterou si na začátku hry vylosovali. V základu jsou ve hře dvě strany, Vlkodlaci a Upíři, kteří se snaží navzájem se vyhladit. Mezi tímto konfliktem proplouvají lidé s vlastními úkoly, ať již jim jde o pouhé přežití, o nasbírání vybavení či přesně cílených artefaktů nebo například o to, aby umřeli ve hře jako první.

Autorem této deskové hry je Yasutaka Ikeda (dle serveru boardgamegeek.com má na svém kontě 18 různých her) a byla vydána pod názvem Shadow Hunters. Původně bylo ve hře pouze 10 hratelných postav, ale později se dočkala rozšíření o dalších 10 postav. Tím se dosáhlo stavu, kdy Upíři a Vlkodlaci mají po šesti charakterech a strana Lidí má osm hratelných postav.

Hra Shadow Hunters byla v roce 2006 nominována na cenu nejlepší japonské deskové hry (Japan Boardgame Prize Best Japanese Game Nominee) a v roce 2010 byla nominována na cenu As d'Or - Jeu de l'Année. (12)

4.2 Pravidla hry

Zkrácená verze pravidel hry v této práci, která platí v programu, se zabývá jen klíčovými faktory hry. Plná verze pravidel se nachází v příloze. Pravidla hry umožňují výběr z více variant hry, v práci jsem se ale soustředil pouze na variantu, která nám jako hráčům během hraní hry přišla dlouhodobě nejzábavnější a podporovala hraní složitějších strategií, založených na řádném zjišťování, kdo je kdo, a na sledování celé partie, místo bezhlavého vyřazování jednotlivých hráčů.

4.2.1 **Začátek hry**

Před samotnou hrou se musí připravit herní plán, kam se nahodile rozmístí žetony lokací a kde dvě vedle sebe ležící lokace tvoří loviště. Dále se pak musí zamíchat Červené, Zelené a Modré karty a podle předem zvoleného pravidla rozdat hráčům karty postav.

"*Při hře čtyř hráčů, dle pravidel, hrají dva vlkodlaci a dva upíři. Postavy lidí se při tomto počtu hráčů nehrají."* (13)

4.2.2 **Průběh kola**

"Průběh hry v krátkosti:
Hráč, který je na řadě (aktivní hráč), provede ve svém tahu následující akce:
1. Hráč musí hodit kostkami a pohnout svou figurkou.
2. Hráč může využít možnosti MÍSTA, na které se pohnul.
3. Hráč může zahájit ÚTOK (od druhého kola).
Poté ve hře pokračuje jeho soused po levici svou akcí 1." (13)

Kromě citovaného se může hráč kdykoliv odhalit, tj. ukázat ostatním hráčům za jakou postavu hraje a hráči znají jeho rasu, cíl, počet životů a schopnost jeho postavy, a to i mimo své kolo, a používat svou schopnost tak, jak je u karty jeho postavy popsáno.

Hra končí ve chvíli, kdy alespoň jeden hráč splní svůj cíl. Svůj cíl ale může splnit více hráčů najednou, a tedy i vítězů může být vícero.

4.2.3 **Pohyb po mapě**

Akcí POHYB začíná kolo každého hráče. Pokud se má nějaká herní akce (většinou použití schopnosti postavy či použití několika speciálních karet) vykonat na začátku kola hráče, je to ještě před pohybem/hodem kostkami. Hod kostkami a především následný pohyb figurky po mapě ukončuje akci pohybu.

Pro pohyb po mapě může hráč použít tři různé možnosti. První možností je pomocí kostek, kdy hráč hodí šestistěnnou a čtyřstěnnou kostkou. Součet hozených čísel je číslem lokace, kam se hráč po hodu přesune. Pokud padne číslo sedm, může se hráč přesunout na jakoukoliv lokaci. Druhou možností je použít předmět Mystický kompas a třetí možností je použít schopnost postavy. Obě tyto možnosti mají svá vlastní omezení, která jsou v jejich popiscích napsaná.

4.2.4 **Využití možnosti MÍSTA**

Všech šest míst je před hrou náhodně rozmístěno po herním plánu. Díky tomu je každá hra trochu jiná.

"Na čtyřech ze šesti MÍST si hráč může vzít jednu lícem dolů otočenou kartu (červenou, modrou, nebo zelenou). Karty přinášejí hráčům různé výhody. Je-li některý balíček karet spotřebován, zamíchejte příslušný odkládací balíček a použijte ho jako nový dobírací balíček." (13)

4.2.5 **Akce ÚTOK**

Pokud chce aktivní hráč na někoho zaútočit, musí se daná postava nacházet ve stejném lovišti jako útočící hráč. Útok se provede jednoduše. Aktivní hráč hodí šestistěnnou i čtyřstěnnou kostkou. Od čísla na šestistěnné kostce odečte číslo, které padlo na čtyřstěnné kostce. Pokud je výsledek nula a nižší, útok nebyl úspěšný a k žádnému zranění nedošlo.

V originálních pravidlech se odečítá nižší číslo od vyššího, ale toto pravidlo bylo v počítačové verzi upraveno na základě dříve na živo odehraných partií pro větší zábavnost a vymýcení bezplánovitého útočení.

4.2.6 **Zvláštní schopnosti postav**

Každá postava má jednu zvláštní schopnost. Tu může použít, pouze pokud je postava odhalena[2].

Ve hře existuje několik typů schopností. Některé schopnosti je možno použít pouze na začátku kola, jiné pouze na konci kola. Další schopnosti je možné použít kdykoliv, ale pouze jednou za hru a poslední druhy schopností jsou aktivní stále.

4.2.7 **Karty**

"Modré i červené karty jsou rozděleny na dva druhy:
Událost: Instrukce na těchto kartách přečtěte nahlas.
Instrukce „musí" se provedou ihned, instrukcí bez „musí" je možno se zříci (nemusí být provedeny). Poté kartu vždy odložte lícem nahoru na odkládací balíček odpovídající barvy.

[2] Viz kapitola 4.2.2. Průběh kola

Vybavení: Tyto karty také přečtěte nahlas, poté si je položte lícem nahoru před sebe. Tyto karty účinkují pořád a mohou být využity ihned.
Většina z vyložených karet vybavení musí být používána vždy, pokud hráč kartu vlastní.
U některých karet s formulací „může“ nebo podobnou si hráč smí zvolit, zda kartu použije či nepoužije." (13)

Zelené karty, věštby, mají jiná pravidla, která je třeba dodržovat. Aktivní hráč se těchto pravidel musí při použití těchto karet držet.

"Text si přečte pouze sám pro sebe a poté se musí rozhodnout, kterému hráči tuto kartu dá. Poté tuto kartu skrytě předá zvolenému hráči – příjemci. Příjemce si nyní přečte text rovněž pouze sám pro sebe a zkontroluje, zda se týká jeho vlastní postavy.

Týká-li se jej text, musí příjemce instrukci vykonat, ale nesmí prozradit nic dalšího.

Netýká-li se jej text, řekne příjemce: „Nic se nestalo“.

Poté kartu vrátí zpět aktivnímu hráči, který ji odloží lícem dolů (!) na odkládací balíček zelených karet.

Aktivní hráč tímto získal informaci o příjemci. Zelené karty slouží převážně k získávání informací o tom, kdo hraje za jakou postavu." (13)

4.3 Lovci noci z pohledu teorie her

Z pohledu výše zmíněné teorie her, jde deskovou hru Lovci noci charakterizovat takto:

- Počet hráčů – hry pro *n* hráčů, kde *n* je více jak tři hráči.
- Součet výplat hráčů
    - Hry s nekonstantním součtem – co hráč A získá, nemusí nutně hráč B ztratit.
- Velikost prostoru strategií
    - Hry konečné – pokud je množina strategií hráčů konečná, je i sama hra hrou konečnou.
- Informovanost hráčů
    - Hry s úplnou informovaností – hráči znají aktuální stav hry, její dosavadní průběh.
- Zájem hráčů
    - Hry neantagonistické – co hráč A získá, nemusí nutně hráč B ztratit.

        - Kooperativní – hráči mohou mezi sebou vytvářet koalice.
- Důsledky voleb
    - Hry stochastické – do hry vstupuje náhoda, při zvolené strategii má výplata pravděpodobnostní rozdělení.
- Racionalita hráčů – hra hraná proti inteligentnímu hráči (hra dle minimaxu).

Ale v rámci zvolených pravidel k počtu hráčů se musí charakteristika hry lehce změnit:

- Součet výplat hráčů
    - Hry s konstantním součtem – hráč A může získat pouze tolik, kolik hráč B ztratí.

O tom, zda je hra antagonistická či nikoliv, můžeme diskutovat, neboť máme sice čtyři různé hráče, ale pouze dvě strany. Tedy z pohledu hráčů je hra neantagonistická a hráči mohou, dokonce musí, v rámci rasy své postavy tvořit koalice. Na druhou stranu z pohledu herních ras dle vybraných pravidel, je to hra antagonistická, neboť Upíři nemohou uzavírat koalici s Vlkodlaky.

Samozřejmě máme výjimku z toho pravidla a tou je postava upírky Ursany, která se snaží hráče rasy Vlkodlaků zmást a tvářit se jako jeden z nich. A mezi tím si snaží připravit pozici pro jejich poražení.

4.4 Strategie AI

Strategie hry je postup, jak bude AI ve hře reagovat, aby dosáhla vítězství dle vylosované karty postavy. Neboť každá rasa, v případě rasy člověk i postava, má vlastní strategii, jak splnit svůj cíl a vyhrát hru.

Níže uvedené strategie se zakládají na skutečně odehraných partiích hry Lovci noci a zvolených přístupech hráčů ke hře. Hráči byli ve věku od 10 do 50 let. Většina partií byla odehrána v období od vydání dekové hry Lovci noci firmou Mindok s.r.o. (2010) do konce roku 2018 v dětském počítačovém klubu Kapsa.[3]

[3] https://kapsa.cz/cs/klub-kapsa

4.4.1 **Obecné**

Pro většinu postav je úplně primární cílem zjistit, kdo je jeho spoluhráč, kolik spoluhráčů ve hře má a kdo je jeho nepřítel. Proto, až na výjimky, se snaží všechny postavy v prvních kolech hry získat a zahrát zelené karty [4], díky kterým zjistí, kdo je kdo.

Po zjištění nepřítele a případně spoluhráče je potřeba začít plnit svůj cíl daný rasou a postavou. A tedy brát primárně červené karty k posílení svého bojového potenciálu a sekundárně modré karty[5] určené pro léčení a obranu. Během hry je stále vhodné příležitostně používat zelené karty ke kradení vybavení nepřátelům.

Zjišťování, kdo je kdo, neprobíhá jen pomocí zelených karet, ale i sledováním dění na herní desce. Kdo na koho útočí, kdo koho léčí, kdo se zajímá jen o sběr předmětů.

Pokud již AI bude vědět, kdo je její protivník, tak postava, která na protivníka bude útočit, bude považována za spojence AI. A samozřejmě naopak, pokud bude někdo útočit na spojence AI, bude považován za protivníka. Praxe ukázala, že tento postup může vést k chybám, ale pouze v případě, kdy lidský hráč chybně odpověděl na věštbu. Tato situace by v rámci programu neměla nastat.

Tedy je nutné, aby jednotlivé AI měly tabulku se jmény hráčů, kam si budou zapisovat informace, které během hry zjistí. Tabulka bude aktuální pouze pro danou partii a před každou novou partií se obnoví do výchozího, tj. prázdného, stavu.

	Upír	Vlkodlak
Uživatel		
Červenka		
Zelenka		
Žluťka		

Tabulka 1 - Prázdná paměťová tabulka se všemi hráči

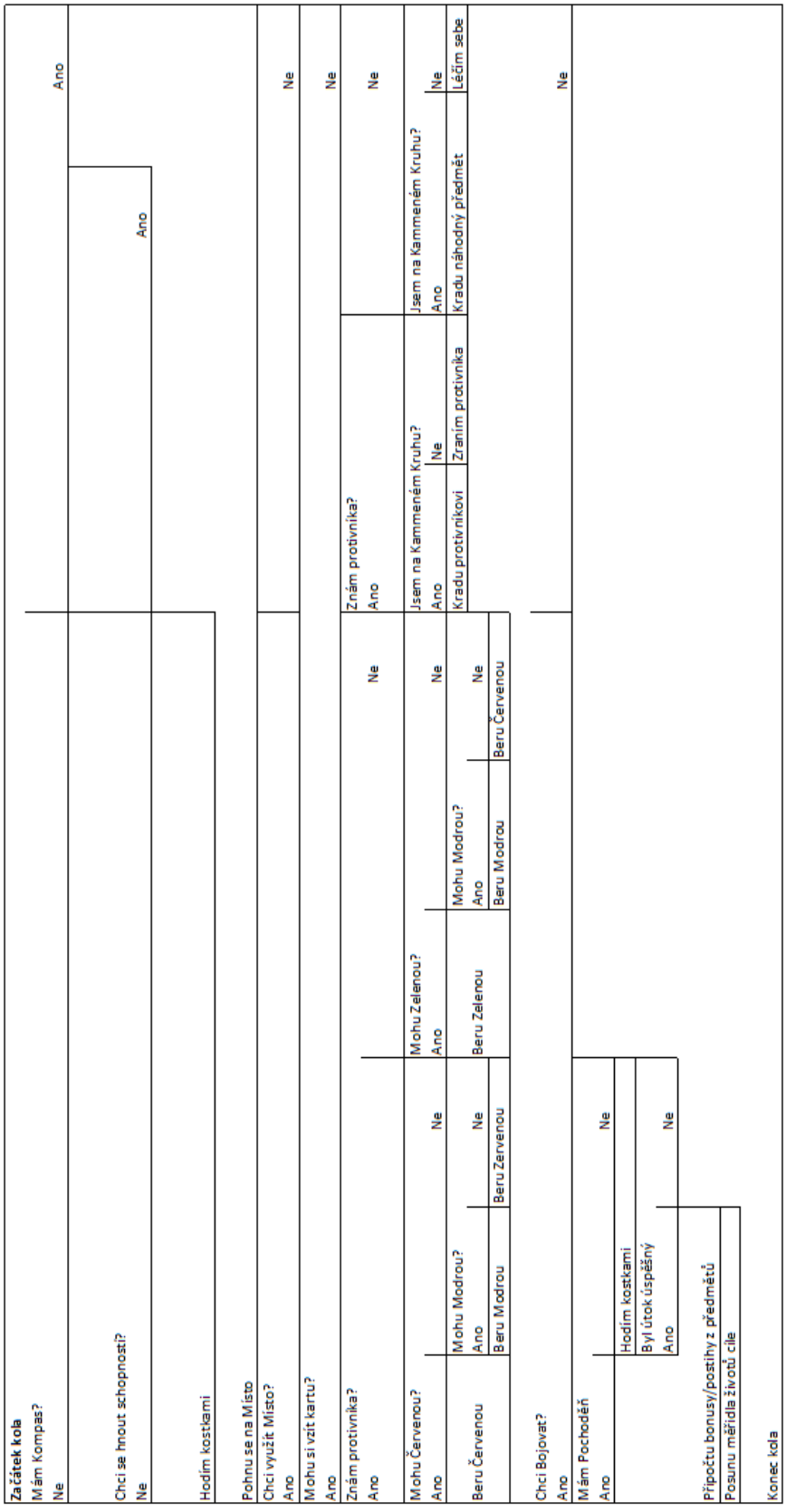
Samozřejmě v tabulkách pro jednotlivá AI nebude chybět jejich vlastní označení, se kterým AI bude sama dál pracovat. Tyto tabulky se budou optimálně řešit jako databázová tabulka, do které si bude algoritmus hráče zapisovat zjištěné informace a ověřovat to, co již zjistil.

AI si bude muset dle později zmíněných pravidel hlídat, kdy a za jakých podmínek smí a "chce" použít schopnost své postavy.

[4] Viz kapitola 7. Příloha Lovci noci – pravidla hry kapitola 2. Využití možnosti MÍSTA
[5] Viz kapitola 4.1.6. Karty či kapitola 7. Příloha Lovci noci – Almanach str. 6 Karty podrobněji

Postavy se odhalují dle popsaných strategií, nebo aby se vyhnuly možnosti, že na ně zaútočí spoluhráč.

4.4.2 Obecné algoritmy



Obrázek 1 - Obecný algoritmus AI v rámci svého kola

Jak je vidět na obrázku výše, obecný algoritmus pro celé kolo je velký a nepřehledný. A to i přes to, že neobsahuje veškeré nutné podmínky (např. odhalení postavy, využití schopnosti, možnosti karet a míst). Z toho důvodu byl algoritmus rozdělen podobně, jako bude rozdělen kód aplikace.

Algoritmy pro jednotlivé schopnosti nejsou v tomto přehledu uvedeny, neboť pro každou postavu bude muset být individuální, minimálně co se vlastní schopnosti postavy týká. Obecné algoritmy se budou chovat dle níže popsaných modelů.

Kolo
volám **Odhlal_start**
Volám **Pohyb**
Volám **Místo**
volám **Boj**
volám **Odhlal_konec**
Konec kola
volám **Cizi_k**

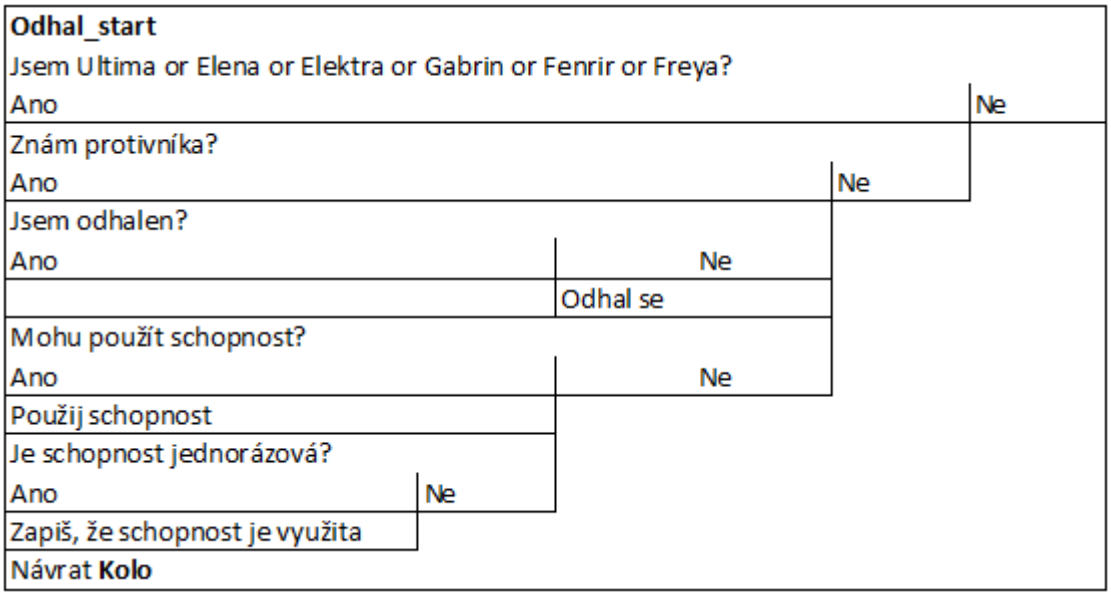
Obrázek 2 - Obecný algoritmus AI volající jednotlivé části

Všechny níže uvedené algoritmy se týkají pouze postav z ras Vlkodlaků a Upírů, neboť v tomto počtu hráčů se dle pravidel hry postavy rasy Člověk nepoužívají.

4.4.2.1 Odhalení postavy

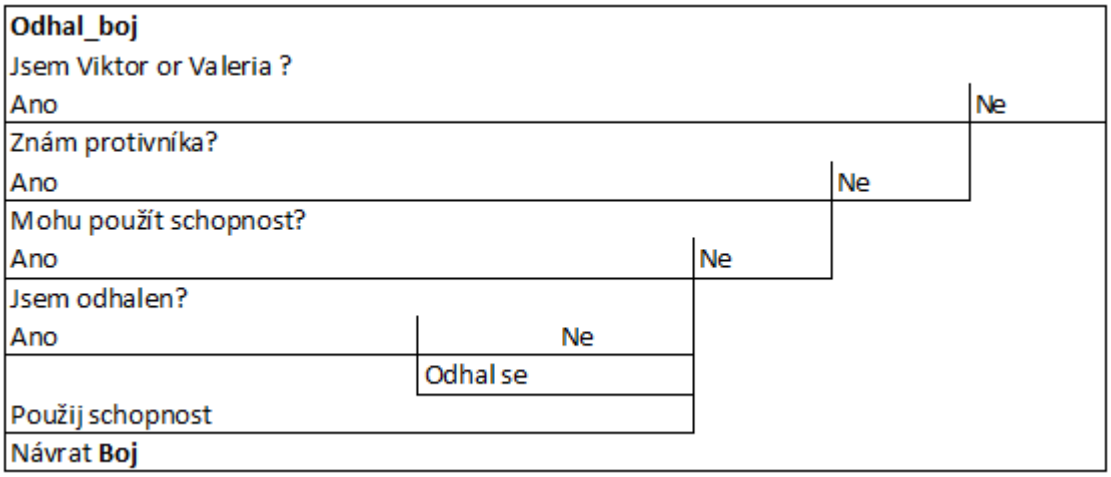
Pro odhalení postavy existují dva vážné důvodu. Chci použít schopnost je prvním z nich. V tomto případě je nutné řešit, kdy se schopnost aktivuje. Zda na začátku kola, při boji nebo až na konci kola.

Druhým případem je situace, kdy jsem napaden spoluhráčem a chci takovému útoku v budoucnu zamezit. Například jsem ve stejném Lovišti jako můj spoluhráč a jeden z protihráčů. Je spoluhráčův tah a ten se chystá zaútočit nebo pro nedostatek informací ukončit kolo. Odhalením identity své postavy zamezím teoretickému útoku na sebe a naopak dodám spoluhráči informaci na koho zaútočit. Ideální je ten stav, pokud po svém spoluhráči jsem na tahu já a protihráči nemohou znalosti mé postavy hned využít, zatím co já mohu použít svou schopnost postavy.



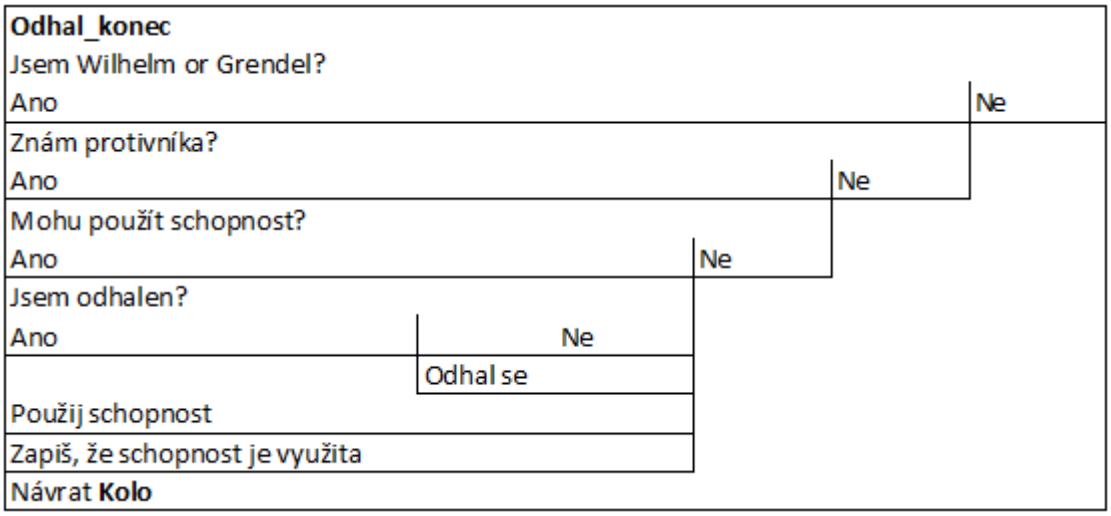
Obrázek 3 - Algoritmus AI v rámci odhalení se a použití schopnosti na začátku kola

Tento algoritmus řeší případ postav, kterým se schopnost aktivuje na začátku kola. Vždy je důležitým aspektem vědomost, kdo je protivník, a tedy můj cíl.



Obrázek 4 - Algoritmus AI v rámci odhalení se a použití schopnosti před Bojem

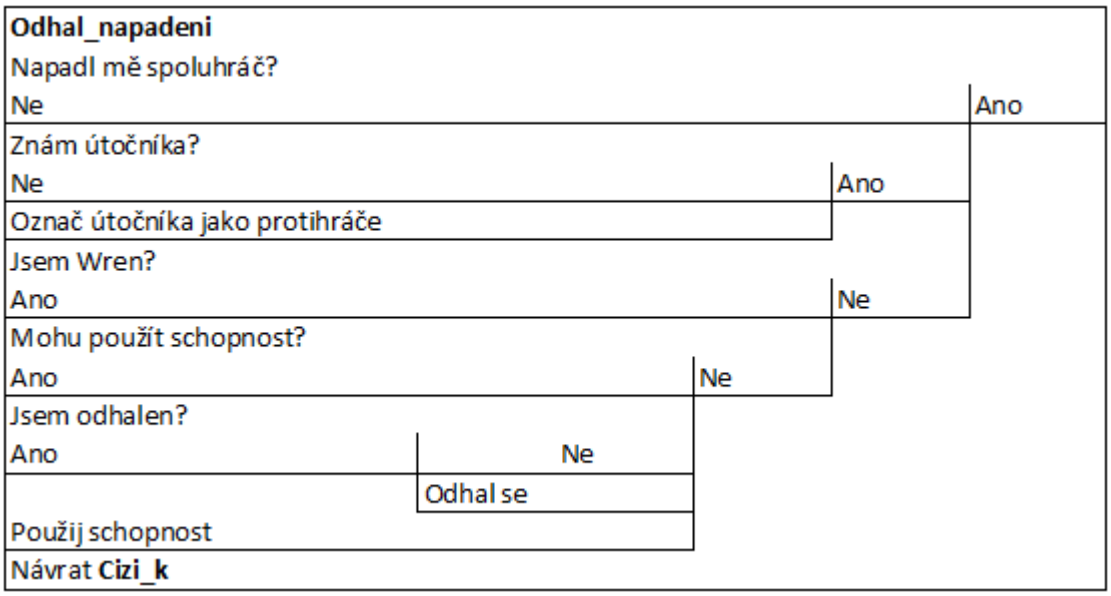
Dle stejného principu se algoritmus Odhal_boj týká postav, které mají schopnost vázanou na boj, a z toho důvodu se musí odhalit ještě před tím, než boj započne.



Obrázek 5 - Algoritmus AI v rámci odhalení se a použití schopnosti na konci kola

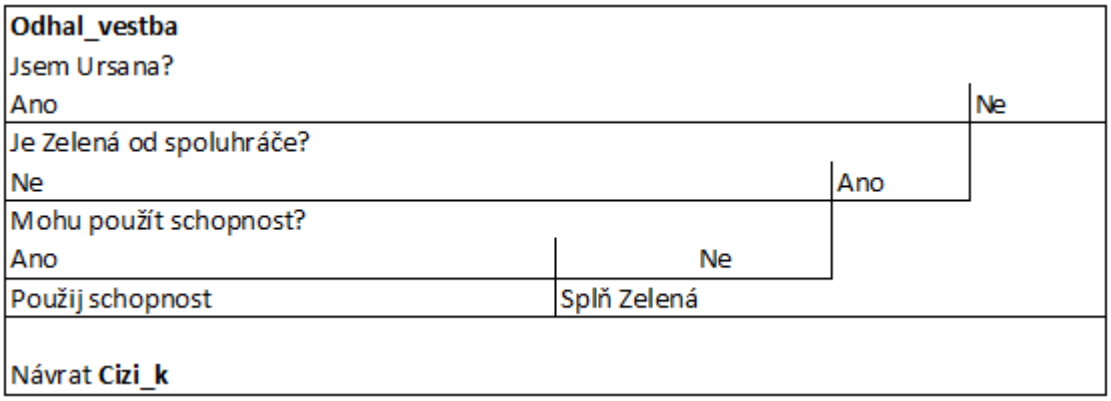
Stejně tak Odhal_konec se týká pouhých dvou postav, které mají schopnost aktivní ke konci kola.

Jelikož i v této hře existují výjimky z pravidla, máme zde dvě postavy, jejichž schopnost je aktivní v rámci cizího kola. Obě postavy byly zapracovány do obecných algoritmů.



Obrázek 6 - Algoritmus AI v rámci reakce na napadení jiným hráčem

Prvním z nich je algoritmus pro napadení, který rovnou rozlišuje, kdo mě napadl a umožňuje dál s touto informací pracovat.

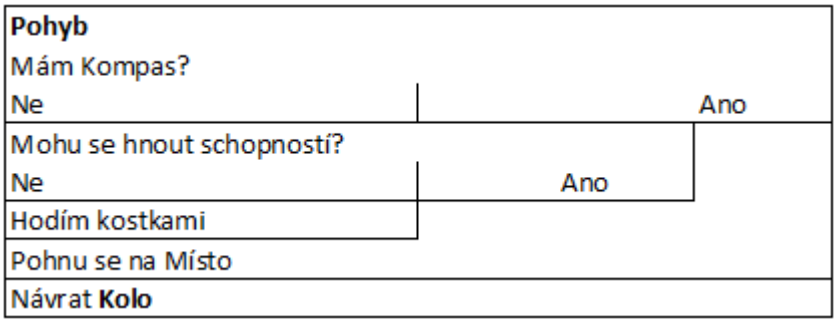


Obrázek 7 - Algoritmus AI v rámci reakce na Zelenou kartu

Druhým je reakce AI na Zelené karty věšteb a k tomu přidaná schopnost Ursany, které je jedinou postavou ve hře, jejíž schopnost není podmíněna odhalením své postavy.

4.4.2.2 Pohyb

Jak už bylo zmíněno výše, pohyb po mapě jde provést třemi různými způsoby. Následující algoritmus všechny možnosti zohledňuje a následně pokračuje v kole aktivního hráče.



Obrázek 8 - Algoritmus AI v rámci akce Pohyb

4.4.2.3 Místo

Algoritmus pro využití Místa je opět o něco větší, než by bylo příjemné, zvlášť kvůli Zeleným kartám. Proto byl rozdělen na čtyři samostatné části, obecnou a pak dle barev karet.

Červená Vezmi **Červenou** kartu	
Je to **Vybavení?**	
Ne	Ano
Vyhodnoť efekt karty	Přidej k seznamu vybavení
Odhoď kartu	
Návrat **Místo**	

Obrázek 9 - Algoritmus AI v rámci hraní Červené karty

Modrá Vezmi **Modrou** kartu	
Je to **Vybavení?**	
Ne	Ano
Vyhodnoť efekt karty	Přidej k seznamu vybavení
Odhoď kartu	
Návrat **Místo**	

Obrázek 10 - Algoritmus AI v rámci hraní Modré karty

Jak je vidět, Modré a Červené karty mají prakticky stejné algoritmy, jediný rozdíl je balíček, ze kterého se dané karty tahají. Proto je ve vlastním kódu pro tyto dvě karty stejný algoritmus lišící se právě jen tím, z jakého balíčku kartu vybrat.

Algoritmus pro Zelené karty bere v potaz i věštby, které hráč již dal i dostal. Stejně tak jako v identifikaci hráčů řeší, zda byl již hráč napaden.[6]

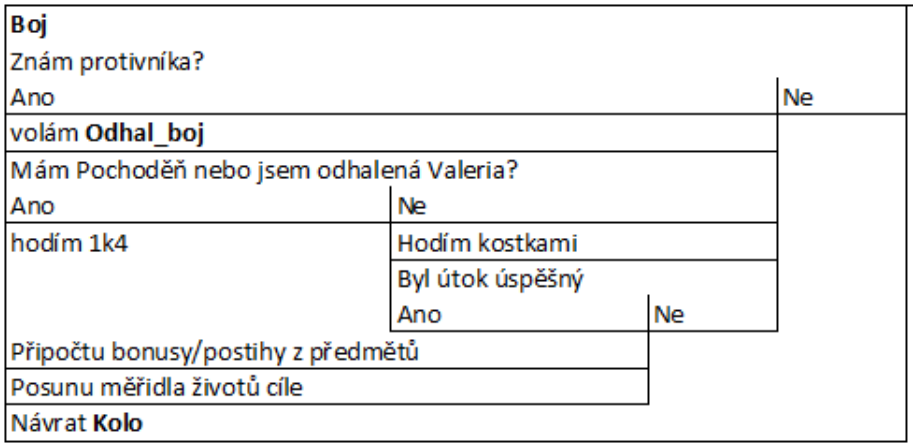
[6] Viz 4.4.2.5 Cizí kolo

Obrázek 11 - Algoritmus AI v rámci využití Místa

Obrázek 12 - Algoritmus AI v rámci hraní Zelené

4.4.2.4 Boj

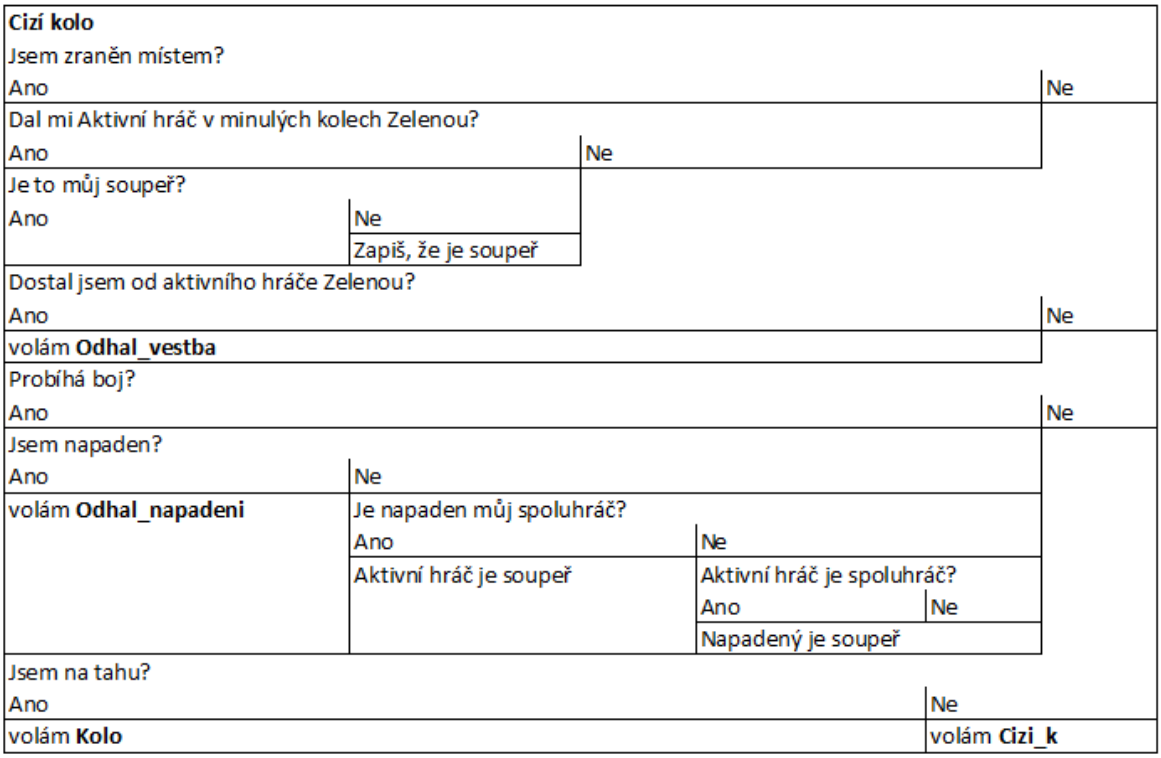
Důležité je při boji rozlišit, kdo je protivník a kdo spoluhráč. Jelikož se v tomto počtu hráčů vyplatí napadnout protivníka hned, jak nějakého odhalím, nepočítá algoritmus se zdrženlivostí v boji.



Obrázek 13 - Algoritmus AI v rámci Boje

4.4.2.5 Cizí kolo

Cizí kolo je pro většinu postav možností, jak získat informace o dalších hráčích dle jejich stylu hry a rozhodnutí, která během svého kola udělají.



Obrázek 14 - Algoritmus AI v rámci Cizího kola

4.4.3 **Úprava strategie dle postavy hráče**

4.4.3.1 Upír

Cílem upírů je vyhladit všechny vlkodlaky. V praxi jsou upíři samostatně silnější než vlkodlaci. Původní hra nejspíše zohledňuje legendy, kde upíři byli samotáři a vlkodlaci lovili ve smečkách.

Upíři mají velmi prostou strategii boje. Najdu nepřítele, napadnu ho a v průběhu boje hledám vybavení pomocí červených karet. Odhaluji se, až když chci použít schopnost. Složitější strategie u schopnosti není nutná, neboť většina schopností této rasy funguje stále.

4.4.3.2 Ultima

Odhalí se a použije schopnost na začátku svého tahu, pokud některý z hráčů, který je vlkodlak, stojí na území Hřbitov. Toto se opakuje každé kolo, pokud je splněna podmínka.

4.4.3.3 Ursana

Schopností této upírky je lhaní na věštby (vybere si, zda chce, aby se jí věštba týkala či nikoliv a dle toho dá odpověď aktivnímu hráči). Z pohledu AI odpovídá v základu na věštby, jako by byla člověk. Výjimkou je, když dostane věštbu od hráče, kterého již zná. Pokud je tento známý hráč upír, přizná, že sama k této rase patří, ale pro všechny ostatní se stále tváří jako vlkodlak (při větším počtu hráčů nebo když jsou ve hře lidé, tak se tváří jako člověk)

Pro ostatní je odhalitelná pouze podle sledování chování hráče, podle toho, na koho útočí a komu pomáhá. Cílem Ursany je vnést zmatek mezi ostatní hráče, sama je, díky malému počtu životů, po odhalení snadno porazitelná.

4.4.3.4 Viktor

Odhalí se a použije schopnost, pokud je jeho útok úspěšný a je již zraněn.

4.4.3.5 Valeria

Odhalí se v případě, že útočí na Vlkodlaka. V ideálním případě má svůj vlastní útok podpořený některým z Předmětů z Červených karet.

4.4.3.6 Wren

Odhalí se v případě, že na ni zaútočí jiná postava. Útok nemusí být úspěšný, aby mohla být použita schopnost postavy.

4.4.3.7 Wilhelm

Z pohledu AI je to nejproblémovější postava z řad Upírů. V tomto počtu hráčů se vyplatí schopnost použít ve dvou případech. V prvním případě to je, když již byl vyřazen alespoň jeden hráč, v druhém, když Wilhelm potřebuje dva pohyby místo jednoho. Ideálně v situaci, kdy potřebuje dorazit vlkodlaka, který jeho předchozí útok stěží přežil.

4.4.4 **Vlkodlak**

Vlkodlaci chtějí vyhladit všechny upíry ve hře. Vlkodlaci jsou samostatně slabší než upíři. Čím má vlkodlak více životů, tím slabší schopnost má.

Strategie vlkodlaka je o něco složitější, protože kromě nalezení upírů potřebuje před začátkem utočení najít ideálně dalšího vlkodlaka. Použití schopností je též o něco složitější, neboť většinu vlkodlačích schopností lze využít pouze jednou.

4.4.4.1 Elany

Elany je díky své schopnosti dobrou stopařkou a lovkyní. Její schopnost jí pomáhá chytnout zraněného upíra a nenechat ho uniknout před lovící smečkou.

Z pohledu AI se Elany odhaluje a používá svou schopnost na začátku tahu, před hodem kostkami, a to v případě, že v dosahu své schopnosti má možnost napadnout již zraněného upíra.

4.4.4.2 Elektra

Elektra dokáže díky své schopnosti zvrátit hru. Schopnost používá v případě, že je již odhalený upír nebo ji použije na hráče, kterého si jako upíra odhalila pomocí Zelených karet. Ale to pouze v případě, že je jen jeden upír ve hře. Jejím primárním cílem jsou Viktor, Valeria a Wren.

4.4.4.3 Fenrir

Fenrirova schopnost je ideální v případě, že chybí 3-4 životy k tomu, aby byl protivník vyřazen. Z pohledu AI cílí na postavy se zraněním větším polovině jejich životů.

4.4.4.4 Freya

Schopnost této vlkodlačky dokáže buďto velmi uškodit soupeři nebo zachránit spoluhráče. Freya změní zranění jakékoliv postavy na hodnotu 7.

Pro AI to znamená použít tuto schopnost před útokem na silné protivníky jako je Viktor, Valeria, Wren a Willhelm. Nebo může zachránit krk vlkodlakům Fenrir, Freya, Gabrin a Grandel.

4.4.4.5 Gabrin

Gabrinova schopnost je podobná Fenrirově, jen místo používání 1k6[7] používá 1k4[8]. Z pohledu AI využívá schopnost dle stejný pravidel jako v případě Fenrira, jen obvyklá hodnota útoku se mění na 1-2 životy.

4.4.4.6 Grandel

Grandelovou schopností je jednokolová ochrana před veškerým zraněním, které by na něj bylo cíleno. Schopnost aktivuje na konci kola. Například je to vhodné, pokud lze předpokládat, že by další kolo mohl dostat větší zranění, než by mohl přežít.

Pro AI to znamená použít schopnost, když jsou jeho životy pod 6 a nemá žádnou jinou kartu, která by ho chránila (Modrá karta Anděl strážný).

4.4.5 **Člověk**

Lidé mají cíle různé a liší se postavu od postavy. Někteří jsou sběrači předmětů, jiní mají přežít, další mají za cíl pomoci jinému hráči či mají určitou postavu zabít. Každá postava má svou vlastní strategii, podle které se řídí chování AI ve vlastní hře. V praxi jsou některé postavy lidí velmi složité na hraní a v menším počtu hráčů, než jsou čtyři, jsou naprosto nehratelné, neboť nejde splnit jejich cíl.

[7] Značka 1k6 znamená hod jednou šestistěnnou kostkou
[8] Značka 1k4 znamená hod jednou čtyřstěnnou kostkou

4.4.5.1 Abigail

Abigail má jediný cíl, přežít. Její strategie je prostá. Sbírá Modré karty s obranným vybavením, sleduje, kdo s kým bojuje a případně pomůže vyhrávající straně. Po odhalení kdo je, nemají ostatní hráči důvod se ji snažit zabít. Většinou.

Z pohledu AI schopnost používá, když její životy klesnou pod třetinu, aby se plně vyléčila, dále si pomocí zelených karet určuje, kdo je kdo, a přiklání se ke straně, která má početní převahu.

4.4.5.2 Angela

Angela vyhraje, pokud vyhraje hráč hrající před ní nebo hráč hrající po ní. Nemusí být ani naživu, aby svůj cíl splnila.

AI stačí zjistit pomocí Zelených karet, kdo je vpravo a vlevo a pomáhat buď jedné nebo druhé postavě. Rozhodování je dle toho, kdo je méně zraněn. Primárně pomáhá hráči vpravo. Pokud by to vypadalo, že prohraje, odhalí se a svojí schopností změní strany.

4.4.5.3 Bella

Bella je sběratelka. Základní strategií je u ní prvně procházet modré karty a hledat předměty. Zelené používá na kradení předmětů od ostatních hráčů. Schopnost používá ve chvíli, kdy svým útokem zabije hráče, který má vybavení.

4.4.5.4 Barabas

Barabas je jednou z nejtěžších postav na hraní. Má těžký úkol a nepříjemnou schopnost. Jeho strategií je sbírat Červené karty vybavení a vyčkávat, až se postavy odhalí, a ideálně zraněnou postavu zabít. Snažit se dostat na lokaci Kamenný kruh je složité a v praxi se to zatím nikomu z hráčů nepovedlo.

Z pohledu AI sbírá vybavení, vyčkává a doráží oslabené postavy. Schopnost řeší jen v případě, že není odhalen a zabije postavu, která má 12 a méně životů. Jeho cílem je zabít postavy Daniel, Desmond, Viktor, Valeria, Wren, Wilhelm, Gabrin a Grandel. Pokud věštbou zjistí, že jeden ze soupeřů je upír, má velkou šanci, že svůj úkol splní (4:2).

4.4.5.5 Cyrus

Tato postava je druhou nejhůře hratelnou postavou. Zvlášť v počtu 4 hráčů. Jeho cílem je vyřadit hráče ve chvíli, kdy jsou již dva další hráči vyřazeni. Jeho strategií je sbírat Červené karty vybavení a vyčkávat, jak se bude hra vyvíjet. Do boje se zapojí, až když jsou ostatní hráči zraněni.

V případě, že je splněna podmínka dvou vyřazených hráčů, zaútočí na třetího a pokud je možnost odhalit se bez ukončení hry (splnění cíle jiného hráče), odhalí se a použije svou schopnost, aby hráče zabil.

4.4.5.6 Celeste

Celeste je prvním ze dvou sebevrahů ve hře. Jejím primárním cílem je umřít jako první. Strategie je snadná, rozhodí zelené karty mezi hráče a poté si ideálně vybere upíra nebo vlkodlaka, kterého napadne. Velkým plusem pro ni je, pokud má napadený hráč spoluhráče, který ho začne bránit.

Druhou možností je pomáhat slabší straně v boji a zůstat tak mezi posledními dvěma hráči (Celeste + 1). Pokud má možnost odhalit se bez toho, že by ukončila hru, tak se svou schopností na začátku každého kola vyléčí o jeden život. Díky tomu dokáže zůstat ve hře až do konce.

4.4.5.7 Daniel

Daniel je druhým sebevrahem, vzhledem k počtu jeho životů není pro něj dobrou strategií pokoušet se nechat se prvoplánově zabít. Lepší možností je přes zelené karty odhalit upíry ve hře a napadnout je. Nebo se chovat jako vlkodlak. Buď bude zabit upíry nebo pomůže vlkodlakům vybít upíry a splní si tak svůj druhý cíl. Pokud přežije.

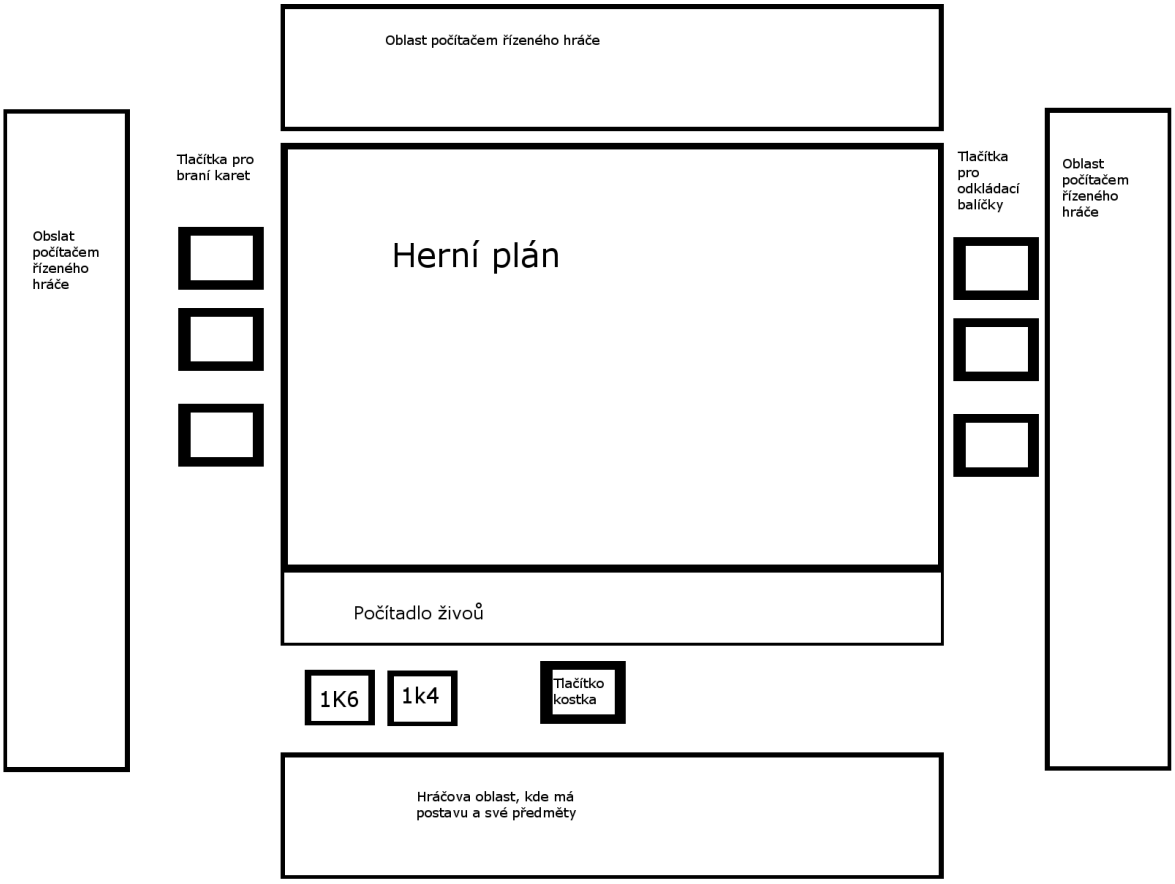
4.4.5.8 Desmond

Desmond je sběratel. Co se děje na plánu, ho nezajímá. Jeho jedinou starostí je procházení Modrého balíčku karet a hledání cílených předmětů. Ve chvíli, kdy získá tři ze čtyř možných předmětů, odhalí se a tím ukončí hru.

Schopnost používá jen v případě, kdy jeden z jeho hledaných předmětů skončí v odkládacím balíčku a zároveň Desmond již dva další má ve svém držení.

4.5 Nákresy

Původní návrh jak by mohlo vypadat vizuální ztvárnění hry. Od finální verze se liší i kvůli snazší ovladatelnosti pro hráče a lepší přehlednosti.



Obrázek 15 - Herní plán

4.6 Datový slovník

V práci nebyla použita databáze ale více rozměrná pole fungující jako tabulky. Pro jejich vysvětlení je datový slovník potřeba.

4.6.1 Karty postav

Jméno: jméno postavy

Životy: počet životů postavy. Jaké maximální poškození (hit points, zkráceně hp) postava snese, než je zabita

Schopnost: co dělá schopnost postavy, kdy se používá a kolikrát za hru jde schopnost použít

Stav: hodnota 0 značí, že karta nebyla použita. Hodnota 1, že již využita byla.

Jelikož jsou dle pravidel ve hře jen dvě rasy, a to Upíři a Vlkodlaci, kde každá má vlastní "tabulku", nemusí být u karet rozlišení rasy a cíle. Ty jsou řešeny při losování postav.

4.6.2 **Zelené karty událostí**

Jméno: jméno karty

Text: text karty pro uživatele

Rasa_A: rasa, které se věštba týká

Rasa_B: rasa, které se věštba týká

Možnost 1: co hráč ztrácí, pokud splňuje rasu

možnost 2: co hráč ztrácí, pokud splňuje rasu a nesplňuje Když

Stav: hodnota 0 značí, že karta nebyla použita. Hodnota 1, že již využita byla.

4.6.3 **Modré karty událostí**

Jméno: jméno karty

Použití: rozlišení, zda se jedná o jednorázovou kartu či o vybavení

Text: text karty

Stav: hodnota 0 značí, že karta nebyla použita. Hodnota 1, že již využita byla.

4.6.4 **Červené karty událostí**

Jméno: jméno karty

Použití: rozlišení, zda se jedná o jednorázovou kartu či o vybavení

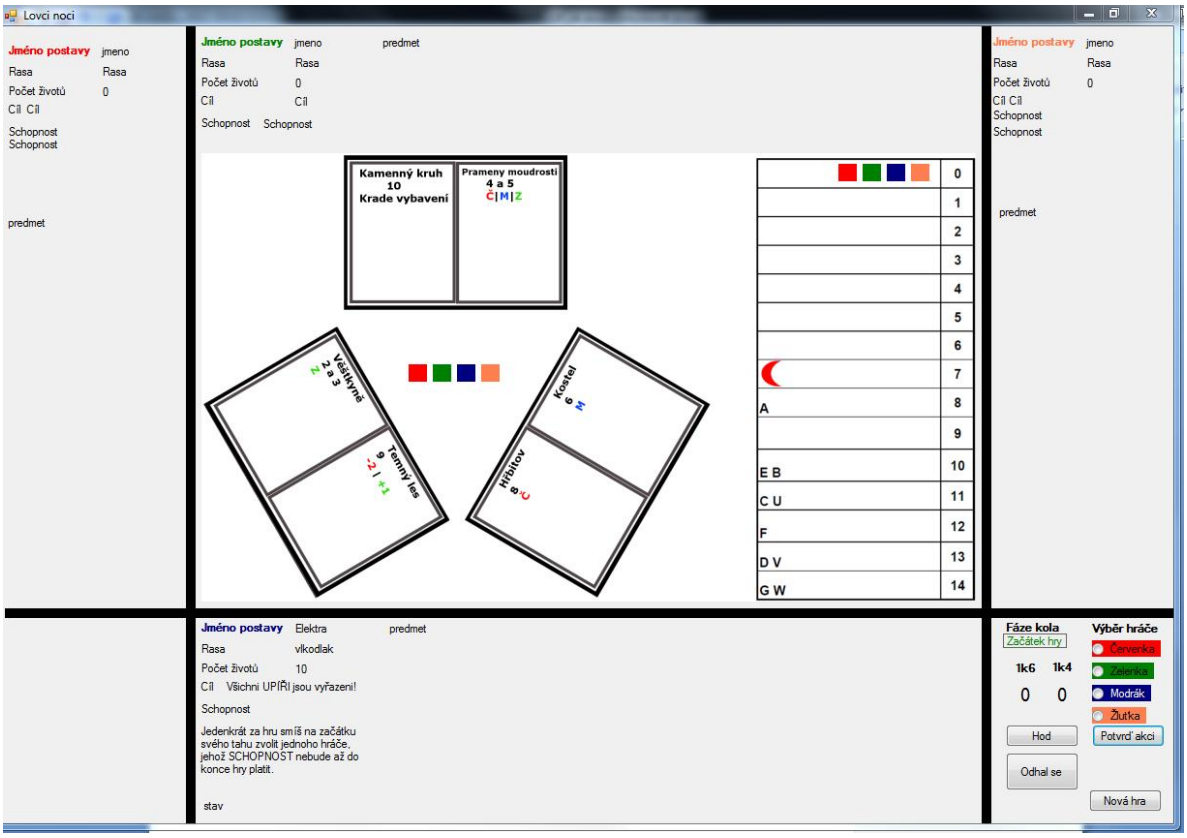
Text: text karty

Stav: hodnota 0 značí, že karta nebyla použita. Hodnota 1, že již využita byla.

Jak je vidět modré a červené karty jsou totožné, neb se liší jen ve svém účelu, ale hodnot nabývají stejných. Stav krom 0 a 1 nabývá ještě jedné hodnoty. Pokud se jedná o předmět a je ve vlastnictví některého z hráčů, zapíše se jméno vlastníka.

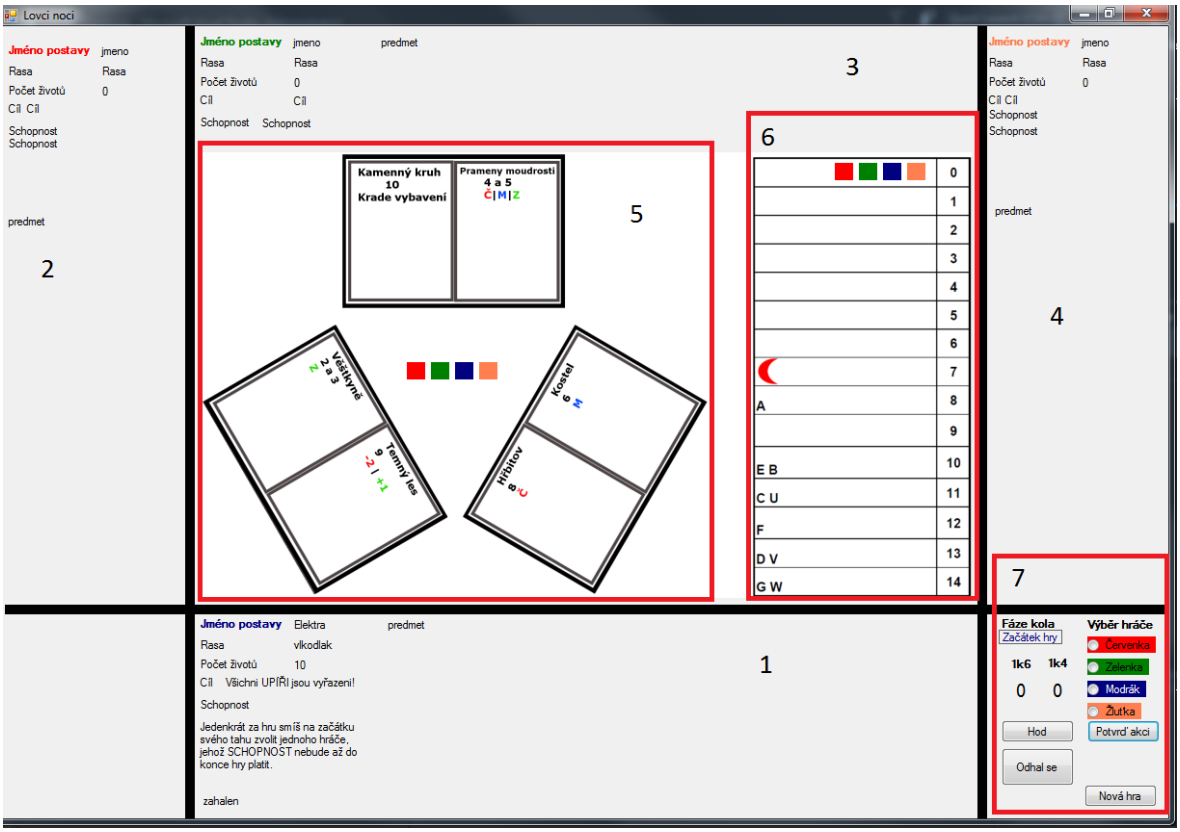
4.7 Grafické zpracování aplikace

Po startu programu se uživateli rovnou objeví herní plocha a aplikace je již rovnou připravena ke hře. Všichni hráči mají přiřazené své postavy a lidský hráč má svou postavu zobrazenou. Ostatní postavy jsou zatím zahaleny, a tak to zůstane až do doby, než se AI rozhodne svou postavu odhalit. Stejně tak hráč je pro AI zahalen, tzn. AI nevědí, za jakou stranu hraje.



Obrázek 16 - Zobrazení aplikace po spuštění

4.7.1 Popis obrazovky



Obrázek 17 - Popis spuštěné aplikace

Čísla 1 - 4 označují oblasti jednotlivých hráčů (1 - Modrák, 2 - Červenka, 3 - Zelenka, 4 - Žluťka). Oblast 1 je uživatele a oblasti 2 - 4 patří AI. Oblast 5 je vlastní herní plán, kde jsou graficky znázorněna místa spojená do jednotlivých lovišť.

Oblast 6 je počítadlo životů, respektive zranění, která jednotliví hráči během hry obdrželi. Písmena na počítadle jsou první písmena ze jmen všech postav ve hře a označují hodnotu zranění, kdy je jaká postava vyřazena ze hry. Červený měsíc na zranění 7 znázorňuje zranění, které postava utrží, pokud je cílem schopnosti postavy Freya či karty Krvavý úplněk.

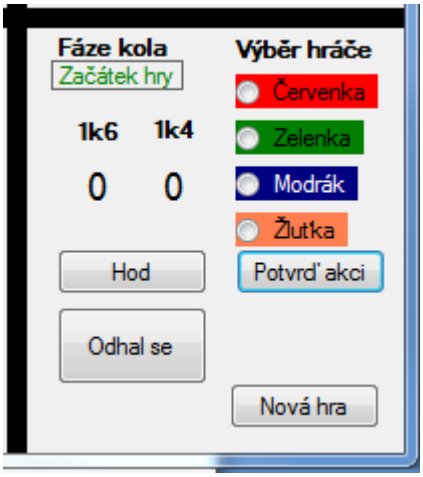
V oblasti 7 jsou ovládací prvky aplikace[9].

[9] Viz kapitola 4.8.2 Ovládání hry

4.7.2 Ovládání hry

Fáze kola zobrazují, jaká fáze kola zrovna je, což se hodí hráčům pro určení, zda mohou použít svoji schopnost. Mimo to Fáze kola zobrazuje i barvu aktuálního hráče. Tedy hráč vidí, kdo je zrovna na tahu.

Tlačítko Potvrď akci potvrzuje zvolenou akci a ukončuje danou fázi kola ať již pro hráče nebo pro AI. Je plně ovládáno hráčem, tzn. že hráč nemusí spěchat, aby zjistil, co zrovna provádí AI, neboť bez jeho zásahu a potvrzení se hra dál nepohne. Pokus s použitím časovače byl značně uživatelsky nepříjemný.



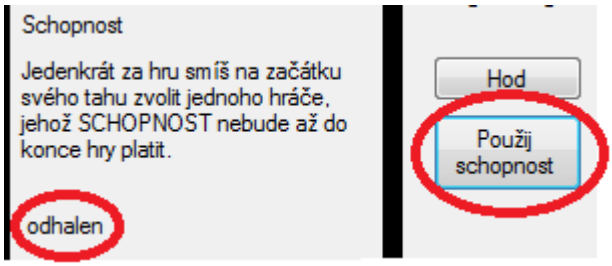
Obrázek 18 - Ovládací panel

Výběr hráče je výběr cíle akcí (cíl karet, schopností, boje a míst) pro hráče. AI toto menu nevyužívá.

Tlačítko Hod ovládá kostky (čísla pod 1k6 a 1k4), tlačítko je jen pro hráče, ale své hody na toto místo vypisuje i AI.

Tlačítko Nová hra restartuje celou aplikaci.

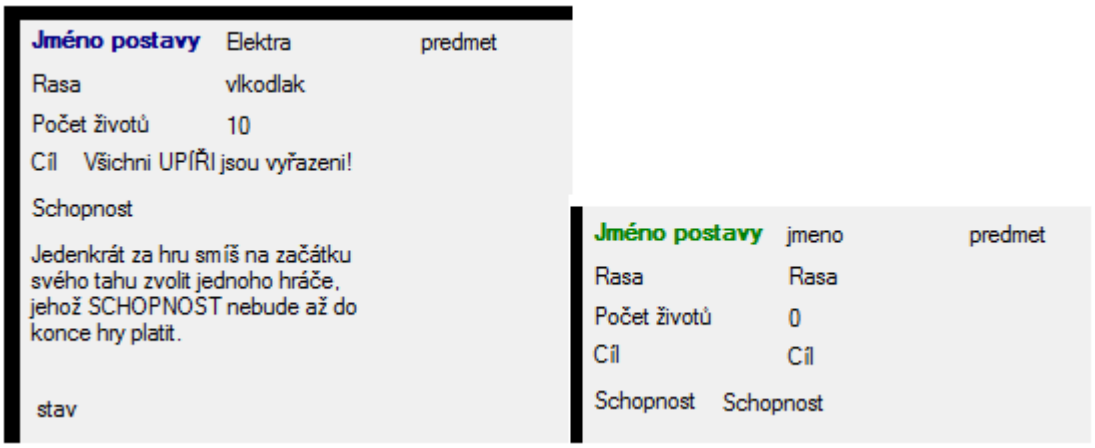
Tlačítko Odhal se, změní stav hráče na odhalen, AI si načte do paměti, za jakou stranu hráč hraje a samotné tlačítko se změní na tlačítko Použij schopnost.



Obrázek 19 - Tlačítko Použij schopnost a změna stavu

Tlačítko Použij schopnost umožňuje odhalenému hráči použít schopnost postavy, pokud se tato schopnost nepoužívá sama automaticky.

4.7.3 Rozložení pole hráče

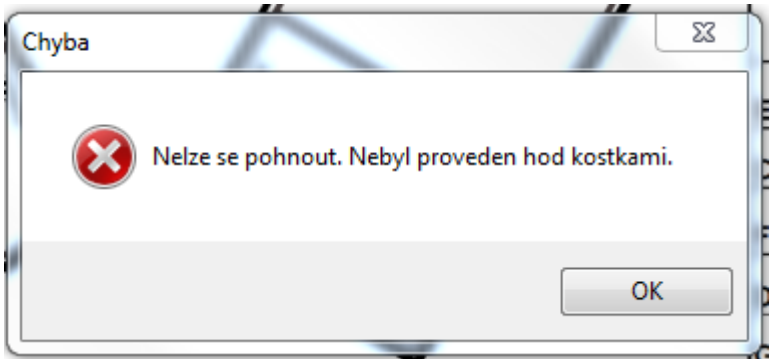


Obrázek 20 - Pole hráče + pole zahaleného AI

Každý hráč má své jméno odvozené od barvy, za kterou hraje. Hráč je Modrák a AI zastupují Červenka, Zelenka a Žluťka. Každý z hráčů má svou postavu. Její vlastnosti se vypisují do pole hráče. Modrák svou postavu rovnou vidí, neboť za něj uživatel hraje. Proto má, na rozdíl od AI, přidanou i položku "stav", která zobrazuje, v jakém stavu se jeho postava nachází (zahalen, odhalen, vyřazen).

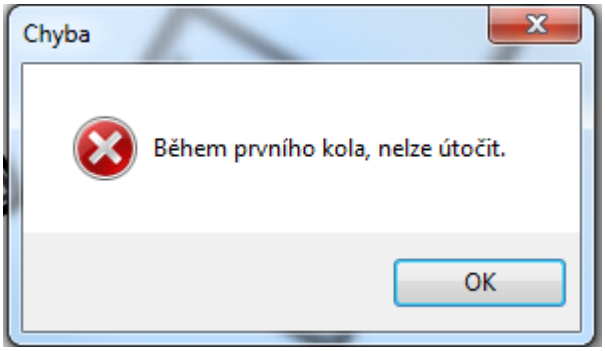
4.7.4 Běžné chybové hlášky

Běžné chybové hlášky se zobrazí hráči, ať hraje za jakoukoliv postavu. Dále existují i chybové hlášky týkající se postav a jejich schopností.



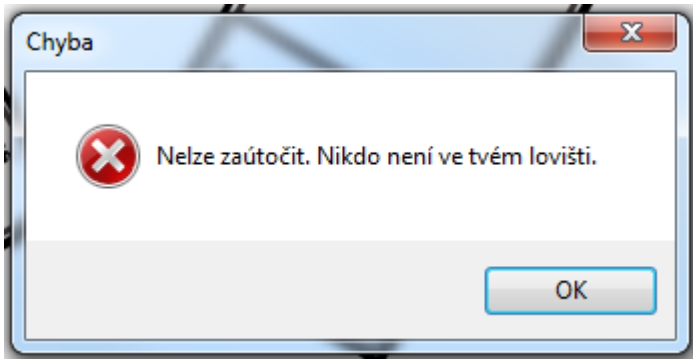
Obrázek 21 - Chyba pohyb

Tato chybová hláška (obrázek č. 22) vyskočí uživateli, pokud potvrdí akci Pohyb bez toho, aby hodil kostkami a měl vybavení Kompas.



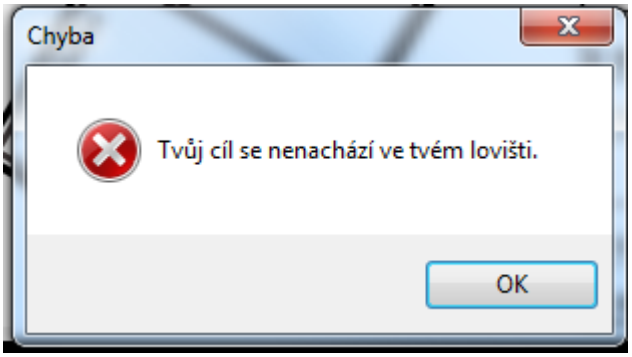
Obrázek 22 - Chyba boj v prvním kole

Tato chybová hláška (obrázek č. 23) naskočí, pokud se uživatel pokusí zaútočit během prvního kola. Takový útok pravidla nedovolují. Působit zranění pomocí karet či místa Temný les ale pravidla umožňují.



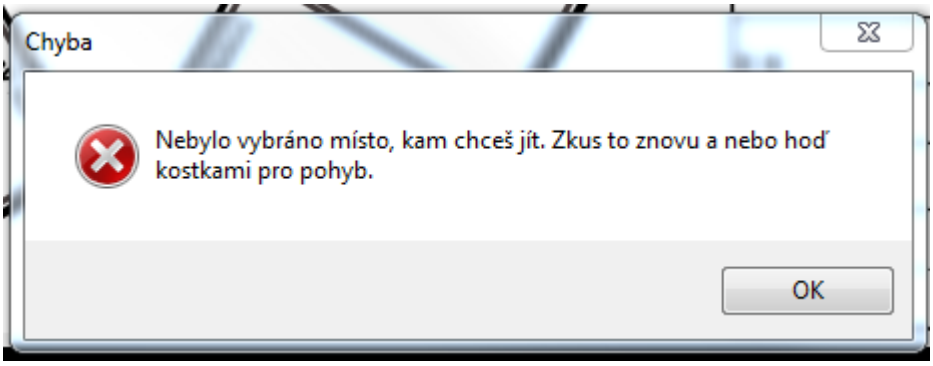
Obrázek 23 - Chyba v lovišti není žádný z hráčů

Tato chyba (obrázek č. 24) se zobrazí, pokud se uživatel pokusí zaútočit, ale nikdo z hráčů není v jeho lovišti, a to i když má předmět upravující jeho loviště (Pistole).



Obrázek 24 - Chyba vybraný cíl je mimo dosah

Tuto chybovou hlášku (obrázek č. 25) uživatel uvidí, pokud vybere cíl útoku, ale tento cíl je mimo jeho loviště. Aby se zobrazila, musí mít jiného z hráčů ve svém lovišti.



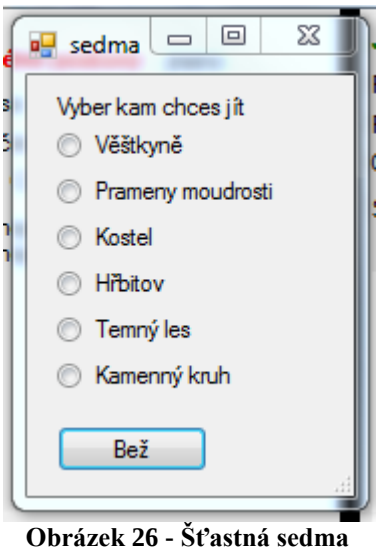
Obrázek 25 - Chyba nebyla vybrána lokace pro přesun

Tato chyba (obrázek č. 26) se zobrazí pouze v případě, že uživatel hodí na kostkách součet sedm nebo má vybavení Mystický kompas a v příslušném okně nevybere, kam chce jít.

Chybové hlášky se týkají i AI, ale ty se nevypisují uživateli na obrazovku. Bylo by to zbytečné zahlcení uživatele a kazilo by to zážitek ze samotné hry.

4.7.5 Šťastná sedma

Toto okno se zobrazí uživateli, pouze pokud hodil při fázi Pohyb na kostkách hodnotu sedm.



Obrázek 26 - Šťastná sedma

4.8 Kód aplikace

V této části práce bude postupně popsán celý kód aplikace.

4.8.1 Proměnné hlavní aplikace

V této kapitole jsou uvedeny veškeré proměnné v hlavní části aplikace včetně inicializace použitých tříd. Tato část kódu se nachází ještě před samotným vytvořením herního plánu.

```
//promenne
 Start zacatek = new Start();

 Kostka kostka;

 public int pocet_tahu;
 private int pocet_vyrazenych;
 public int x, y, z;

 public string lokaceA, lokaceNew;
 Color barvaA, barvaN;


 public string error = "";

 //pomocne promene pro mista a karty
 public int mistoHP;
 public int figurkaHP;
 public string odkladiste, akce;

 //pro urceni zraneni
 public int zran = 0;

 //zda jiz byl pripsan bonus z Kopi
 public int kopi = 0;

 //cile
 public string cil;
 public string cil1;
 public string cil2;
 public string vysledek;

 //hraci
 Hrac cervenka;
 Hrac zelenka;
 Hrac modrak;
 Hrac zlutka;

 //pohyb figurek po mape
 Pohyb pohyb;
 Postavy postava;
 Zraneni zraneni;

 //karty
 Karty karta = new Karty();
 string jmenoK, textK, typK, rasa1, rasa2, moznost1, moznost2;

 //pamet
 Pamet pamet;
```

```
        //boj
        Boj cilZran;
        Boj cilLov;
        Boj cilHled;
        Boj masakr;
        Boj overStav;

        //aktivni hrac
        public string aktivni;
        int hpA, zranA, utcA;
        string stavA, jmenoA, rasaA, mistoA, lovisteA;
        string rasaC;
        //Cil
        int hpC, zranC;
```

Jednotlivé části budou popsány níže vyjma volání třídy Karty. Tato třída je volána zde a obsahuje pouze vícerozměrná pole, ve kterých jsou zapsány karty dle popisu v datovém slovníku (kapitola 4.7 Datový slovník).

4.8.2 Hod kostkou a nová hra

Jelikož náhodný generátor čísel se používá na více místech v aplikaci, dostal vlastní třídu Kostka.

```
private void hazedlo_Click(object sender, EventArgs e)
        {
            //kostky
            kostka = new Kostka(6);
            sestka.Text = Convert.ToString(kostka.UK);

            kostka = new Kostka(4);
            ctyrka.Text = Convert.ToString(kostka.UK);
        }
```

Do třídy Kostka se posílá počet stěn kostky a následně je její výstup zpracován programem.

Třída Kostka je jednoduchým generátorem náhodných čísel v zadaném rozsahu. Jelikož se ale jedná o kostku a nechtěným výsledkem je číslo 0, které na skutečné kostce neexistuje. Proto k výsledku generátoru přičítáme jedničku.

```
class Kostka
    {
        public int UK;
        public Kostka(int PocetSten)
        {
            Random random = new Random();

            UK = random.Next(PocetSten) + 1;
        }
    }
```

Tlačítko Nová hra má jedinou funkci a to je restart celé aplikace.

```
  private void newG_Click(object sender, EventArgs e)
        {
            Application.Restart();
        }
```

4.8.3 Začátek hry a rozlosování

Po spuštění hry se při inicializaci proměnných a jednotlivých tříd spustí třída Start, které určuje, jaká barva bude začínajícím hráčem a která umisťuje figurky na jejich výchozí pozice.

```
        Start zacatek = new Start();
```

V třídě Start opět nejprve proběhne inicializace proměnných a používaných tříd a až pak je samotný kód.

```
class Start
    {
        public int f_rx, f_y, f_gx, f_bx, f_yx;
        public int hp_rx, hp_y, hp_gx, hp_bx, hp_yx;

        public Color barva;

        public int n;
        Kostka kostka;

        public Start()
        {

            //nahodny vyber zacinajiciho hrace
            kostka = new Kostka(4);
            n = kostka.UK;

            switch (n)
            {
                case 1:
                    barva = Color.Navy;
                    break;
                case 2:
                    barva = Color.Red;
                    break;
                case 3:
                    barva = Color.Green;
                    break;
                case 4:
                    barva = Color.Coral;
                    break;
            }
```

Kód dle hodu kostkou určí, která barva bude začínat. Proto bylo třeba do třídy dodat možnost kreslení.

```
using System.Drawing;
```

Dále kód určí výchozí pozice jak figurek hráčů, tak figurek na počítadle zranění. Pevně se určí umístění červené figurky a od ní se dál výpočtem odvozují pozice ostatních.

```
            //postavy hracu
            f_rx = 419;
            f_y = 349;

            f_gx = f_rx + 25;
            f_bx = f_gx + 25;
            f_yx = f_bx + 25;

            //hp
            hp_rx = 863;
            hp_y = 142;

            hp_gx = hp_rx + 25;
            hp_bx = hp_gx + 25;
            hp_yx = hp_bx + 25;

        }
    }
```

Následně pokračuje spuštění aplikace v hlavní části programu, kam se načtou hodnoty z třídy Start, tj. barva začínajícího hráče či výchozí pozice všech figurek. Do ukazatele, jaká je fáze hry, se zapíše "Začátek hry". Dále proběhne volání třídy Postavy, kde se hráčům přiřadí jejich postavy.

```
public Form1()
        {

            InitializeComponent();

            //zavolam start, ktery nastavi vychozi pozici vsech prvku

            /*
            Label do ktereho se prubezne zapisuje jaka cast kola zrovna je.
            Hrac uvidi kde je a kdo je na tahu. Stejne tak AI
            */
            fazovac.Text = "Začátek hry";

            //nahodny hrac zacne dle kostky 1k4 a kazdemu cislu bude prirazena
spravna barva pro fazovac

            fazovac.ForeColor = zacatek.barva;

            //zakladni lokace figurek
            red.Location = new Point(zacatek.f_rx, zacatek.f_y);
            green.Location = new Point(zacatek.f_gx, zacatek.f_y);
            blue.Location = new Point(zacatek.f_bx, zacatek.f_y);
            yellow.Location = new Point(zacatek.f_yx, zacatek.f_y);

            //hp
            red_hp.Location = new Point(zacatek.hp_rx, zacatek.hp_y);
            green_hp.Location = new Point(zacatek.hp_gx, zacatek.hp_y);
            blue_hp.Location = new Point(zacatek.hp_bx, zacatek.hp_y);
            yellow_hp.Location = new Point(zacatek.hp_yx, zacatek.hp_y);

            //R G Y B
            postava = new Postavy();
```

V třídě Postavy se opět prvně vytvoří veškeré použité proměnné a dvě dvojrozměrná pole. Každé z polí obsahuje postavy jedné rasy dle datového slovníku (kapitola 4.7).

```
class Postavy
    {
        public string[,] vlkodlak = {
                { "Elena","10"," Při pohybu se smíš, místo házení kostkami, přímo
přesunout na vpravo či vlevo ležící sousední MÍSTO." , "0"},
                { "Elektra","10","Jedenkrát za hru smíš na začátku svého tahu zvolit
jednoho hráče, jehož SCHOPNOST nebude až do konce hry platit." , "0"},
                { "Fenrir","12"," Jedenkrát za hru smíš na začátku svého tahu vybrat
jednoho hráče: Hoď 1k6 kostkou a způsob vybranému hráči hozený počet zranění." ,
"0"},
                { "Freya","12","Jedenkrát za hru smíš na začátku svého tahu
přesunout žeton libovolného hráče na políčko s krvavým měsícem (zranit/vyléčit na 7
životů)." , "0"},
                { "Gabrin","14"," Jedenkrát za hru smíš na začátku svého tahu vybrat
jednoho hráče: Hoď 1k4 kostkou a způsob vybranému hráči hozený počet zranění." ,
"0"},
                { "Grendel","14","Jedenkrát za hru smíš na konci svého tahu oznámit,
že ti až do začátku tvého následujícího tahu nesmí být způsobeno žádné zranění.",
"0" },
            };

        public string[,] upir = {
                {"Ultima","11", "Na začátku svého tahu smíš jednomu hráči, který se
nachází na hřbitově, způsobit 3 zranění.", "0" },
                {"Ursana","11"," Když od aktivního hráče obdržíš zelenou kartu, smíš
při odpovědi lhát. Nemusíš přitom odhalit svou kartu postavy.", "0" },
                {"Viktor","13","Způsobíš-li některému hráči při svém ÚTOKU zranění,
smíš si ihned vyléčit 2 svá vlastní zranění." , "0"},
                {"Valeria","13","Svůj ÚTOK musíš provádět pouze čtyřstěnnou kostkou.
Způsobíš tolik zranění, kolik padlo na kostce." , "0"},
                {"Wren","14","Provede-li na tebe nějaký hráč ÚTOK, smíš proti němu
ihned po tomto ÚTOKU zahájit ÚTOK.", "0" },
                {"Wilhelm","14","Jedenkrát za hru jsi po svém tahu znovu na řade. Za
každého do této chvíle vyřazeného hráče máš další kompletní tah navíc." , "0"},
            };

        public string jmeno1, hp1, rasa1, schopnost1, cil1;
        public string jmeno2, hp2, rasa2, schopnost2, cil2;
        public string jmeno3, hp3, rasa3, schopnost3, cil3;
        public string jmeno4, hp4, rasa4, schopnost4, cil4;
        private int n, vlk, pejr;
        Kostka kostka;
```

Dále jsou proměnné pro jednotlivé postavy načteny do jednorozměrných polí a postupně je hráčům dle hodu kostkou přiřazena rasa a příslušný cíl. Vybrání rasy vlkodlak je dvakrát z důvodu, že pokud první dvě postavy byly upíři a následně nepadlo na kostce správné číslo, vlkodlaci nebyli do hry vygenerováni.

```
public Postavy()
        {
            string[] jmeno = { jmeno1, jmeno2, jmeno3, jmeno4 };
            string[] hp = { hp1, hp2, hp3, hp4 };
```

```
            string[] rasa = { rasa1, rasa2, rasa3, rasa4 };
            string[] schopnost = { schopnost1, schopnost2, schopnost3, schopnost4 };
            string[] cil = { cil1, cil2, cil3, cil4 };

            for (int j = 0; j < 4; j++)
            {
                vlk = 0;
                pejr = 0;

                kostka = new Kostka(4);
                n = kostka.UK;

                if (((n == 1) | (n == 3)) & (vlk < 2))
                {
                    rasa[j] = "vlkodlak";
                    cil[j] = "Všichni UPÍŘI jsou vyřazeni!";
                    vlk = vlk + 1;
                }
                else if (pejr < 2)
                {
                    rasa[j] = "upír";
                    cil[j] = "Všichni VLKODLACI jsou vyřazeni!";
                    pejr = pejr + 1;
                }
                else if(vlk < 2)
                {
                    rasa[j] = "vlkodlak";
                    cil[j] = "Všichni UPÍŘI jsou vyřazeni!";
                    vlk = vlk + 1;
                }
```

Dle vylosované rasy se náhodně zvolí postava a její stav se změní na 1, tedy použitá. Díky tomu nedochází k opakování postav ve hře.

```
                if (rasa[j] == "upír")
                {
                    do
                    {
                        kostka = new Kostka(6);
                        n = kostka.UK - 1;
                        if (upir[n, 3] == "0")
                        {
                            jmeno[j] = upir[n, 0];
                            hp[j] = upir[n, 1];
                            schopnost[j] = upir[n, 2];
                        }

                    }
                    while (upir[n, 3] == "1");
                    upir[n, 3] = "1";

                }

                if (rasa[j] == "vlkodlak")
                {
                    do
                    {
                        kostka = new Kostka(6);
                        n = kostka.UK - 1;
                        if (vlkodlak[n, 3] == "0")
```

```
                    {
                        jmeno[j] = vlkodlak[n, 0];
                        hp[j] = vlkodlak[n, 1];
                        schopnost[j] = vlkodlak[n, 2];
                    }
                }
                while (vlkodlak[n, 3] == "1");
                vlkodlak[n, 3] = "1";
            }
        }
```

Nakonec se hodnoty z jednorozměrných polí načtou do proměnných připravených pro jednotlivé hráče a následuje návrat do hlavního kódu.

```
            //cervenka
            jmeno1 = jmeno[0];
            hp1 = hp[0];
            rasa1 = rasa[0];
            schopnost1 = schopnost[0];
            cil1 = cil[0];
            //zelenka
            jmeno2 = jmeno[1];
            hp2 = hp[1];
            rasa2 = rasa[1];
            schopnost2 = schopnost[1];
            cil2 = cil[1];
            //modrak
            jmeno3 = jmeno[2];
            hp3 = hp[2];
            rasa3 = rasa[2];
            schopnost3 = schopnost[2];
            cil3 = cil[2];
            //zlutka
            jmeno4 = jmeno[3];
            hp4 = hp[3];
            rasa4 = rasa[3];
            schopnost4 = schopnost[3];
            cil4 = cil[3];
        }
    }
}
```

V hlavním kódu se hodnoty z třídy Postavy přiřadí jednotlivým hráčům pomocí třídy Hrac.

```
            cervenka = new Hrac("cervenka", postava.jmeno1, postava.hp1,
postava.cil1,
                postava.schopnost1, postava.rasa1, Color.Red, Color.Green);

            zelenka = new Hrac("zelenka", postava.jmeno2, postava.hp2, postava.cil2,
                postava.schopnost2, postava.rasa2, Color.Green, Color.Coral);

            modrak = new Hrac("modrak", postava.jmeno3, postava.hp3, postava.cil3,
                postava.schopnost3, postava.rasa3, Color.Navy, Color.Red);

            zlutka = new Hrac("zlutka", postava.jmeno4, postava.hp4, postava.cil4,
                postava.schopnost4, postava.rasa4, Color.Coral, Color.Navy);
```

Do této třídy byla opět přidána možnost kreslení kvůli barvě samotného hráče a barvě hráče, který následuje při hře po něm.

```
using System.Drawing;
```

Opět se prvně v třídě vytvoří používané proměnné a následně jsou naplněny dle hodnot hráče, které mu byly přiřazeny v třídě Postavy. Ty jsou doplněny dalšími hodnotami, které jsou pro hru důležité.

```
class Hrac
    {
        //kdo to je
        public string hrac;
        //postava
        public string jmeno;
        public string zivoty;
        public string cil;
        public string schopnost;
        public string rasa;
        //zmeny v prubehu hry
        public string stav;
        public int zraneni;
        public int utocneC;
        public string misto;
        public string loviste;
        public int schop;

        //barva aktivniho hrace, barva nasledujiciho hrace
        public Color barvaA;
        public Color barvaN;

        //pamet
        public string pametR, pametG, pametB, pametY;


        public Hrac(string hrac, string jmeno, string zivoty, string cil, string
schopnost, string rasa,Color barvaA,
            Color barvaN)
         {
            this.hrac = hrac;

            this.jmeno = jmeno;
            this.zivoty = zivoty;
            this.cil = cil;
            this.schopnost = schopnost;
            this.rasa = rasa;

            this.barvaA = barvaA;
            this.barvaN = barvaN;

            stav = "zahalen";
            zraneni = 0;
            misto = "N/A";
            loviste = "N/A";
            utocneC = 0;
            schop = 0;

            pametR = "";
```

```
            pametG = "";
            pametB = "";
            pametY = "";
        }
    }
```

V hlavním kódu následuje vypsání hodnot uživatele na příslušné místo a načtení vlastní rasy do paměti AI.

```
            jmenoB.Text = modrak.jmeno;
            rasaB.Text = modrak.rasa;
            cilB.Text = modrak.cil;
            hpB.Text = modrak.zivoty;
            schopnostB.Text = modrak.schopnost;
            modrak.pametB = modrak.rasa;
            stavB.Text = modrak.stav;
//Pridani do pameti vlastni rasa
            cervenka.pametR = cervenka.rasa;
            zelenka.pametG = zelenka.rasa;
            zlutka.pametY = zlutka.rasa;
        }
```

Tím je hra připravena pro své první kolo. Všichni hráči mají přiřazené postavy, AI má první hodnotu ve své paměti a všechny figurky jsou na svých místech.

4.8.4 Fáze kola

Jak již bylo popsáno dříve, kolo se dělí na několik fází. V reálné aplikaci byla oproti pravidlům přidána fáze Událost, kde se řeší, co se stalo ve fázi Místo. Veškeré fáze se řeší pomocí tlačítka Potvrď akci a některé události se vyhodnocuji vždy při jeho aktivaci.

První, co aplikace řeší, je vypsání hodnot AI hráčů, pokud se odhalili. Následně určuje hodnoty aktivního hráče a nakonec ověřuje, zda aktivní hráč není náhodou vyřazen. Pokud vyřazen je, jsou jeho figurky vyhozeny mimo plán.

```
private void tmbod_Click(object sender, EventArgs e)
        {
            #region Odhaleni
            if (cervenka.stav == "odhalen")
            {
                jmenoR.Text = cervenka.jmeno;
                rasaR.Text = cervenka.rasa;
                cilR.Text = cervenka.cil;
                hpR.Text = cervenka.zivoty;
                schopnostR.Text = cervenka.schopnost;
                cervenka.pametR = cervenka.rasa;
            }

            if (zelenka.stav == "odhalen")
            {
                jmenoG.Text = zelenka.jmeno;
                rasaG.Text = zelenka.rasa;
                cilG.Text = zelenka.cil;
                hpG.Text = zelenka.zivoty;
```

```
                schopnostG.Text = zelenka.schopnost;
                zelenka.pametG = zelenka.rasa;
            }

            if (zlutka.stav == "odhalen")
            {
                jmenoY.Text = zlutka.jmeno;
                rasaY.Text = zlutka.rasa;
                cilY.Text = zlutka.cil;
                hpY.Text = zlutka.zivoty;
                schopnostY.Text = zlutka.schopnost;
                zlutka.pametY = zlutka.rasa;
            }
            #endregion

#region Aktivni hrac
            //modrak - clovek
            if ((fazovac.ForeColor == Color.Navy) & (modrak.stav != "vyrazen"))
            {
                aktivni = "modrak";
                barvaN = modrak.barvaN;
                barvaA = modrak.barvaA;
                stavB.Text = modrak.stav;
            }
            else if ((fazovac.ForeColor == Color.Navy) & (modrak.stav == "vyrazen"))
            {
                fazovac.Text = "Pohyb";
                fazovac.ForeColor = modrak.barvaN;
                blue.Location = new Point(zacatek.f_bx, zacatek.f_y);
                blue_hp.Location = new Point(zacatek.f_bx, zacatek.f_y);
                sestka.Text = "0";
                ctyrka.Text = "0";
                return;

            }

            //Cervenka
            if ((fazovac.ForeColor == Color.Red) & (cervenka.stav != "vyrazen"))
            {
                aktivni = "cervenka";
                barvaN = cervenka.barvaN;
                barvaA = cervenka.barvaA;
            }
            else if ((fazovac.ForeColor == Color.Red) & (cervenka.stav ==
"vyrazen"))
            {
                fazovac.Text = "Pohyb";
                fazovac.ForeColor = cervenka.barvaN;
                red.Location = new Point(zacatek.f_rx, zacatek.f_y);
                red_hp.Location = red.Location;//new Point(zacatek.f_rx,
zacatek.f_y);
                sestka.Text = "0";
                ctyrka.Text = "0";
                return;
            }

            //zelenka
            if ((fazovac.ForeColor == Color.Green) & (zelenka.stav != "vyrazen"))
            {
```

```
                aktivni = "zelenka";
                barvaN = zelenka.barvaN;
                barvaA = zelenka.barvaA;
            }
            else if ((fazovac.ForeColor == Color.Green) & (zelenka.stav ==
"vyrazen"))
            {
                fazovac.Text = "Pohyb";
                fazovac.ForeColor = zelenka.barvaN;
                green.Location = new Point(zacatek.f_gx, zacatek.f_y);
                green_hp.Location = new Point(zacatek.f_gx, zacatek.f_y);
                sestka.Text = "0";
                ctyrka.Text = "0";
                return;

            }

            //zlutka
            if ((fazovac.ForeColor == Color.Coral) & (zlutka.stav != "vyrazen"))
            {
                aktivni = "zlutka";
                barvaN = zlutka.barvaN;
                barvaA = zlutka.barvaA;
            }
            else if ((fazovac.ForeColor == Color.Coral) & (zlutka.stav ==
"vyrazen"))
            {
                fazovac.Text = "Pohyb";
                //nahozeni kostek do puvodni pozice kvuli osetreni chyby
                sestka.Text = "0";
                ctyrka.Text = "0";
                fazovac.ForeColor = zlutka.barvaN;
                yellow.Location = new Point(zacatek.f_yx, zacatek.f_y);
                yellow_hp.Location = new Point(zacatek.f_yx, zacatek.f_y);
                return;
            }
            #endregion
            #region Vyrazeni
            if (modrak.stav == "vyrazen")
            {
                blue.Location = new Point(zacatek.f_bx, zacatek.f_y);
                blue_hp.Location = new Point(zacatek.f_bx, zacatek.f_y);
            }

            //Cervenka
           if(cervenka.stav == "vyrazen")
            {
                red.Location = new Point(zacatek.f_rx, zacatek.f_y);
                red_hp.Location = red.Location;

            }

            //zelenka
           if (zelenka.stav == "vyrazen")
            {
                green.Location = new Point(zacatek.f_gx, zacatek.f_y);
                green_hp.Location = new Point(zacatek.f_gx, zacatek.f_y);

            }
```

```
            //zlutka
           if  (zlutka.stav == "vyrazen")
            {
                yellow.Location = new Point(zacatek.f_yx, zacatek.f_y);
                yellow_hp.Location = new Point(zacatek.f_yx, zacatek.f_y);
            }
            #endregion
```

Dále proběhne příprava a aktualizace jednotlivých jednorozměrných polí, která se využívají dále v kódu v jednotlivých fázích. Také je určena hodnota x, která odkazuje na pořadové číslo aktivního hráče v rámci těchto polí.

```
//zjisteni hodnot aktualniho hrace
            string[] hraci = { "cervenka", "zelenka", "modrak", "zlutka" };
            string[] cile = { "cervenka", "zelenka", "modrak", "zlutka" };
            string[] jmeno = { cervenka.jmeno, zelenka.jmeno, modrak.jmeno,
zlutka.jmeno };
            string[] rasa = { cervenka.rasa, zelenka.rasa, modrak.rasa, zlutka.rasa
};
            string[] stav = { cervenka.stav, zelenka.stav, modrak.stav, zlutka.stav
};
            string[] hp = { cervenka.zivoty, zelenka.zivoty, modrak.zivoty,
zlutka.zivoty };
            string[] schopnostiT = { cervenka.schopnost, zelenka.schopnost,
modrak.schopnost, zlutka.schopnost };
            int[] zran = { cervenka.zraneni, zelenka.zraneni, modrak.zraneni,
zlutka.zraneni };
            int[] utc = { cervenka.utocneC, zelenka.utocneC, modrak.utocneC,
zlutka.utocneC };
            string[] loviste = { cervenka.loviste, zelenka.loviste, modrak.loviste,
zlutka.loviste };
            string[] misto = { cervenka.misto, zelenka.misto, modrak.misto,
zlutka.misto };
            string[] pametR = { cervenka.pametR, cervenka.pametG, cervenka.pametB,
cervenka.pametY };
            string[] pametG = { zelenka.pametR, zelenka.pametG, zelenka.pametB,
zelenka.pametY };
            string[] pametY = { zlutka.pametR, zlutka.pametG, zlutka.pametB,
zlutka.pametY };
            int[] schopnosti = {cervenka.schop, zelenka.schop, modrak.schop,
zlutka.schop };

            x = 0;
            y = 0;

            while (aktivni != hraci[x])
            {
                x++;
            }
```

Jako poslední před řešením samostatných fází kola se dle aktivního hráče zvolí hledání cíle pro AI podle aktuální paměti, která odkazuje na třídu Boj.

```
switch (aktivni)
            {
                case "cervenka":
                    cilHled = new Boj(aktivni, cervenka.pametR, cervenka.pametG,
cervenka.pametB, cervenka.pametY);
```

```
                    break;
                case "zelenka":
                    cilHled = new Boj(aktivni, zelenka.pametR, zelenka.pametG,
zelenka.pametB, zelenka.pametY);
                    break;
                case "modrak":
                    cilHled = new Boj(aktivni, modrak.pametR, modrak.pametG,
modrak.pametB, modrak.pametY);
                    break;
                case "zlutka":
                    cilHled = new Boj(aktivni, zlutka.pametR, zlutka.pametG,
zlutka.pametB, zlutka.pametY);
                    break;
            }
```

V třídě Boj se řeší veškeré algoritmy týkající se boje hráčů mezi sebou ať již útokem nebo kartami. Tato třída bude ještě v práci několikrát zmíněna. Část, na kterou se výše zmíněný kód odkazuje, pouze porovnává rasu aktivního hráče s rasami ostatních. Pokud o ostatních nic neví, označí je za cíl. Jelikož ví, že má jednoho spojence a dva protihráče, vybírá cíl 1 a cíl 2.

```
public Boj(string aktivni, string rasaR, string rasaG, string rasaB, string rasaY)
        {
            string[] hraci = { "cervenka", "zelenka", "modrak", "zlutka" };
            string[] rasy = { rasaR, rasaG, rasaB, rasaY };
            cil1 = "";

            while (rasaA == null)
            {

                if (hraci[i] == aktivni)
                    rasaA = rasy[i];
                else
                    i++;
            }

            for (int j = 0; j < 4; j++)
            {
                if (rasaA != rasy[j])
                {
                    if (cil1 == "")
                        cil1 = hraci[j];
                    else
                        cil2 = hraci[j];
                }
            }
        }
```

Cíle se načtou v hlavním kódu a následně se z nich opět v třídě Boj vybere cíl s větším zraněním. Pokud je zranění stejné, cílem je cíl 1.

```
            cil1 = cilHled.cil1;
            cil2 = cilHled.cil2;
//vyber cile s vetsim zranenim, pokud je stejne nacti cil1
            cilZran = new Boj(cil1, cil2, cervenka.zraneni, cervenka.zraneni,
modrak.zraneni, zlutka.zraneni);
            cil = cilZran.cil;
```

Porovnání zranění cílů v třídě Boj.

```
public Boj (string cil1, string cil2, int zranR, int zranG, int zranB, int zranY)
        {
            i = 0;
            j = 0;

            if ((cil1 == "cervenka") | (cil2 == "cervenka"))
                if (i == 0)
                    i = zranR;
                else
                    j = zranR;

            if ((cil1 == "zelenka") | (cil2 == "zelenka"))
                if (i == 0)
                    i = zranG;
                else
                    j = zranG;

            if ((cil1 == "modrak") | (cil2 == "modrak"))
                if (i == 0)
                    i = zranB;
                else
                    j = zranB;

            if ((cil1 == "zlutka") | (cil2 == "zlutka"))
                if (i == 0)
                    i = zranY;
                else
                    j = zranY;

            if (i >= j)
                cil = cil1;
            else
                cil = cil2;
        }
```

4.8.4.1 Pohyb

Pohyb je první fází kola, řeší se v něm schopnosti některých postav (bude popsáno v kapitole 4.9.5) a efekt několika karet.

Prvně se zjistí, jaká fáze kola je, následně se přičte hodnota jedna k počítadlu tahů, pak se vyvolá třída Pohyb, která zpracuje hodnoty aktivního hráče a umístění všech hráčů.

```
if ((fazovac.Text == "Začátek hry") | (fazovac.Text == "Pohyb"))
{
                pocet_tahu++;
                pohyb   =   new   Pohyb(aktivni,   cervenka.misto,   zelenka.misto,
modrak.misto, zlutka.misto);
```

Pomocí tohoto kódu se určí, kde aktivní hráč právě stojí. To je důležité při jeho přesunu, neboť pravidla neumožňují zůstat více kol na jednom místě.

```
public Pohyb(string aktivni, string mistoR, string mistoG, string mistoB, string
mistoY)
        {
            string[] hraci = { "cervenka", "zelenka", "modrak", "zlutka" };
```

```
            string[] misto = { mistoR, mistoG, mistoB, mistoY};

            x = 0;
            mistoA = null;

            while (mistoA == null)
            {

                if (hraci[x] == aktivni)
                    mistoA = misto[x];
                else
                    x++;
            }
        }
```

V hlavním kódu proběhne načtení lokace aktivního hráče a vynulování některých proměnných používaných později. Dále se řeší, zda aktivní hráč má aktivní kartu Anděl strážný a pokud tomu tak je, změní se stav karty na použitá.

```
                lokaceA = pohyb.mistoA;
                mistoHP = 0;

                //zmizeni andela
                if (aktivni == karta.modra[0, 3])
                    karta.modra[0, 3] = "1";
```

Následně se řeší pohyb hráčů po mapě včetně řešení efektu karty Mystický kompas.

```
              do
                {
                    //Kompas
                    if (aktivni == karta.modra[15, 3])
                    {
                        sestka.Text = "4";
                        ctyrka.Text = "3";
                    }

                    //hod kostkami na pohyb
                    if (aktivni == "modrak")
                    {
                        Error error = new Error(sestka.Text, ctyrka.Text);

                        if (error.chybka != "")
                        {
                            MessageBox.Show(error.chybka, "Chyba",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                            return;
                        }
                    }
```

Pokud hráč nemá kompas a nehodil kostkami, program mu vypíše chybovou hlášku, která ho na tuto skutečnost upozorní. Běžné chybové hlášky jsou ve třídě Error.

```
public Error(string sestK, string ctyriK)
        {
            if (sestK == "0" || ctyriK == "0")
            {
                chybka = "Nelze se pohnout. Nebyl proveden hod kostkami.";
            }
```

```
            else
                chybka = "";
        }
```

AI si hází kostkou samo. Následně je hodnota hodu přenesena do třídy Pohyb, kde se uskuteční vlastní řešení pohybu.

```
//reseni AI pokud nema Kompas
                    else if ((aktivni != "modrak") & (aktivni != karta.modra[15, 3]))
                     {
                         kostka = new Kostka(6);
                         sestka.Text = Convert.ToString(kostka.UK);
                         kostka = new Kostka(4);
                         ctyrka.Text = Convert.ToString(kostka.UK);
                     }

                     pohyb = new Pohyb(sestka.Text, ctyrka.Text, aktivni);
```

První, co se v této třídě ověřuje, je, zda aktivní hráč nehodil náhodou na kostkách číslo sedm. Toto číslo umožňuje zvolit si dle pravidel hry místo, na které se figurka přesune. AI volí náhodnou lokaci a hráči se zobrazí oznámení ve formě windows form sedma, které mu umožňuje výběr.

```
class Pohyb
    {
        public int x, y, z;
        public int ry;
        public string misto;
        public string mistoA;
        Kostka kostka;
        public string lokaceR, lokaceG, lokaceB, lokaceY;


        public Pohyb(string sest, string ctyri, string aktivni)
        {
            x = Convert.ToInt32(sest);
            y = Convert.ToInt32(ctyri);

            z = x + y;
            //stastna sedma
            if (z == 7)
            {
                //pro zatim nez bude AI vnimat hru
                if (aktivni != "modrak")
                {
                    do
                    {
                        kostka = new Kostka(6);
                        x = kostka.UK;
                        kostka = new Kostka(4);
                        y = kostka.UK;
                        z = x + y;
                    }
                    while (z == 7);
                }

                if (aktivni == "modrak")
                {
```

```
                    //novy form s vyberem mista a odeslanim hodnoty z nej sem pro
dalsi vyuziti
                    sedma s = new sedma();

                    //showdialog zobrazi vyber a pocka do jeho ukonceni
                    s.ShowDialog();
                    if (s.hodnotaB == null)
                    {
                        misto = null;
                    }
                    else
                    z = Convert.ToInt32(s.hodnotaB);
                }
            }
```

Windows form sedma je řešen velmi prostě. Uživatel si vybere, kam chce jít, a dle toho se do třídy Pohyb pošle číslo lokace.

```
public partial class sedma : Form
    {
        public sedma()
        {
            InitializeComponent();
        }

        public string hodnotaB = null;

        private void ok_Click(object sender, EventArgs e)
        {
            if (t1.Checked) //vestkyne
            {
                hodnotaB = "2";
            }

            if (t2.Checked) //prameny
            {
                hodnotaB = "4";
            }

            if (t3.Checked) //kostel
            {
                hodnotaB = "6";
            }

            if (t4.Checked) //hrbitov
            {
                hodnotaB = "8";
            }

            if (t5.Checked) //les
            {
                hodnotaB = "9";
            }

            if (t6.Checked) // kruh
            {
                hodnotaB = "10";
            }
        }
```

Následně se dle čísla lokace, ať již z kostek či dle sedmy, vybere lokace, kam se figurka aktivního hráče přesune.

```
            //vestkyne
            if ((z == 2) | (z == 3))
            {
                x = 249;
                ry = 390;
                //zbytecne ale sichr je sichr
                if (aktivni == "cervenka")
                {
                    x = x;
                }

                if (aktivni == "zelenka")
                {
                    x = x + 25;
                }

                if (aktivni == "modrak")
                {
                    x = x + 50;
                }

                if (aktivni == "zlutka")
                {
                    x = x + 75;
                }
                misto = "vestkyne";
            }

            //Prameny moudrosti
            if ((z == 4) | (z == 5))
            {
                x = 476;
                ry = 202;
                //zbytecne ale sichr je sichr
                if (aktivni == "cervenka")
                {
                    x = x;
                }

                if (aktivni == "zelenka")
                {
                    x = x + 25;
                }

                if (aktivni == "modrak")
                {
                    x = x + 50;
                }

                if (aktivni == "zlutka")
                {
                    x = x + 75;
                }
                misto = "prameny";
            }
```

```
            //kostel
            if (z == 6)
            {
                x = 600;
                ry = 402;
                //zbytecne ale sichr je sichr
                if (aktivni == "cervenka")
                {
                    x = x;
                }

                if (aktivni == "zelenka")
                {
                    x = x + 25;
                }

                if (aktivni == "modrak")
                {
                    x = x + 50;
                }

                if (aktivni == "zlutka")
                {
                    x = x + 75;
                }
                misto = "kostel";
            }

            //hřbitov
            if (z == 8)
            {
                x = 540;
                ry = 502;
                //zbytecne ale sichr je sichr
                if (aktivni == "cervenka")
                {
                    x = x;
                }

                if (aktivni == "zelenka")
                {
                    x = x + 25;
                }

                if (aktivni == "modrak")
                {
                    x = x + 50;
                }

                if (aktivni == "zlutka")
                {
                    x = x + 75;
                }
                misto = "hrbitov";
            }

            //temný les
            if (z == 9)
            {
```

```
            x = 302;
            ry = 492;

            //zbytecne ale sichr je sichr
            if (aktivni == "cervenka")
            {
                x = x;
            }

            if (aktivni == "zelenka")
            {
                x = x + 25;
            }

            if (aktivni == "modrak")
            {
                x = x + 50;
            }

            if (aktivni == "zlutka")
            {
                x = x + 75;
            }
            misto = "les";
        }

        //kamenný kruh
        if (z == 10)
        {
            x = 366;
            ry = 211;
            //zbytecne ale sichr je sichr
            if (aktivni == "cervenka")
            {
                x = x;
            }

            if (aktivni == "zelenka")
            {
                x = x + 25;
            }

            if (aktivni == "modrak")
            {
                x = x + 50;
            }

            if (aktivni == "zlutka")
            {
                x = x + 75;
            }
            misto = "kruh";
        }

        y = ry;
    }
```

Nová lokace se v hlavním kódu načte do příslušné proměnné a ověří se, zda je nová lokace odlišná od současné. Pokud tomu tak není, je uživatel vyzván, aby hodil znovu. AI

si hází rovnou samo. Pokud uživatel při vyvolání windows form sedm nevybere žádnou lokaci pro přesun, je na to též upozorněn.

```
                    lokaceNew = pohyb.misto;
                    //Rozlisuje ktery error cilovat
                    x++;
                    if (aktivni == "modrak")
                    {
                        Error error = new Error(lokaceA, lokaceNew, x);

                        if (error.chybka != "")
                        {
                            MessageBox.Show(error.chybka, "Chyba",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                            return;
                        }
                    }

                } while (lokaceNew == lokaceA);

                if(lokaceNew == null)
                {
                    MessageBox.Show("Nebylo vybráno místo, kam chceš jít. Zkus to
znovu a nebo hoď kostkami pro pohyb.", "Chyba", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    return;
                }
```

Až nyní se hýbe figurka aktivního hráče po herním plánu. A to podle hodnot, které byly určeny dle místa ve třídě Pohyb.

```
            switch(aktivni)
              {
                  case "cervenka":
                      red.Location = new Point(pohyb.x, pohyb.y);
                      cervenka.misto = lokaceNew;
                      break;
                  case "zelenka":
                      green.Location = new Point(pohyb.x, pohyb.y);
                      zelenka.misto = lokaceNew;
                      break;
                  case "modrak":
                      blue.Location = new Point(pohyb.x, pohyb.y);
                      modrak.misto = lokaceNew;
                      break;
                  case "zlutka":
                      yellow.Location = new Point(pohyb.x, pohyb.y);
                      zlutka.misto = lokaceNew;
                      break;
              }
```

Na konci fáze se řeší, zda byl někdo z hráčů zraněn (v této části pouze schopností postav).

```
            switch (cil)
              {
                  case "cervenka":
                      cervenka.zraneni = cervenka.zraneni + mistoHP;
```

```
                        zraneni = new Zraneni(cil, cervenka.zraneni,
zelenka.zraneni, modrak.zraneni, zlutka.zraneni);
                        red_hp.Location = new Point(zacatek.hp_rx, zraneni.y);
                        break;
                    case "zelenka":
                        zelenka.zraneni = zelenka.zraneni + mistoHP;
                        zraneni = new Zraneni(cil, cervenka.zraneni,
zelenka.zraneni, modrak.zraneni, zlutka.zraneni);
                        green_hp.Location = new Point(zacatek.hp_gx, zraneni.y);
                        break;
                    case "modrak":
                        modrak.zraneni = modrak.zraneni + mistoHP;
                        zraneni = new Zraneni(cil, cervenka.zraneni,
zelenka.zraneni, modrak.zraneni, zlutka.zraneni);
                        blue_hp.Location = new Point(zacatek.hp_bx, zraneni.y);
                        break;
                    case "zlutka":
                        zlutka.zraneni = zlutka.zraneni + mistoHP;
                        zraneni = new Zraneni(cil, cervenka.zraneni,
zelenka.zraneni, modrak.zraneni, zlutka.zraneni);
                        yellow_hp.Location = new Point(zacatek.hp_yx, zraneni.y);
                        break;
                }
```

Zranění a následný posun figurek po počítadle zranění se řeší v třídě Zraneni.

```
class Zraneni
    {
        public int y, x, z, p, zraneni;
        public string vysledek;
        private long v;

        public Zraneni(string cil, int zraneniR, int zraneniG, int zraneniB, int
zraneniY)
        {
            x = 0;
            y = 142;


            z = 30;
            p = 0;
            zraneni = 0;

            if (cil == "cervenka")
                zraneni = zraneniR;
            if (cil == "zelenka")
                zraneni = zraneniG;
            if (cil == "modrak")
                zraneni = zraneniB;
            if (cil == "zlutka")
                zraneni = zraneniY;

            //0
            do
            {
                x = (p * z);
                p++;

            } while ((zraneni >= p) & (zraneni < 16));

            y = y + x;
```

```
        }
```

Následně se do této třídy z hlavního kódu posílají data pro ověření, zda nebyl někdo z hráčů náhodou vyřazen.

```
zraneni = new Zraneni(Convert.ToInt32(cervenka.zivoty),
Convert.ToInt32(zelenka.zivoty), Convert.ToInt32(modrak.zivoty),
Convert.ToInt32(zlutka.zivoty), cervenka.zraneni, zelenka.zraneni, modrak.zraneni,
zlutka.zraneni);
```

Ověření je prosté, pokud hráčovo zranění je rovno či vyšší než životy jeho postavy, je vyřazen.

```
public Zraneni(int hpR, int hpG, int hpB, int hpY, int zranR, int zranG, int zranB,
int zranY)
        {
            vysledek = "";

            if (zranR >= hpR)
                vysledek = "Červenka";
            if (zranG >= hpG)
                vysledek = "Zelenka";
            if (zranB >= hpB)
                vysledek = "Modrák";
            if (zranY >= hpY)
                vysledek = "Žluťka";
        }
```

Pokud je hráč vyřazen, jeho figurky se přesunou doprostřed herního plánu, je mu změněn stav, místo a loviště, aby nemohl být cílem uživatele či AI a všem jeho předmětům je změněn stav na 1, tedy jsou přesunuty do odhazovacího balíčku. Uživateli je vypsána hláška o tom, který z hráčů byl vyřazen ze hry.

```
                vysledek = zraneni.vysledek;
                if (vysledek != "")
                {
                    if ((vysledek == "Červenka") & (cervenka.stav != "vyrazen"))
                    {
                        red.Location = new Point(zacatek.f_rx, zacatek.f_y);
                        red_hp.Location = new Point(zacatek.f_rx, zacatek.f_y);
                        cervenka.stav = "vyrazen";
                        cervenka.misto = "N/A";
                        cervenka.loviste = "N/A";
                        for (int i = 10; i < 16; i++)
                        {
                            if (karta.modra[i, 3] == "cervenka")
                                karta.modra[i, 3] = "1";
                            if (karta.cervena[i, 3] == "cervenka")
                                karta.cervena[i, 3] = "1";
                        }
                        MessageBox.Show("Hráč " + vysledek + " byl vyřazen ze hry.",
"Hráč vyřazen", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    }
                    if ((vysledek == "Zelenka") & (zelenka.stav != "vyrazen"))
                    {
                        green.Location = new Point(zacatek.f_gx, zacatek.f_y);
                        green_hp.Location = new Point(zacatek.f_gx, zacatek.f_y);
                        green_hp.Location = green.Location;
                        zelenka.stav = "vyrazen";
```

```
                    zelenka.misto = "N/A";
                    zelenka.loviste = "N/A";
                    for (int i = 10; i < 16; i++)
                    {
                        if (karta.modra[i, 3] == "zelenka")
                            karta.modra[i, 3] = "1";
                        if (karta.cervena[i, 3] == "zelenka")
                            karta.cervena[i, 3] = "1";
                    }
                    MessageBox.Show("Hráč " + vysledek + " byl vyřazen ze hry.",
"Hráč vyřazen", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }
                if ((vysledek == "Modrák") & (modrak.stav != "vyrazen"))
                {
                    blue.Location = new Point(zacatek.f_bx, zacatek.f_y);
                    blue_hp.Location = new Point(zacatek.f_bx, zacatek.f_y);
                    modrak.stav = "vyrazen";
                    modrak.misto = "N/A";
                    modrak.loviste = "N/A";
                    for (int i = 10; i < 16; i++)
                    {
                        if (karta.modra[i, 3] == "modrak")
                            karta.modra[i, 3] = "1";
                        if (karta.cervena[i, 3] == "modrak")
                            karta.cervena[i, 3] = "1";
                    }
                    MessageBox.Show("Hráč " + vysledek + " byl vyřazen ze hry.",
"Hráč vyřazen", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }
                if ((vysledek == "Žluťka") & (zlutka.stav != "vyrazen"))
                {
                    yellow.Location = new Point(zacatek.f_yx, zacatek.f_y);
                    yellow_hp.Location = new Point(zacatek.f_yx, zacatek.f_y);
                    zlutka.stav = "vyrazen";
                    zlutka.misto = "N/A";
                    zlutka.loviste = "N/A";
                    for (int i = 10; i < 16; i++)
                    {
                        if (karta.modra[i, 3] == "zlutka")
                            karta.modra[i, 3] = "1";
                        if (karta.cervena[i, 3] == "zlutka")
                            karta.cervena[i, 3] = "1";
                    }
                    MessageBox.Show("Hráč " + vysledek + " byl vyřazen ze hry.",
"Hráč vyřazen", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }
            }
```

Na konci fáze je již jen zápis změn stavů (odhalení, vyřazení hráče), případné použití schopností, aktualizace paměti AI a změna fáze kola.

```
            //stav hracu
            cervenka.stav = stav[0];
            zelenka.stav = stav[1];
            modrak.stav = stav[2];
            zlutka.stav = stav[3];

            //stav schopnosti
            cervenka.schop = schopnosti[0];
            zelenka.schop = schopnosti[1];
```

```
                modrak.schop = schopnosti[2];
                zlutka.schop = schopnosti[3];

                //RGBY
                cervenka.pametG = pametR[1];
                cervenka.pametB = pametR[2];
                cervenka.pametY = pametR[3];

                zelenka.pametR = pametG[0];
                zelenka.pametB = pametG[2];
                zelenka.pametY = pametG[3];

                zlutka.pametR = pametY[0];
                zlutka.pametG = pametY[1];
                zlutka.pametB = pametY[2];

                fazovac.Text = "Místo";

                        return;
            }
```

4.8.4.2 Místo

V této fázi hry se řeší, kde hráč je a jaký efekt místa bude aplikován. Tedy zda bude brát kartu, krást vybavení jiným hráčům, či bude léčit nebo způsobovat zranění. Taktéž se zde nulují proměnné, které se v této fázi používají.

```
if (fazovac.Text == "Místo")
            {
               Mista mista = new Mista(aktivni, cervenka.misto, zelenka.misto,
modrak.misto, zlutka.misto, cervenka.rasa, zelenka.rasa, modrak.rasa, zlutka.rasa,
cervenka.zraneni, zelenka.zraneni, modrak.zraneni, zlutka.zraneni,
Convert.ToInt32(cervenka.zivoty), Convert.ToInt32(zelenka.zivoty),
Convert.ToInt32(modrak.zivoty), Convert.ToInt32(zlutka.zivoty), pocet_tahu);
                jmenoK = "";
                typK = "";
                textK = "";
                z = 0;
                y = 0;
                x = 0;
```

Velká část toho, co se na různých místech děje, se řeší v třídě Mista.

```
class Mista
    {
        public int i = 0;
        public string hrac = "";
        public string mistoA = "";

        public string rozhodnuti = "";
        public string jmeno = "";
        public string text = "";

        public int zranA = 0;
        public int hpA = 0;

        //rasu nepotrebuji, leda bych resil zda si AI ma pamatovat zda leceni pro
jeji rasu jiz probehlo ci ne
```

```
        public Mista(string aktivni, string mistoR, string mistoG, string mistoB,
string mistoY,
           string rasaR, string rasaG, string rasaB, string rasaY,
            int zranR, int zranG, int zranB, int zranY, int zivR, int zivG, int
zivB, int zivY, int pocet_tahu)
        {
            string[] hraci = { "cervenka", "zelenka", "modrak", "zlutka" };
            string[] kdeje = { mistoR, mistoG, mistoB, mistoY };
            int[] zraneni = { zranR, zranG, zranB, zranY };
            int[] zivoty = { zivR, zivG, zivB, zivY };

            //vybrani aktivniho hrace a mista kde je
            while (mistoA == "")
            {

                if (hraci[i] == aktivni)
                    mistoA = kdeje[i];
                else
                    i++;
            }
            //zraneni aktivniho hrace
            for (int j = 0; j < 4; j++)
            {
                if (hraci[i] == aktivni)
                    zranA = zraneni[i];
                else
                    i++;
            }
            //hp aktivniho hrace
            for (int j = 0; j < 4; j++)
            {
                if (hraci[i] == aktivni)
                    hpA = zivoty[i];
                else
                    i++;
            }
```

Nejdůležitější zde je část, kdy se AI rozhoduje, co bude dělat na území Temný les a Prameny moudrosti.

```
if (mistoA == "les")
            {
                rozhodnuti = "";

                // pro modraka novy form ktery bude vracet leceni ci zraneni. Nutno
poresit aby hrac nezranil sam sebe? cca jak pri boji
                if (aktivni == "modrak")
                {
                    les l = new les();

                    //showdialog zobrazi vyber a pocka do jeho ukonceni
                    l.ShowDialog();
                    rozhodnuti = l.hodnotaB;
                }
                else if (aktivni != "modrak")
                {
                    if (hpA - zranA < 3)
                        rozhodnuti = "leceniS";
                    else
                        rozhodnuti = "zranenS";
```

```
            }
        }
```

Třída les umožňuje uživateli vybrat si, zda chce léčit či zraňovat pomocí efektu místa Temný les. AI se rozhoduje podle svého aktuálního zranění. Rozhodování je přizpůsobeno dle reálně odehraných partií, kdy se ukázalo, že pokud není hráč těsně před vyřazením ze hry, vyplatí se více zranit svého protihráče, než se léčit.

Samotná třída les je velmi prostá.

```
private void ok_Click(object sender, EventArgs e)
        {
            if (t1.Checked) //zranen
            {
                hodnotaB = "zranenS";
            }

            if (t2.Checked) //leceni
            {
                hodnotaB = "leceniS";
            }
```

Dále se ve třídě Mista řeší místo Prameny moudrosti. Uživateli je opět pro rozhodnutí vypsán windows form Prameny a AI se řídí dle toho, kolikáté je kolo a kolik zranění již má. Dvanáct tahů hry se rovná třem celým kolům. Tři kola by mělo být dostatkem času k tomu, aby AI nasbíralo informace o ostatních hráčích.

```
if (mistoA == "prameny")
            {
                if (aktivni == "modrak")
                {
                    rozhodnuti = "";

                    prameny p = new prameny();

                    //showdialog zobrazi vyber a pocka do jeho ukonceni
                    p.ShowDialog();
                    rozhodnuti = p.hodnotaB;
                }
                else if (aktivni != "modrak")
                {
                    if (pocet_tahu < 12)
                    {
                        rozhodnuti = "zelena";
                    }
                    else
                        rozhodnuti = "cervena";

                    //v pripade velkeho zraneni voli modrou pro mozne leceni
                    if (hpA - zranA < 5)
                        rozhodnuti = "modra";
                }
            }
```

Samotná třída Prameny je opět velmi jednoduchá.

```
        public string hodnotaB;
        private void ok_Click(object sender, EventArgs e)
```

```
        {
            if (t1.Checked) //cervena
            {
                hodnotaB = "cervena";
            }

            if (t2.Checked) //zelena
            {
                hodnotaB = "zelena";
            }

            if (t3.Checked) //modra
            {
                hodnotaB = "modra";
            }
        }
```

Zbytek třídy Mista je taktéž již velmi prostý, a proto ho zde není vypsán celý. Pro příklad stačí řešení místa Kamenný kruh.

```
if (mistoA == "kruh")
       rozhodnuti = "zlodej";
```

Zpět v hlavním kódu se řeší, co se tedy na jakém místě děje, respektive zda aktivní hráč táhne nějakou kartu. Červené a Modré karty jsou řešeny obdobně, a proto jsou zde uvedeny jen Modré, neboť mají navíc jednu jednorázovou kartu, která se ale po dobu jednoho kola chová jako vybavení.

Karta je náhodně vybrána z příslušného pole a její hodnoty jsou zaneseny do připravených proměnných. Hledá se karta, které ještě nebyla hraná. Pokud již hraná byla a ani na šestý pokus se nepovede kartu vylosovat, změní se stav odhozených karet zpět na 0, tedy na použitelné. Následně se uživateli vypíše, co za kartu bylo zahráno. Pokud je vytažená karta vybavení, její stav se změní na jméno aktivního hráče, pokud je jednorázová, je odhozena (její stav se změní na 1).

Pokud je vytažena karta Anděl strážný, aplikace s ní zachází jako s předmětem.

```
                //modre karty
                if (mista.rozhodnuti == "modra")
                {
                    do
                    {
                        kostka = new Kostka(16);
                        x = kostka.UK - 1;
                        if (karta.modra[x, 3] == "0")
                        {
                            jmenoK = karta.modra[x, 0];
                            typK = karta.modra[x, 1];
                            textK = karta.modra[x, 2];
                            z = 1;
                        }
                        else
                            y++;
```

```
                    //osetreni navraceni karet co jsou odhozene zpet do balicku
                    if (y > 6)
                    {
                        y = 0;
                        for (y = 0; y < 16; x++)
                        {
                            if (karta.modra[y, 3] == "1")
                                karta.modra[y, 3] = "0";
                        }
                    }

                }
                while (z == 0);

                if (typK == "Jednorázové použití")
                    karta.modra[x, 3] = "1";
                else
                    karta.modra[x, 3] = aktivni;

                //objeveni andela
                if (jmenoK == "Anděl strážný")
                    karta.modra[x, 3] = aktivni;

                MessageBox.Show(textK, jmenoK, MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
```

Zelené karty fungují obdobně, rozdílem je, že proměnných je více a v této fázi kola se text karty vypíše uživateli, pouze pokud je aktivním hráčem.

```
            //zelena karta
            //jmeno, text, typ, rasa1, rasa2, moznost1, moznost2, stav
            if (mista.rozhodnuti == "zelena")
            {
                while (z == 0)
                {
                    kostka = new Kostka(16);
                    x = kostka.UK - 1;

                    if (karta.zelena[x, 7] == "0")
                    {
                        jmenoK = karta.zelena[x, 0];
                        textK = karta.zelena[x, 1];
                        typK = karta.zelena[x, 2];
                        rasa1 = karta.zelena[x, 3];
                        rasa2 = karta.zelena[x, 4];
                        moznost1 = karta.zelena[x, 5];
                        moznost2 = karta.zelena[x, 6];

                        z = 1;
                    }
                    else
                        y++;

                    //osetreni navraceni karet co jsou odhozene zpet do balicku
                    if (y > 6)
                    {
                        for (x = 0; x < 16; x++)
                        {
                            if (karta.zelena[x, 7] == "1")
```

```
                                karta.zelena[x, 7] = "0";
                            }
                        }
                    }
                    karta.zelena[x, 7] = "1";

                    if(aktivni == "modrak")
                        MessageBox.Show(textK, jmenoK, MessageBoxButtons.OK,
MessageBoxIcon.Information);
  }
```

Poté se již jen vyřeší, co za akci nastalo, a změní se fáze hry na událost. Též proběhne aktualizace výpisu stavu u Modráka, neboť uživatel se může odhalit kdykoliv. Pro AI nemá smysl, aby se v této fázi hry odhalovalo.

```
if ((mista.rozhodnuti != "Jednorázové použití") & (mista.rozhodnuti != "VYBAVENÍ"))
                    {
                        akce = mista.rozhodnuti;
                    }


                    fazovac.Text = "Událost";
                    //nahozeni kostek do puvodni pozice kvuli osetreni chyby
                    sestka.Text = "0";
                    ctyrka.Text = "0";

                    stavB.Text = modrak.stav;

                    return;
```

4.8.4.3 Událost

V této fázi kola se řeší efekty karet a míst z fáze Místo. První, co se zde řeší, jak pro uživatele tak pro AI, je, zda zvolený cíl není vyřazen. To se děje ve třídě Boj.

```
if (fazovac.Text == "Událost")
            {
                mistoHP = 0;
                figurkaHP = 0;
                cil = "";

                if (aktivni == "modrak")
                {
                    if (hr1.Checked)
                        cil = "cervenka";
                    if (hr2.Checked)
                        cil = "zelenka";
                    if (hr3.Checked)
                        cil = "modrak";
                    if (hr4.Checked)
                        cil = "zlutka";

                    overStav = new Boj(cil, cervenka.stav, zelenka.stav,
modrak.stav, zlutka.stav, x);
                    if (overStav.stavA == "vyrazen")
                    {
                        MessageBox.Show("Tebou vybraný cíl, je již vyřazen, zvol
jiný cíl.", "Chyba", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
                        return;
                    }
                }
                else if (aktivni != "modrak")
                {
                    //overeni zda je cil vyrazen
                    overStav = new Boj(cil, cervenka.stav, zelenka.stav,
modrak.stav, zlutka.stav, x);


                    if ((overStav.stavA == "vyrazen") & (cil == cil1))
                        cil = cil2;
                    else if ((overStav.stavA == "vyrazen") & (cil == cil2))
                        cil = cil1;

                }

                //vynulovani hodnot
                mistoHP = 0;
                figurkaHP = 0;
                odkladiste = "";
```

Následně jsou postupně řešeny jednotlivé efekty. Prvním z jich je Temný les. Zranění z tohoto efektu je redukováno jedním z předmětů a to Talismanem. Pokud ho cíl efektu má, je uživatel upozorněn, aby si zvolil jiný cíl, AI ověří stav druhého cíle, zda nebyl vyřazen a nechá efekt dopadnout na něj. Léčení jen umožňuje uživateli zvolit za cíl efektu kohokoliv z hráčů, AI léčí zásadně jen sebe.

```
#region Srom
                if (akce == "zranenS")
                {
                    mistoHP = 2;
                    //reseni talismanu
                    if (karta.modra[12, 3] == cil)
                    {
                        if (aktivni == "modrak")
                        {
                            MessageBox.Show("Tebou vybraný cíl má ve svém
vlastnictví Brož štěstěny. Vyber si jiný cíl.", "Upozornění", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            return;
                        }
                        else
                        {
                            if (cil == cil1)
                            {
                                cil = cil2;
                                Boj overStav = new Boj(cil, cervenka.stav,
zelenka.stav, modrak.stav, zlutka.stav, x);
                                if ((overStav.stavA == "vyrazen") & (cil == cil2))
                                    cil = null;
                            }
                            else if (cil == cil2)
                                cil = cil1;
                        }
                    }
                }
```

```
                if (akce == "leceniS")
                {
                    mistoHP = -1;
                    if (aktivni != "modrak")
                        cil = aktivni;
                }
```

Další v pořadí je řešení místa Kamenný kruh potažmo červené karty Mrzutý goblin, protože efekt obou je totožný. Prvně se ověří, zdali některý z hráčů má některé vybavení ve svém držení. Pokud ano, je uživateli zobrazena možnost vybrání si z vybavení k ukradení pomocí windows formu Zlodej.

AI preferuje prvně výběr z Červených karet, pokud žádnou nenalezne, krade kartu Modrou. Dále je u specifických karet ovlivňující sílu útoku hráčů nutno u jejich předchozího vlastníka upravit jeho útočné číslo o hodnotu sebraného vybavení. Posledním krokem zloděje je načtení jména ukradené karty k dalšímu zpracování v rámci této fáze.

```
                #region Kamenny kruh
                if ((akce == "zlodej") | (jmenoK == "Mrzutý goblin"))
                {
                    for (int i = 0; i < 4;i++)
                    {
                        for(int a = 10; a < 16; a++)
                        {
                            if (((karta.modra[a, 3] == hraci[i]) | (karta.cervena[a,
3] == hraci[i]))& hraci[i] != aktivni)
                                odkladiste = "ok";
                        }


                    }

                    if ((aktivni == "modrak") & (odkladiste == "ok"))
                    {
                        Zlodej kradez = new Zlodej(aktivni, akce,
                            karta.modra[10, 3], karta.modra[11, 3], karta.modra[12,
3], karta.modra[13, 3], karta.modra[14, 3], karta.modra[15, 3],
                            karta.cervena[10, 3], karta.cervena[11, 3],
karta.cervena[12, 3], karta.cervena[13, 3], karta.cervena[14, 3], karta.cervena[15,
3]);

                        kradez.ShowDialog();

                        if (kradez.barva == "modra")
                        {
                           jmenoK = karta.modra[kradez.vybrane_vybaveni, 0];

                            //reseni kopi
                            if(kradez.vybrane_vybaveni == 10)
                            {
                                y = 0;

                                while (karta.modra[kradez.vybrane_vybaveni,3] !=
hraci[y])
                                {
                                    y++;
```

```
                                }
                                if(kopi == 1)
                                {
                                    utc[y] = utc[y] - 2;
                                    kopi = 0;
                                }

                            }
                            //reseni roucha
                            if (kradez.vybrane_vybaveni == 14)
                            {
                                y = 0;

                                while (karta.modra[kradez.vybrane_vybaveni, 3] !=
hraci[y])
                                {
                                    y++;
                                }
                                utc[y] = utc[y] + 1;

                            }

                            karta.modra[kradez.vybrane_vybaveni, 3] = aktivni;
                        }

                        if (kradez.barva == "cervena")
                        {
                            jmenoK = karta.cervena[kradez.vybrane_vybaveni, 0];

                            //reseni cervenych ut predmetu
                            if ((kradez.vybrane_vybaveni == 11)|
(kradez.vybrane_vybaveni == 12) | (kradez.vybrane_vybaveni == 13))
                            {
                                y = 0;

                                while (karta.cervena[kradez.vybrane_vybaveni, 3] !=
hraci[y])
                                {
                                    y++;
                                }
                                utc[y] = utc[y] - 1;

                            }
                            karta.cervena[kradez.vybrane_vybaveni, 3] = aktivni;
                        }
                    }
                    else if((aktivni != "modrak") & (odkladiste == "ok"))
                    {
                        z = 0;

                        for (int i = 10; i < 16; i++)
                        {
                            if (z == 0)
                            {
                                if ((karta.cervena[i, 3] != aktivni) &
(karta.cervena[i, 3] != "0"))
                                    z = i;
                            }
                        }
```

```
                        if ((z != 0))
                        {
                            if (z != 0)
                            {
                                //reseni cervenych ut predmetu
                                if ((z == 11) | (z == 12) | (z == 13))
                                {
                                    y = 0;

                                    while (karta.cervena[z, 3] != hraci[y])
                                    {
                                        y++;
                                    }
                                    utc[y] = utc[y] - 1;
                                }
                                karta.cervena[z, 3] = aktivni;
                                jmenoK = karta.cervena[z, 0];
                            }
                        }
                        else if ((z == 0))
                        {
                            for (int i = 10; i < 16; i++)
                            {
                                if (z == 0)
                                {
                                    if ((karta.modra[i, 3] != aktivni)
&(karta.modra[i, 3] != "0"))
                                        z = i;
                                }
                            }

                            //reseni kopi
                            if (z == 10)
                            {
                                y = 0;

                                while (karta.modra[z, 3] != hraci[y])
                                {
                                    y++;
                                }
                                if (kopi == 1)
                                {
                                    utc[y] = utc[y] - 2;
                                    kopi = 0;
                                }
                            }
                            //reseni roucha
                            if (z == 14)
                            {
                                y = 0;

                                while (karta.modra[z, 3] != hraci[y])
                                {
                                    y++;
                                }
                                utc[y] = utc[y] + 1;
                            }
                            if(z > 0)
                            {
```

```
                                karta.modra[z, 3] = aktivni;
                                jmenoK = karta.modra[z, 0];
                            }
                        }
                    }
                }
                #endregion
```

Samotná třída Zlodej načte stavy karet Vybavení a dle toho je vypíše. Pokud uživatel zkusí ukrát předmět, který ještě není ve vlastnictví žádného hráče, je na to upozorněn. Třída Zlodej rozlišuje zdroj kradení (viz Zelené karty). Jelikož je kód totožný pro všechny karty, je zde uvedena jen jeho část.

```
public Zlodej(string aktivni, string stav,
            string kopi, string ruzenec, string broz, string talisman, string
roucho, string kompas,
            string kulomet, string sekacek, string pila, string sekera, string
pochoden, string pistol)
        {
            InitializeComponent();

            t1.Text = null;
            t2.Text = null;
         if (stav == "zelena")
            {
                if (kopi == aktivni)
                    t1.Text = "Longinovo kopí";
                if (ruzenec == aktivni)
                    t2.Text = "Stříbrný růženec";
              }
        if (stav == "zlodej")
            {
                if ((kopi != aktivni) & (kopi != "0"))
                    t1.Text = "Longinovo kopí";
                if ((ruzenec != aktivni) & (ruzenec != "0"))
                    t2.Text = "Stříbrný růženec";
              }
public int vybrane_vybaveni = 35505;
        public string barva;
        private void ok_Click(object sender, EventArgs e)
        {
            #region Modre
            if (t1.Checked) //kopi
            {
                if (t1.Text != null)
                {
                    vybrane_vybaveni = 10;
                    barva = "modra";
                }
                else
                {
                    MessageBox.Show("Vybral jsi předmět, který nemá žádného
vlastníka, tedy nejde ukrást či darovat.","Chyba",MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    return;
                }
            }
```

Následně se řeší Zelené karty, které rozlišujeme na čtyři různé typy. Díky těmto kartám získávají hráči informace o ostatních hráčích. AI zde používá jiný typ vyhledávání cíle než během boje. Proto zde musí být znovu ověření stavu cíle.

```
#region ZELENE
                if (((typK == "A") | (typK == "B") | (typK == "C") | (typK == "D"))
& (aktivni != "modrak"))
                {
                    odkladiste = "";

                    switch (aktivni)
                    {
                        case "cervenka":
                            pamet = new Pamet(aktivni, cervenka.pametR,
cervenka.pametG, cervenka.pametB, cervenka.pametY);
                            break;
                        case "zelenka":
                            pamet = new Pamet(aktivni, zelenka.pametR,
zelenka.pametG, zelenka.pametB, zelenka.pametY);
                            break;
                        case "zlutka":
                            pamet = new Pamet(aktivni, zlutka.pametR, zlutka.pametG,
zlutka.pametB, zlutka.pametY);
                            break;
                    }
```

Ve třídě Pamet se určí aktivní hráč a podle toho se načte jeho rasa do příslušné proměnné. Jelikož je předem znám počet zástupců jednotlivých ras, stačí, když AI odhalí dva hráče stejné rasy, aby bylo známo vše potřebné pro další hru. Cíl se vybírá náhodně a následně je u něj ověřeno, že AI o vybraném cíli nic neví. Pokud náhodný výběr neuspěje, algoritmus vybere první cíl, o kterém nic neví.

Náhodný výběr byl přidán z důvodu, že všechna AI za první cíl volila hráče Červenka a hráč Červenka volil za svůj cíl vždy hráče Zelenku.

```
public Pamet(string aktivni, string pametR, string pametG, string pametB, string
pametY)
        {
            string[] hraci = { "cervenka", "zelenka", "modrak", "zlutka" };
            string[] pamet = { pametR, pametG, pametB, pametY };
            cil = "";
            i = 0;
            rasaA = null;
            //pocitadlo
            vlk = 0;
            upir = 0;

            while (rasaA == null)
            {
                if (hraci[i] == aktivni)
                    rasaA = pamet[i];
                else
                    i++;
            }
```

```
            for (int p = 0; p < 4; p++)
            //while(cil == "")
            {
                if (pamet[p] == "vlkodlak")
                    vlk++;
                if (pamet[p] == "upír")
                    upir++;
            }

            if ((upir < 2) | (vlk < 2))
            {
                kostka = new Kostka(4);
                x = kostka.UK - 1;

                for (int j = 0; j < 4; j++)
                {
                    if (rasaA != pamet[x])
                    {
                        if ((cil == "") & (pamet[x] == ""))
                            cil = hraci[x];
                    }
                    else
                    {
                        kostka = new Kostka(4);
                        x = kostka.UK - 1;
                    }

                }

                if (cil == "")
                {
                    x = 0;
                    for (int j = 0; j < 4; j++)
                    {
                        if (rasaA != pamet[x])
                        {
                            if ((cil == "") & (pamet[x] == ""))
                                cil = hraci[x];
                        }
                        else
                            x++;
                    }
                }
            }
            else
                cil = "";
        }
```

Pokud v rámci třídy Pamet není nalezen žádný cíl, či vybraný cíl je již vyřazen, vyhledá se cíl dle pravidel pro boj. Pokud je cílem uživatel, je mu vypsáno, o jakou kartu se jedná.

```
                    cil = pamet.cil;
                    overStav = new Boj(cil, cervenka.stav, zelenka.stav,
modrak.stav, zlutka.stav, x);
                    if (overStav.stavA == "vyrazen")
                        cil = "";
if (aktivni != "modrak")
                    {
```

```
                        if (cil == "")
                        {
                            cilZran = new Boj(cil1, cil2, cervenka.zraneni,
cervenka.zraneni, modrak.zraneni, zlutka.zraneni);
                            cil = cilZran.cil;

                            overStav = new Boj(cil, cervenka.stav, zelenka.stav,
modrak.stav, zlutka.stav, x);


                            if ((overStav.stavA == "vyrazen") & (cil == cil1))
                                cil = cil2;
                            else if ((overStav.stavA == "vyrazen") & (cil == cil2))
                                cil = cil1;
                        }

                        if (cil == "modrak")
                            MessageBox.Show(textK, jmenoK, MessageBoxButtons.OK,
MessageBoxIcon.Information);
                    }
```

Typ A obsahuje dvě rasy, a pokud cíl je jednou z nich, stane se jeden z efektů karty. Cíl musí odevzdat jednu svou kartu vybavení aktivnímu hráči, a pokud žádné vybavení nemá, je zraněn. Opět musí proběhnout řešení ukradení specifických předmětů. Uživateli je umožněno přes třídu Zloděj vybrat předmět který, dá aktivnímu hráči, pokud splňuje podmínku a AI dává primárně Modrou kartu, pokud ji má. Pokud ji nemá, tak volí vybavení z karet Červených. Na konec se řeší zápis do paměti AI s tím, že pokud jedna z ras v podmínce byl člověk, AI ví, že cílový hráč musel být rasou druhou. Pokud byly na kartě obe dvě herní rasy, AI neví nic.

```
if (typK == "A")
                {
                    //jmeno, text, typ, rasa1, rasa2, moznost1, moznost2, stav
                    if ((rasa[y] == rasa1) | (rasa[y] == rasa2))
                    {
                        z = 0;

                        for (int i = 10; i < 16; i++)
                        {
                            if (z == 0)
                            {
                                if (karta.modra[i, 3] == cil)
                                    z = i;
                            }
                        }

                        if ((z != 0) & (cil != "modrak"))
                        {


                            //reseni kopi
                            if (z == 10)
                            {
                                y = 0;
```

```
                        while (karta.modra[z, 3] != hraci[y])
                        {
                            y++;
                        }
                        if (kopi == 1)
                        {
                            utc[y] = utc[y] - 2;
                            kopi = 0;
                        }
                    }
                    //reseni roucha
                    if (z == 14)
                    {
                        y = 0;

                        while (karta.modra[z, 3] != hraci[y])
                        {
                            y++;
                        }
                        utc[y] = utc[y] + 1;
                    }
                    karta.modra[z, 3] = aktivni;
                    jmenoK = karta.modra[z, 0];
                }
                else if ((z == 0) & (cil != "modrak"))
                {
                    for (int i = 10; i < 16; i++)
                    {
                        if (z == 0)
                        {
                            if (karta.cervena[i, 3] == cil)
                                z = i;
                        }
                    }
                    if (z != 0)
                    {
                        //reseni cervenych ut predmetu
                        if ((z == 11) | (z == 12) | (z == 13))
                        {
                            y = 0;

                            while (karta.cervena[z, 3] != hraci[y])
                            {
                                y++;
                            }
                            utc[y] = utc[y] - 1;


                        }
                        karta.cervena[z, 3] = aktivni;
                        jmenoK = karta.cervena[z, 0];
                    }
                }
                if(cil == "modrak")
                {
                    for (int i = 10; i < 16; i++)
                    {
                        if (z == 0)
                        {
```

```
                                    if ((karta.modra[i, 3] == cil) &
(karta.cervena[i, 3] == cil))
                                        z = i;
                                }

                            }
                        }

                        if (z != 0)
                        {
                            Zlodej kradez = new Zlodej(aktivni, "zelena",
                    karta.modra[10, 3], karta.modra[11, 3], karta.modra[12, 3],
karta.modra[13, 3], karta.modra[14, 3], karta.modra[15, 3],
                    karta.cervena[10, 3], karta.cervena[11, 3], karta.cervena[12, 3],
karta.cervena[13, 3], karta.cervena[14, 3], karta.cervena[15, 3]);

                            kradez.ShowDialog();

                            if (kradez.barva == "modra")
                            {
                                //reseni kopi
                                if (kradez.vybrane_vybaveni == 10)
                                {
                                    y = 0;

                                    while (karta.modra[kradez.vybrane_vybaveni, 3]
!= hraci[y])
                                    {
                                        y++;
                                    }
                                    if (kopi == 1)
                                    {
                                        utc[y] = utc[y] - 2;
                                        kopi = 0;
                                    }
                                }
                                //reseni roucha
                                if (kradez.vybrane_vybaveni == 14)
                                {
                                    y = 0;

                                    while (karta.modra[kradez.vybrane_vybaveni, 3]
!= hraci[y])
                                    {
                                        y++;
                                    }
                                    utc[y] = utc[y] + 1;


                                }

                                karta.modra[kradez.vybrane_vybaveni, 3] = aktivni;
                                jmenoK = karta.modra[kradez.vybrane_vybaveni, 0];
                            }

                            if (kradez.barva == "cervena")
                            {
                                jmenoK = karta.cervena[kradez.vybrane_vybaveni, 0];
```

```
                                //reseni cervenych ut predmetu
                                if ((kradez.vybrane_vybaveni == 11) |
(kradez.vybrane_vybaveni == 12) | (kradez.vybrane_vybaveni == 13))
                                {
                                    y = 0;

                                    while (karta.cervena[kradez.vybrane_vybaveni, 3]
!= hraci[y])
                                    {
                                        y++;
                                    }
                                    utc[y] = utc[y] - 1;

                                }
                                karta.cervena[kradez.vybrane_vybaveni, 3] = aktivni;
                                jmenoK = karta.cervena[kradez.vybrane_vybaveni, 0];
                            }
                        }

                        if (z == 0)
                        {
                            mistoHP = Convert.ToInt32(moznost2);
                            z = 1;
                        }

                        if (z != 0)
                        {

                            //reseni vysledku pro aktivniho
                            if (rasa1 == rasa[y])
                                rasaC = rasa1;
                            if (rasa2 == rasa[y])
                                rasaC = rasa2;

                            if ((rasa1 == "vlkodlak") & (rasa2 == "upír"))
                                rasaC = "";
                            if (rasa1 == "člověk")
                                rasaC = rasa2;
                            if (rasa2 == "člověk")
                                rasaC = rasa1;
                        }
                    }
```

Druhým typem jsou karty mající v podmínce jen jednu rasu, a jestliže je podmínka splněna, způsobí zranění. Do tohoto typu spadají i karty ptající se na začínající písmeno jména postavy.

```
            if (typK == "B")
              {
                  if (rasa[y] == rasa1)
                  {
                      mistoHP = Convert.ToInt32(moznost1);

                      rasaC = rasa1;
                      //reseni vysledku pro aktivniho
                      if ((rasa1 == "vlkodlak") & (rasa2 == "upír"))
                          rasaC = "";
                      if (rasa1 == "člověk")
                          rasaC = rasa2;
```

```
                        if (rasa2 == "člověk")
                            rasaC = rasa1;
                    }

                    if (rasa1 == "jmena1")
                    {

                        if (jmeno[y].StartsWith("A") | jmeno[y].StartsWith("B") |
jmeno[y].StartsWith("C")
                            | jmeno[y].StartsWith("E") | jmeno[y].StartsWith("U"))
                        {
                            mistoHP = Convert.ToInt32(moznost1);
                        }

                    }
                    if (rasa1 == "jmena2")
                    {

                        if (jmeno[y].StartsWith("D") | jmeno[y].StartsWith("F") |
jmeno[y].StartsWith("G")
                            | jmeno[y].StartsWith("V") | jmeno[y].StartsWith("W"))
                        {
                            mistoHP = Convert.ToInt32(moznost1);
                        }
                    }

                    //zapis do pameti
                    switch (aktivni)
                    {
                        case "cervenka":
                            pametR[y] = rasaC;
                            break;

                        case "zelenka":
                            pametG[y] = rasaC;
                            break;

                        case "zlutka":
                            pametY[y] = rasaC;
                            break;
                    }
```

Typ C se opět týká pouze jedné rasy, ale má dvě možnosti toho, co se stane. Pokud cíl splňuje danou rasu a nemá žádné zranění, je mu zranění uděleno, ale pokud již zranění dostal, je naopak vyléčen.

```
                if (typK == "C")
                {
                    //jmeno, text, typ, rasa1, moznost1, moznost2, stav

                    if (rasa[y] == rasa1)
                    {
                        if (zran[y] == 0)
                            mistoHP = Convert.ToInt32(moznost1);
                        else
                            mistoHP = Convert.ToInt32(moznost2);

                        rasaC = rasa1;
```

```
                        //reseni vysledku pro aktivniho
                        if ((rasa1 == "vlkodlak") & (rasa2 == "upír"))
                            rasaC = "";
                        if (rasa1 == "člověk")
                            rasaC = rasa2;
                        if (rasa2 == "člověk")
                            rasaC = rasa1;
                    }

                    //zapis do pameti
                    switch (aktivni)
                    {
                        case "cervenka":
                            pametR[y] = rasaC;
                            break;

                        case "zelenka":
                            pametG[y] = rasaC;
                            break;

                        case "zlutka":
                            pametY[y] = rasaC;
                            break;
                    }
                }
```

A posledním typem je pouhopouhá jedna karta, která odhaluje pro aktivního hráče kartu postavy jeho cíle. Pro AI to znamená, že si načte do paměti přímo rasu daného hráče. V případě uživatele se karta postavy vypíše jako hláška.

```
if (typK == "D")
                {
                    y = 0;
                    string cilR = "";
                    //{"Poustevníkovo proroctví", "Musíš tajně ukázat svou kartu
Postavy aktuálnímu hráči!", "0" },
                    switch (aktivni)
                    {
                        case "cervenka":
                            pametR[y] = rasa[y];
                            break;

                        case "zelenka":
                            pametG[y] = rasa[y];
                            break;

                        case "modrak":
                            if (rasa[y] == "upír")
                                cilR = "Všichni VLKODLACI jsou vyřazeni!";
                            if (rasa[y] == "vlkodlak")
                                cilR = "Všichni UPÍŘI jsou vyřazeni!";

                            MessageBox.Show("Jméno: " + jmeno[y] + "\n Rasa: " +
rasa[y] + "\n Životy: " + hp[y] + "\n Cíl: " + cilR + "\n Schopnost: " +
schopnostiT[y], jmenoK, MessageBoxButtons.OK, MessageBoxIcon.Information);
                            break;

                        case "zlutka":
                            pametY[y] = rasa[y];
```

```
                        break;
                    }
                }
```

Následně se v hlavním kódu řeší efekty jednotlivých karet. První v pořadí jsou karty Červené. První kartou je Pekelný obřad, který je speciálním léčením pro rasu upírů. Pro AI to znamená, že pokud hraje za tuto rasu a zranění její postavy je větší nebo rovno jejím životům, provede odhalení postavy. Samotná karta je pak řešena velmi jednoduše. Pokud je hráč upír a je odhalen, je jeho zranění rovno nule. Změna stavu se následně zapíše k příslušnému hráči.

```
if (jmenoK == "Pekelný obřad")
                {
                    x = 0;

                    while (aktivni != hraci[x])
                    {
                        x++;
                    }

                    stavA = stav[x];
                    hpA = Convert.ToInt32(hp[x]);
                    zranA = zran[x];
                    rasaA = rasa[x];

                    //overeni rasy
                    if (rasaA == "upír")
                    {
                        //AI
                        if (aktivni != "modrak")
                        {
                            //Odhali se pokud jeho zraneni presahuji ci rovnaji se
polovine jeho zivotu
                            if ((zranA >= (hpA / 2)) & (stavA != "odhalen"))
                            {
                                stavA = "odhalen";
                                stav[x] = stavA;
                            }
                        }

                        //Pokud je hrac odhalen a splnuje podminku je vylecen
                        if (stavA == "odhalen")
                        {
                            mistoHP = -zranA;
                            cil = aktivni;
                        }
                    }

                    cervenka.stav = stav[0];
                    zelenka.stav = stav[1];
                    modrak.stav = stav[2];
                    zlutka.stav = stav[3];
                }
```

Červená karta Banánová slupka přikazuje hráči, aby daroval jednu kartu vybavení jinému hráči. V ideálním případě AI za cíl volí hráče, o kterém ví, že by neměl být jejím protivníkem. Zbytek kódu je pro AI i uživatele totožný s kódem u Zelených karet typu A.

```
else if ((aktivni != "modrak") & (odkladiste == "ok"))
                    {
                        cil = "";
                        y = 0;
                        while (cil != "")
                        {
                            if ((cil1 != hraci[y]) & (cil2 != hraci[y]))
                                cil = hraci[y];
                            else if (y > 4)
                            {
                                y = 0;
                                for (int i = 0; i < 4; i++)
                                {

                                    if (aktivni != hraci[y])
                                        cil = hraci[y];
                                    else
                                        y++;
                                }

                            }
                            else
                                y++;
                        }
```

Karty Netopýr a Krvežíznivý pavouk jsou řešeny obdobně. Do proměnných se načtou hodnoty zranění cíle a aktivního hráče, ověří se, zda cíl nemá předmět Talisman, který by ho před efektem chránil, a pokud ho má, je uživatel vyzván zvolit jiný cíl, zatímco AI si zvolí nový cíl samo.

```
            if (jmenoK == "Netopýr")
               {
                    mistoHP = 2;
                    figurkaHP = -1;

                    //reseni talismanu
                    if (karta.modra[13, 3] == cil)
                    {
                        if (aktivni == "modrak")
                        {
                            MessageBox.Show("Tebou vybraný cíl má ve svém
vlastnictví Talisman. Vyber si jiný cíl.", "Upozornění", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            return;
                        }
                        else
                        {
                            if (cil == cil1)
                                cil = cil2;
                            else if (cil == cil2)
                                cil = cil1;
                        }
```

```
                }
            }
```

Poslední červenou jednorázovou kartou v balíčku je Dynamit. Hodí se kostkami na určení místa a následně jsou zraněny všechny postavy, které na daném místě stojí. Pro následný pohyb figurek po počítadle zranění je změněn cíl.

```
if (jmenoK == "Dynamit")
                {
                    kostka = new Kostka(6);
                    x = kostka.UK;
                    kostka = new Kostka(4);
                    y = kostka.UK + x;

                    if ((y == 2) | (y == 3))
                        mistoA = "vestkyne";
                    if ((y == 4) | (y == 5))
                        mistoA = "prameny";
                    if (y == 6)
                        mistoA = "kostel";
                    if (y == 7)
                        mistoA = null;
                    if (y == 8)
                        mistoA = "hrbitov";
                    if (y == 9)
                        mistoA = "les";
                    if (y == 10)
                        mistoA = "kruh";

                    x = 2;
                    y = 0;

                    if (mistoA == cervenka.misto)
                        cervenka.zraneni = cervenka.zraneni + x;
                    if (karta.modra[13, 3] == "cervenka")
                        cervenka.zraneni = cervenka.zraneni + y;

                    if (mistoA == zelenka.misto)
                        zelenka.zraneni = zelenka.zraneni + x;
                    if (karta.modra[13, 3] == "zelenka")
                        zelenka.zraneni = zelenka.zraneni + y;

                    if (mistoA == modrak.misto)
                        modrak.zraneni = modrak.zraneni + x;
                    if (karta.modra[13, 3] == "modrak")
                        modrak.zraneni = modrak.zraneni + y;

                    if (mistoA == zlutka.misto)
                        zlutka.zraneni = zlutka.zraneni + x;
                    if (karta.modra[13, 3] == "zlutka")
                        zlutka.zraneni = zlutka.zraneni + y;

                    vysledek = ("Dynamit vybouchl na poli " + mistoA + ".");

                    if (mistoA == null)
                        vysledek = "Padla sedmicka, zadny vybuch se nekoná.";

                    MessageBox.Show(vysledek, "Vyhodnocení", MessageBoxButtons.OK,
MessageBoxIcon.Information);
```

```
                    cil = "all";
                }
```

Jediné vybavení z balíčku Červených karet, které se řeší v této části kódu je vybavení zvyšující útočné číslo svého vlastníka. Zbylé vybavení má své efekty v jiné fázi kola, a proto se jeho efekty řeší v příslušné fázi. Přidává o jedno zranění víc k útoku, pokud byl útok dle pravidel hry úspěšný.

```
                //VYBAVENI
                if ((jmenoK == "Sekáček na maso") | (jmenoK == "Řetězová pila") |
(jmenoK == "Zrezivělá tesařská sekera"))
                {
                    x = 0;

                    while (aktivni != hraci[x])
                    {
                        x++;
                    }
                    utc[x] = utc[x] + 1;
                }
```

Následuje řešení Modrých karet. První je Výbuch spravedlnosti, který zraňuje všechny hráče kromě aktivního. Kód je řešen velmi podobně jako u Dynamitu.

```
if (jmenoK == "Výbuch spravedlnosti")
                {
                    int r = 2;
                    int g = 2;
                    int b = 2;
                    int y = 2;

                    if (aktivni == "cervenka")
                        r = 0;
                    if (aktivni == "zelenka")
                        g = 0;
                    if (aktivni == "modrak")
                        b = 0;
                    if (aktivni == "zlutka")
                        y = 0;

                    cervenka.zraneni = cervenka.zraneni + r;
                    zelenka.zraneni = zelenka.zraneni + g;
                    modrak.zraneni = modrak.zraneni + b;
                    zlutka.zraneni = zlutka.zraneni + y;

                    cil = "all";
                }
```

Svěcená voda snižuje zranění aktivního hráče o dva body.

```
                if (jmenoK == "Léčivá svěcená voda")
                {
                    mistoHP = -2;
                    cil = aktivni;
                }
```

Karta Zjevení je speciálním léčením pro vlkodlaky. Kód je stejný jako u karty Pekelný obřad, pouze rasa, která se ověřuje, je vlkodlak nikoliv upír.

Zrcadlo odčarování mění stav aktivního hráče na odhalen, pokud aktivní hráč nehraje postavu jménem Ursana. Efekt karty se nevztahuje na rasu člověk, ale ta v této verzi hry není zastoupena žádnou postavou, a tak Zrcadlo nebere tuto možnost v potaz.

```
if (jmenoK == "Zrcadlo odčarování")
  {
      if ((aktivni == "cervenka") & (cervenka.jmeno != "Ursana"))
          cervenka.stav = "odhalen";
      if ((aktivni == "zelenka") & (zelenka.jmeno != "Ursana"))
          zelenka.stav = "odhalen";
      if ((aktivni == "modrak") & (modrak.jmeno != "Ursana"))
          modrak.stav = "odhalen";
      if ((aktivni == "zlutka") & (zlutka.jmeno != "Ursana"))
          zlutka.stav = "odhalen";
  }
```

Karta Krvavý úplněk změní hodnotu zranění cíle na sedm. AI zjišťuje, zda některý z jeho cílů má zranění nižší než sedm, pokud tomu tak je, aplikuje na něj efekt této karty. Pokud jsou oba cíle pod hranicí sedmi a aktivní hráč též, aplikuje AI efekt karty sama na sebe k vyléčení.

```
if (jmenoK == "Krvavý úplněk")
  {
      if (aktivni != "modrak")
      {
          y = 0;
          x = 0;

          while (cil1 != cile[y])
          {
              y++;
          }

          while (cil2 != cile[x])
          {
              x++;
          }

          if ((zran[y] > zran[x]) & (zran[x] < 7))
          {
              cil = cil2;
          }
          else if ((zran[x] > zran[y]) & (zran[y] < 7))
          {
              cil = cil1;
          }
          else
          {
              x = 0;
              while (aktivni != hraci[x])
              {
                  x++;
              }
              if (zran[x] > 7)
                  cil = aktivni;
          }
```

```
                }
                switch (cil)
                {
                    case "cervenka":
                        cervenka.zraneni = 7;
                        break;
                    case "zelenka":
                        zelenka.zraneni = 7;
                        break;
                    case "modrak":
                        modrak.zraneni = 7;
                        break;
                    case "zlutka":
                        zlutka.zraneni = 7;
                        break;
                }
            }
```

Požehnání léčí vybraného hráče o počet životů daných hodem šestistěnnou kostkou. Aktivní hráč nemůže zvolit za cíl efektu sám sebe. Uživatel, pokud se o toto pokusí, je upozorněn chybovou hláškou. AI vybírá za cíl hráče, který není ani jedním z cílů.

```
if (jmenoK == "Požehnání")
            {
                kostka = new Kostka(6);
                x = kostka.UK;
                y = 0;

                if (aktivni != "modrak")
                {
                    while (cil == "")
                    {
                        if ((hraci[y] != cil1) | (hraci[y] != cil1) | (hraci[y]
!= aktivni))
                            cil = hraci[y];
                        else
                            y++;
                    }
                }

                if ((aktivni == "modrak") & (cil == "modrak"))
                {
                    MessageBox.Show("Nemůžes cílovat sám sebe. Vyber si jiný
cíl.", "Chyba", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }

                mistoHP = -x;
            }
```

Utajené vědomosti aktivují ve fázi hry Konec kola nové kolo právě aktivního hráče.

```
            if (jmenoK == "Utajené vědomosti")
            {
                odkladiste = "nove_kolo";
            }
```

Karta Čokoláda funguje podobně jako Požehnání a Pekelný obřad. Pouze místo ověření rasy ověřuje, jaké má postava jméno a zda splňuje podmínku pro spuštění efektu

karty. AI se opět odhaluje pouze v případě, že již utržené zranění přesahuje či se rovná polovině životů postavy.

Následně se řeší dvě karty Vybavení. Obě tyto karty se řeší i později, neboť u karty Longinovo kopí se v této fázi kola nemusí jeho efekt aktivovat a efekt Roucha se vztahuje i na útočníka, tudíž zde se řeší pouze část efektu dopadající na vlastníka předmětu.

```
            //Kopi
              if(jmenoK == "Longinovo kopí")
              {
                  x = 0;

                  while (aktivni != hraci[x])
                  {
                      x++;
                  }

                  utcA = utc[x];
                  rasaA = rasa[x];

                  //overeni rasy
                  if (rasaA == "vlkodlak")
                  {
                      //AI
                      //Pokud je hrac odhalen a splnuje podminku
                      if (stavA == "odhalen")
                      {
                          utc[x] = utc[x] + 2;
                          kopi = 1;
                      }
                  }

                  cervenka.utocneC = utc[0];
                  zelenka.utocneC = utc[1];
                  modrak.utocneC = utc[2];
                  zlutka.utocneC = utc[3];
              }

              //Roucho - tu se resi jen cast tykajici se vlastnika v boji ta
tykajici se utocnika
              if (jmenoK == "Svaté roucho")
              {
                  if (aktivni == "cervenka")
                      cervenka.utocneC = cervenka.utocneC - 1;
                  if (aktivni == "zelenka")
                      zelenka.utocneC = zelenka.utocneC - 1;
                  if (aktivni == "modrak")
                      modrak.utocneC = modrak.utocneC - 1;
                  if (aktivni == "zlutka")
                      zlutka.utocneC = zlutka.utocneC - 1;
              }
```

Další vybavení se řeší v příslušných fázích kola, kdy jsou jejich efekty uplatňovány. Dále se provede aktualizace paměti AI, zapsaní získaných předmětů

do inventáře postav, aktualizace útočného čísla postav a zajištění, aby při záporné hodnotě zranění nevypadla figurka z počítadla zranění.

```
//RGBY
                cervenka.pametG = pametR[1];
                cervenka.pametB = pametR[2];
                cervenka.pametY = pametR[3];

                zelenka.pametR = pametG[0];
                zelenka.pametB = pametG[2];
                zelenka.pametY = pametG[3];

                zlutka.pametR = pametY[0];
                zlutka.pametG = pametY[1];
                zlutka.pametB = pametY[2];


                //inventare
                itemR11.Text = "";
                itemG11.Text = "";
                itemB11.Text = "";
                itemY11.Text = "";
                for (int i = 0; i < 16; i++)
                {

                    if (karta.modra[i, 3] == "cervenka")
                        itemR11.Text = itemR11.Text + ", " + karta.modra[i, 0];
                    if (karta.cervena[i, 3] == "cervenka")
                        itemR11.Text = itemR11.Text + ", " + karta.cervena[i, 0];

                    if (karta.modra[i, 3] == "zelenka")
                        itemG11.Text = itemG11.Text + ", " + karta.modra[i, 0];
                    if (karta.cervena[i, 3] == "zelenka")
                        itemG11.Text = itemG11.Text + ", " + karta.cervena[i, 0];

                    if (karta.modra[i, 3] == "modrak")
                        itemB11.Text = itemB11.Text + ", " + karta.modra[i, 0];
                    if (karta.cervena[i, 3] == "modrak")
                        itemB11.Text = itemB11.Text + ", " + karta.cervena[i, 0];

                    if (karta.modra[i, 3] == "zlutka")
                        itemY11.Text = itemY11.Text + ", " + karta.modra[i, 0];
                    if (karta.cervena[i, 3] == "zlutka")
                        itemY11.Text = itemY11.Text + ", " + karta.cervena[i, 0];
                }

                cervenka.utocneC = utc[0];
                zelenka.utocneC = utc[1];
                modrak.utocneC = utc[2];
                zlutka.utocneC = utc[3];

                //nulace zraneni pokud jsou niz nez je 0
                if (cervenka.zraneni < 0)
                    cervenka.zraneni = 0;
                if (zelenka.zraneni < 0)
                    zelenka.zraneni = 0;
                if (modrak.zraneni < 0)
                    modrak.zraneni = 0;
                if (zlutka.zraneni < 0)
```

```
                    zlutka.zraneni = 0;
```

Následně se pohnou figurky na počítadle zranění jak pro aktivního hráče, tak pro jeho cíl. Kód je stejný jako ve fázi Pohyb, a proto je zde jen uveden kód, kdy jsou cílem všichni hráči.

```
//vybuch spravedlnosti reseni a dynamitu
                if(cil == "all")
                {
                    cil = "cervenka";
                    zraneni = new Zraneni(cil, cervenka.zraneni, zelenka.zraneni,
modrak.zraneni, zlutka.zraneni);
                    red_hp.Location = new Point(zacatek.hp_rx, zraneni.y);

                    cil = "zelenka";
                    zraneni = new Zraneni(cil, cervenka.zraneni, zelenka.zraneni,
modrak.zraneni, zlutka.zraneni);
                    green_hp.Location = new Point(zacatek.hp_gx, zraneni.y);

                    cil = "modrak";
                    zraneni = new Zraneni(cil, cervenka.zraneni, zelenka.zraneni,
modrak.zraneni, zlutka.zraneni);
                    blue_hp.Location = new Point(zacatek.hp_bx, zraneni.y);

                    cil = "zlutka";
                    zraneni = new Zraneni(cil, cervenka.zraneni, zelenka.zraneni,
modrak.zraneni, zlutka.zraneni);
                    yellow_hp.Location = new Point(zacatek.hp_yx, zraneni.y);
                }
```

Poté se opět ověřuje, zda nebyl některý z hráčů vyřazen. Tento kód byl opět již řešen v jedné z předchozích fází kola.

Nyní se provede změna fáze, vynulování hodnot kostek, zrušení výběru cíle uživatelem a aktualizace vypsaného stavu uživatelovy postavy. A tímto tato fáze končí.

```
                fazovac.Text = "Boj";
                //nahozeni kostek do puvodni pozice kvuli osetreni chyby
                sestka.Text = "0";
                ctyrka.Text = "0";
                //nulace vyberu
                hr1.Checked = false;
                hr2.Checked = false;
                hr3.Checked = false;
                hr4.Checked = false;

                //aktualizace stavu postav ciste pro test

                stavB.Text = modrak.stav;

                return;
            }
```

4.8.4.4 Boj

V této fázi se řeší vše, co se týká napadení hráčů aktivním hráčem. Prvně se vynulují používané proměnné a ověří se, zda se nejedná o útok v rámci prvního kola hry, kdy je taková akce zakázaná. To se děje ve třídě Error.

```
            if (fazovac.Text == "Boj")
             {
                 //nulace promenych
                 cil = "";
                 cil1 = "";
                 cil2 = "";
                 //urceni statu aktivniho hrace
                 x = 0;
                 ////urceni statu cile hrace
                 y = 0;

                 //v prvnim kole nelze utocit
                 Error error = new Error(pocet_tahu);
                 if (error.chybka != "")
                 {
                     fazovac.Text = "Konec kola";

                     //reseni kdyz je na tahu hrac
                     if ((aktivni == "modrak") & (hr1.Checked | hr2.Checked |
hr3.Checked))
                     {
                         MessageBox.Show(error.chybka, "Chyba", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                         hr1.Checked = false;
                         hr2.Checked = false;
                         hr4.Checked = false;
                     }
                     return;
                 }
```

Ověření ve třídě Error je jednoduché, porovná se pouze kolikátý tah hry je a pokud je jeho hodnota příliš nízká, k boji nedojde. Uživatel je na tuto událost upozorněn chybovou hláškou.

```
public Error(int pocet_tahu)
        {
            if (pocet_tahu <= 5)
            {
                chybka = "Během prvního kola, nelze útočit.";
            }
            else
                chybka = "";
        }
```

Pokud má k boji dojít, je nutné u všech hráčů zjistit, v jakém lovišti se nacházejí.

```
//urceni lovist vsech postav
                 pohyb = new Pohyb(cervenka.misto, zelenka.misto, modrak.misto,
zlutka.misto);
                 cervenka.loviste = pohyb.lokaceR;
                 zelenka.loviste = pohyb.lokaceG;
                 modrak.loviste = pohyb.lokaceB;
```

```
                zlutka.loviste = pohyb.lokaceY;
```

Toto určení zařídí kód ve třídě Pohyb.

```
public Pohyb (string mistoR, string mistoG, string mistoB, string mistoY)
        {
            //modra
            if ((mistoB == "kostel") | (mistoB == "hrbitov"))
            {
                lokaceB = "A";
            }

            if ((mistoB == "kruh") | (mistoB == "prameny"))
            {
                lokaceB = "B";
            }

            if ((mistoB == "vestkyne") | (mistoB == "les"))
            {
                lokaceB = "C";
            }

            //zelena
            if ((mistoG == "kostel") | (mistoG == "hrbitov"))
            {
                lokaceG = "A";
            }

            if ((mistoG == "kruh") | (mistoG == "prameny"))
            {
                lokaceG = "B";
            }

            if ((mistoG == "vestkyne") | (mistoG == "les"))
            {
                lokaceG = "C";
            }

            //cervena
            if ((mistoR == "kostel") | (mistoR == "hrbitov"))
            {
                lokaceR = "A";
            }

            if ((mistoR == "kruh") | (mistoR == "prameny"))
            {
                lokaceR = "B";
            }

            if ((mistoR == "vestkyne") | (mistoR == "les"))
            {
                lokaceR = "C";
            }

            //zluta
            if ((mistoY == "kostel") | (mistoY == "hrbitov"))
            {
                lokaceY = "A";
            }

            if ((mistoY == "kruh") | (mistoY == "prameny"))
```

```
            {
                lokaceY = "B";
            }

            if ((mistoY == "vestkyne") | (mistoY == "les"))
            {
                lokaceY = "C";
            }
        }
    }
```

Následně se ověří, zda je některý z hráčů v lovišti aktivního hráče a to v případě, že aktivní hráč nemá předmět Pistole, který loviště upravuje.

```
                if (aktivni != karta.cervena[15, 3])
                {

                    error = new Error(aktivni, cervenka.loviste, zelenka.loviste,
modrak.loviste, zlutka.loviste);

                    if (error.chybka != "")
                    {
                        fazovac.Text = "Konec kola";
                        //reseni kdyz je na tahu hrac
                        if ((aktivni == "modrak") & (hr1.Checked | hr2.Checked |
hr3.Checked))
                        {
                            MessageBox.Show(error.chybka, "Chyba",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                            hr1.Checked = false;
                            hr2.Checked = false;
                            hr3.Checked = false;
                            hr4.Checked = false;
                        }
                        return;
                    }
                }
```

Daná situace se opět řeší ve třídě Error.

```
public Error(string aktivni, string lokaceR, string lokaceG, string lokaceB, string
lokaceY)
        {
            if ((aktivni == "modrak") & (lokaceB != lokaceG) & (lokaceB != lokaceR)
& (lokaceB != lokaceY))
            {
                chybka = "Nelze zaútočit. Nikdo není ve tvém lovišti.";
            }
            else if ((aktivni == "cervenka") & (lokaceR != lokaceG) & (lokaceR !=
lokaceB) & (lokaceR != lokaceY))
            {
                chybka = "Nelze zaútočit. Nikdo není ve tvém lovišti.";
            }
            else if ((aktivni == "zelenka") & (lokaceG != lokaceB) & (lokaceG !=
lokaceR) & (lokaceG != lokaceY))
            {
                chybka = "Nelze zaútočit. Nikdo není ve tvém lovišti.";
            }
            else if ((aktivni == "zlutka") & (lokaceY != lokaceG) & (lokaceY !=
lokaceR) & (lokaceY != lokaceB))
            {
```

```
                chybka = "Nelze zaútočit. Nikdo není ve tvém lovišti.";
            }
            else
                chybka = "";
        }
```

Pokud se žádná chyba nevyskytne, může se přikročit k samotnému boji. První na pořadu je řešení volby cíle pro uživatele, neb se v některých ohledech liší od kódu vyhledání a volby cíle pro AI.

Nejdříve se ověřuje, zda uživatel hodil kostkami na útok, pokud tomu tak není, je vyzván chybovou hláškou k nápravě.

```
if ((aktivni == "modrak") & (hr1.Checked | hr2.Checked | hr3.Checked | hr4.Checked))
                {
                    //nejde provest utok pokud nebylo hozeno kostkami
                    error = new Error(sestka.Text, ctyrka.Text);
                    if (error.chybka != "")
                    {
                        MessageBox.Show(error.chybka, "Chyba", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        return;
                    }
```

Chybová hláška se opět řeší ve třídě Error.

```
public Error(string sestK, string ctyriK)
        {
            if (sestK == "0" || ctyriK == "0")
            {
                chybka = "Nelze se pohnout. Nebyl proveden hod kostkami.";
            }
            else
                chybka = "";
        }
```

Následně je provedena volba cíle, určení jeho pořadí v rámci polí a ověření, zda uživatel nezkouší napadnout sám sebe, což je akce zakázaná pravidly hry.

```
// reseni cile
                    if (hr1.Checked)
                        cil = "cervenka";
                    if (hr2.Checked)
                        cil = "zelenka";
                    if (hr4.Checked)
                        cil = "zlutka";

                    if (cil != "")
                    {
                        while (cil != cile[y])
                        {
                            y++;
                        }
                    }

                    //hrac nemuze napadnout sam sebe
                    /*if (hr3.Checked)
                        error = new Error();
                    */
                    if (hr3.Checked)
```

```
                    {
                        error = new Error();
                        MessageBox.Show(error.chybka, "Chyba", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        sestka.Text = "0";
                        ctyrka.Text = "0";
                        hr3.Checked = false;

                        return;
                    }
```

Poté se řeší, zdali je vybraný cíl v lovišti aktivního hráče, a to i v případě, že aktivní hráč má vybavení Pistole, které upravuje loviště. Také se řeší, zda cíl nemá ve svém držení kartu Anděl strážný, která ho chrání před napadením.

```
error = new Error(aktivni, cil, cervenka.loviste, zelenka.loviste, modrak.loviste,
zlutka.loviste, karta.cervena[15, 3]);

                    if (error.chybka != "")
                    {
                        MessageBox.Show(error.chybka, "Chyba", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        hr1.Checked = false;
                        hr2.Checked = false;
                        hr4.Checked = false;

                        return;
                    }

                    //reseni andela pro hrace
                    error = new Error(aktivni, cil, karta.modra[0, 3]);
                    if (error.chybka != "")
                    {
                        MessageBox.Show(error.chybka, "Chyba", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        return;
                    }
```

Opět se to řeší v třídě Error. Řešení loviště:

```
  public Error(string aktivni, string cil1, string cil2, string lovisteR, string
lovisteG, string lovisteB, string lovisteY, string pistole)
        {
            zran1 = "";
            zran2 = "";

            if ((cil1 == "cervenka") | (cil2 == "cervenka"))
                if (zran1 == "")
                    zran1 = lovisteR;
                else
                    zran2 = lovisteR;

            if ((cil1 == "zelenka") | (cil2 == "zelenka"))
                if (zran1 == "")
                    zran1 = lovisteG;
                else
                    zran2 = lovisteG;

            if ((cil1 == "modrak") | (cil2 == "modrak"))
                if (zran1 == "")
```

```
                    zran1 = lovisteB;
                else
                    zran2 = lovisteB;

            if ((cil1 == "zlutka") | (cil2 == "zlutka"))
                if (zran1 == "")
                    zran1 = lovisteY;
                else
                    zran2 = lovisteY;

            if (aktivni == "cervenka")
                lovisteA = lovisteR;
            if (aktivni == "zelenka")
                lovisteA = lovisteG;
            if (aktivni == "modrak")
                lovisteA = lovisteB;
            if (aktivni == "zlutka")
                lovisteA = lovisteY;

            //reseni Pistole po prve
            if (aktivni == pistole)
            {
                if ((lovisteA == zran1) & (lovisteA == zran2))
                {
                    chybka = "Nelze zaútočit. Žádný z cílů není ve tvém lovišti.";
                }
                else
                    chybka = "";
            }
            else
            {
                if ((lovisteA != zran1) & (lovisteA != zran2))
                {
                    chybka = "Nelze zaútočit. Žádný z cílů není ve tvém lovišti.";
                }
                else
                    chybka = "";
            }
        }
```

Řešení Modré karty Anděl Strážný:

```
//Andel
        public Error(string aktivni, string cil, string andel)
        {
            if(cil == andel)
                chybka = "Tvůj cíl nemůže být napaden útokem, neb má Anděla
strážného.";
            else
                chybka = "";
        }
```

Nyní přijde na řadu obdobný kód, ale pro použití AI. Opět se prvně řeší loviště, následuje řešení karty Anděl strážný pro oba potencionální cíle. Ověření loviště i v případě, že aktivní hráč vlastní předmět Pistole, vyhledání pořadového čísla cíle pro použití v rámci příslušných polí a hod kostkami na útok.

```
else if (aktivni != "modrak")
                {
```

```
                    //zjisteni zda je cil1 a cil2 jsou v mem lovisti vcetne reseni
Pistole
                    error = new Error(aktivni, cil1, cil2, cervenka.loviste,
zelenka.loviste, modrak.loviste, zlutka.loviste, karta.cervena[15, 3]);
                    if (error.chybka != "")
                    {
                        fazovac.Text = "Konec kola";
                        //protoze zadny cil neni v mem lovisti

                        return;
                    }

                    //reseni andela
                    error = new Error(aktivni, cil1, karta.modra[0, 3]);
                    if (error.chybka != "")
                    {
                        cil1 = "";
                    }
                    error = new Error(aktivni, cil2, karta.modra[0, 3]);
                    if (error.chybka != "")
                    {
                        cil2 = "";
                    }
                    //overeni zda cil je skutecne v lovisti vcetne reseni Pistole
                    cilLov = new Boj(cil, aktivni, cervenka.loviste,
zelenka.loviste, modrak.loviste, zlutka.loviste, karta.cervena[15, 3]);

                    if (cilLov.cil != cil)
                        cilLov = new Boj(cil2, aktivni, cervenka.loviste,
zelenka.loviste, modrak.loviste, zlutka.loviste, karta.cervena[15, 3]);

                    cil = cilLov.cil;

                    if (cil == null)
                        cil = "";

                    if (cil != "")
                    {
                        while (cil != cile[y])
                        {
                            y++;
                        }
                    }

                    //hod na boj
                    kostka = new Kostka(6);
                    sestka.Text = Convert.ToString(kostka.UK);
                    kostka = new Kostka(4);
                    ctyrka.Text = Convert.ToString(kostka.UK);
                }
```

Dále je třeba znovu vyřešit efekt vybavení karty Longinovo kopí, protože během hry mohlo dojít ke změně stavu (hráč se mohl odhalit, mohl se změnit vlastník).

```
//Kopi po druhe
                if ((aktivni == karta.modra[10, 3]) & (kopi == 0))
                {
                    utcA = utc[x];
                    rasaA = rasa[x];
```

```
                    //overeni rasy
                    if (rasaA == "vlkodlak")
                    {
                        //Pokud je hrac odhalen a splnuje podminku
                        if (stavA == "odhalen")
                        {
                            utc[x] = utc[x] + 2;
                            kopi = 1;
                        }

                    }

                    cervenka.utocneC = utc[0];
                    zelenka.utocneC = utc[1];
                    modrak.utocneC = utc[2];
                    zlutka.utocneC = utc[3];
                }
```

A až nyní se k řešení dostává samotný útok. To se děje v třídě Boj. /tok se dělí na dvě varianty a to pokud má či nemá aktivní hráč vybavení Pochodeň.

```
//Pochoden + Valeria schopnost
                if ((aktivni == karta.cervena[14, 3]) | ((jmeno[x] ==
"Valeria")&(stav[x] == "odhalen") & (schopnosti[x] == 0)))
                {
                    masakr = new Boj(aktivni, ctyrka.Text,"0", cervenka.utocneC,
zelenka.utocneC, modrak.utocneC,
                    zlutka.utocneC);
                }
                else
                {
                    //cil nepotrebuji - Misto nej pak prijde moznost vybrat si
pravidla pro boj na zacatku hry
                    masakr = new Boj(aktivni, sestka.Text, ctyrka.Text,
cervenka.utocneC, zelenka.utocneC, modrak.utocneC,
                    zlutka.utocneC);
                }
```

Do třídy Boj se načte aktivní hráč, hodnoty kostek, které jsou následně převedeny na čísla a útočná čísla jednotlivých postav. Program vyhodnotí, zda byl útok úspěšný a pokud tomu tak je, přičte k hodnotě zranění útočné číslo aktuálního hráče.

```
public Boj (string aktivni, string sestka, string ctyrka, int utR, int utG, int utB,
int utY)
        {
            i = Convert.ToInt32(sestka);
            j = Convert.ToInt32(ctyrka);
            // -3 nejde ve hre ziskat. Minimalni hodnota dosazitelna kartami hry je
-1
            ut = -3;

            z = i - j;

            i = 0;
            string[] hraci = { "cervenka", "zelenka", "modrak", "zlutka" };
            int[] utocneC = { utR, utG, utB, utY };

            while (ut == -3)
            {

                if (hraci[i] == aktivni)
```

```
                    ut = utocneC[i];
                else
                    i++;
            }


            if (z < 1)
            {
                vysledek = "Tvůj útok nebyl úspěšný";
                z = 0;
            }

            //modrak
            if (z > 0)
            {
                z = z + ut;
                vysledek = "Tvůj útok byl úspěšný";
            }
        }
```

Poté je vypsán uživateli v hlavním kódu výsledek útoku z třídy Boj, za předpokladu že byl aktivním hráčem. Dále je třeba ošetřit plošné zranění způsobené vybavením Kulomet, stejně tak jako efekt karty Anděl Strážný, pokud by ho náhodou někdo ze zasažených hráčů měl.

```
//vypsani vysledku utoku pro hrace
                if ((aktivni == "modrak") & (hr1.Checked | hr2.Checked | hr3.Checked
| hr4.Checked))
                {
                    MessageBox.Show(masakr.vysledek, "Výsledek útoku",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
zranA = 0;

                //Kulomet + reseni andela z plosneho dmg
                if (cilLov != null)
                {
                    if (aktivni == karta.cervena[10, 3])
                    {

                        if ((cervenka.loviste == cilLov.lovisteA) & (aktivni !=
"cervenka"))
                        {
                            if (karta.modra[0, 3] == "cervenka")
                                zranA = -masakr.z;

                            cervenka.zraneni = cervenka.zraneni + masakr.z + zranA;
                        }

                        if ((zelenka.loviste == cilLov.lovisteA) & (aktivni !=
"zelenka"))
                        {
                            if (karta.modra[0, 3] == "zelenka")
                                zranA = -masakr.z;

                            zelenka.zraneni = zelenka.zraneni + masakr.z + zranA;
                        }
```

```
                        if ((modrak.loviste == cilLov.lovisteA) & (aktivni !=
"modrak"))
                        {
                            if (karta.modra[0, 3] == "modrak")
                                zranA = -masakr.z;

                            modrak.zraneni = modrak.zraneni + masakr.z + zranA;
                        }

                        if ((zlutka.loviste == cilLov.lovisteA) & (aktivni !=
"zlutka"))
                        {
                            if (karta.modra[0, 3] == "zlutka")
                                zranA = -masakr.z;

                            zlutka.zraneni = zlutka.zraneni + masakr.z + zranA;
                        }
                        cil = "all";
                    }
                }
```

Nyní je třeba vyřešit zranění a posun figurek po počítadle zranění. Kód je obdobný, jak již bylo popsáno dříve, jen v tomto případě se u veškerého zranění ověřuje, zda cíl nemá ve svém vlastnictví vybavení Roucho, které zranění ovlivňuje. Jako názorný příklad je uveden kód týkající se hráče Červenka pro oba případy zranění.

```
switch (cil)
                {
                    case "cervenka":
                        if (karta.modra[14, 3] == "cervenka")
                        {
                            zranA = masakr.z - 1;
                            if (zranA < 0)
                                zranA = 0;

                            cervenka.zraneni = cervenka.zraneni + zranA;
                        }
                        else
                            cervenka.zraneni = cervenka.zraneni + masakr.z;

                        zraneni = new Zraneni(cil, cervenka.zraneni,
zelenka.zraneni, modrak.zraneni, zlutka.zraneni);
                        red_hp.Location = new Point(zacatek.hp_rx, zraneni.y);
                        break;
```

A nyní řešení pokud zranění dostává více hráčů.

```
if (cil == "all")
                {
                    cil = "cervenka";
                    if (karta.modra[14, 3] == cil)
                        cervenka.zraneni = cervenka.zraneni - 1;
                    zraneni = new Zraneni(cil, cervenka.zraneni, zelenka.zraneni,
modrak.zraneni, zlutka.zraneni);
                    red_hp.Location = new Point(zacatek.hp_rx, zraneni.y);
```

Po udělení zranění se ověřuje, zdali nebyl některý z hráčů vyřazen, kód tohoto ověření byl již uveden v předchozích fázích kola.

Pokud byl některý z hráčů v této fázi vyřazen a aktivní hráč má ve svém vlastnictví předmět Růženec, aktivuje se efekt této karty. Následně tato fáze končí a začíná fáze konce kola.

```
//Ruzenec
                if(aktivni == karta.modra[11,3])
                {
                    x = 0;
                    while (cil == hraci[x])
                    {
                        x++;
                    }

                    if (stav[x] == "vyrazen")
                    {
                        for (y = 0; y < 15; y++ )
                        {
                            if (karta.modra[y, 3] == cil)
                                karta.modra[y, 3] = aktivni;
                            if (karta.cervena[y, 3] == cil)
                                karta.cervena[y, 3] = aktivni;
                        }
                    }
                }

                fazovac.Text = "Konec kola";
                return;
            }
```

4.8.4.5 Konec kola

Tato fáze ukončuje kolo hráče. Zjišťuje, zda nebyl aktivován efekt Modré karty Utajené vědomosti a pokud ano, bude aktivním hráčem hráč, jehož kolo právě končí. Jsou nastaveny hodnoty nového aktivního hráče a spočítají se vyřazené postavy za jednotlivé rasy. Dle toho je případně vyhlášeno, která strana vyhrála.

```
            if (fazovac.Text == "Konec kola")
            {
                //zruseni vybehu hracu
                hr1.Checked = false;
                hr2.Checked = false;
                hr3.Checked = false;
                hr4.Checked = false;

                fazovac.Text = "Pohyb";
                //nahozeni kostek do puvodni pozice kvuli osetreni chyby
                sestka.Text = "0";
                ctyrka.Text = "0";

                //reseni utajenych vedomosti pripadne schopnosti
                if (odkladiste == "nove_kolo")
                {
                    barvaN = barvaA;
                }
```

```
                //reseni zda jiz nenastal konec hry tzn nevyhrala jedna ze stran
                int upir = 0;
                int vlkodlak = 0;
                //cervenka
                if (cervenka.stav == "vyrazen")
                {
                    if (cervenka.rasa == "upír")
                        upir++;
                    if (cervenka.rasa == "vlkodlak")
                        vlkodlak++;
                }
                //zelenka
                if (zelenka.stav == "vyrazen")
                {
                    if (zelenka.rasa == "upír")
                        upir++;
                    if (zelenka.rasa == "vlkodlak")
                        vlkodlak++;
                }
                //modrak
                if (modrak.stav == "vyrazen")
                {
                    if (modrak.rasa == "upír")
                        upir++;
                    if (modrak.rasa == "vlkodlak")
                        vlkodlak++;
                }
                //zlutka
                if (zlutka.stav == "vyrazen")
                {
                    if (zlutka.rasa == "upír")
                        upir++;
                    if (zlutka.rasa == "vlkodlak")
                        vlkodlak++;
                }

                pocet_vyrazenych = upir + vlkodlak;

                fazovac.ForeColor = barvaN;

                #region Aktivni hrac
                //modrak - clovek
                if ((fazovac.ForeColor == Color.Navy) & (modrak.stav != "vyrazen"))
                {
                    aktivni = "modrak";
                    barvaN = modrak.barvaN;
                    barvaA = modrak.barvaA;
                }
                else if ((fazovac.ForeColor == Color.Navy) & (modrak.stav ==
"vyrazen"))
                {
                    fazovac.Text = "Pohyb";
                    fazovac.ForeColor = modrak.barvaN;
                    blue.Location = new Point(zacatek.f_bx, zacatek.f_y);
                    blue_hp.Location = new Point(zacatek.f_bx, zacatek.f_y);
                    sestka.Text = "0";
                    ctyrka.Text = "0";
                    return;
```

```
                }

                //Cervenka
                if ((fazovac.ForeColor == Color.Red) & (cervenka.stav != "vyrazen"))
                {
                    aktivni = "cervenka";
                    barvaN = cervenka.barvaN;
                    barvaA = cervenka.barvaA;
                }
                else if ((fazovac.ForeColor == Color.Red) & (cervenka.stav ==
"vyrazen"))
                {
                    fazovac.Text = "Pohyb";
                    fazovac.ForeColor = cervenka.barvaN;
                    red.Location = new Point(zacatek.f_rx, zacatek.f_y);
                    red_hp.Location = red.Location;//new Point(zacatek.f_rx,
zacatek.f_y);
                    sestka.Text = "0";
                    ctyrka.Text = "0";
                    return;
                }

                //zelenka
                if ((fazovac.ForeColor == Color.Green) & (zelenka.stav !=
"vyrazen"))
                {
                    aktivni = "zelenka";
                    barvaN = zelenka.barvaN;
                    barvaA = zelenka.barvaA;
                }
                else if ((fazovac.ForeColor == Color.Green) & (zelenka.stav ==
"vyrazen"))
                {
                    fazovac.Text = "Pohyb";
                    fazovac.ForeColor = zelenka.barvaN;
                    green.Location = new Point(zacatek.f_gx, zacatek.f_y);
                    green_hp.Location = new Point(zacatek.f_gx, zacatek.f_y);
                    sestka.Text = "0";
                    ctyrka.Text = "0";
                    return;

                }

                //zlutka
                if ((fazovac.ForeColor == Color.Coral) & (zlutka.stav != "vyrazen"))
                {
                    aktivni = "zlutka";
                    barvaN = zlutka.barvaN;
                    barvaA = zlutka.barvaA;
                }
                else if ((fazovac.ForeColor == Color.Coral) & (zlutka.stav ==
"vyrazen"))
                {
                    fazovac.Text = "Pohyb";
                    //nahozeni kostek do puvodni pozice kvuli osetreni chyby
                    sestka.Text = "0";
                    ctyrka.Text = "0";
                    fazovac.ForeColor = zlutka.barvaN;
                    yellow.Location = new Point(zacatek.f_yx, zacatek.f_y);
```

```
                    yellow_hp.Location = new Point(zacatek.f_yx, zacatek.f_y);
                    return;
                }
                #endregion

                //vysledek
                if (upir == 2)
                {
                    MessageBox.Show("Všichni upíři byli vyřazeni. Vyhráli
vlkodlaci.", "Konec hry", MessageBoxButtons.OK, MessageBoxIcon.Stop);
                }
                else if (vlkodlak == 2)
                {
                    MessageBox.Show("Všichni vlkodlaci byli vyřazeni. Vyhráli
upíři.", "Konec hry", MessageBoxButtons.OK, MessageBoxIcon.Stop);
                }

                stavB.Text = modrak.stav;

                return;
            }
```

4.8.5 Schopnosti postav

Schopnosti postav dělíme na pasivní, které běží stále (pokud nejsou náhodou vyřazeny ze hry) a aktivní, které se spouští hráč dle svého vlastního uvážení. Aktivní schopnosti se budou řešit samostnatně jak pro uživatele, tak pro AI. Pasivní jsou popsány zde.

4.8.5.1 Pasivní schopnosti

4.8.5.1.1 Valeria

Schopnost této postavy funguje, jako by měla vybavení Pochodeň, za splnění podmínek odhalení a nezakázání její schopnosti. Schopnost se řeší ve fázi kola Boj. Pokud za tuto postavu hraje AI, tak se před bojem odhalí.

```
if (aktivni != "modrak")
                {
                    //Valeria odhaleni
                    if ((jmeno[x] == "Valeria") & (stav[x] != "odhalen"))
                        stav[x] = "odhalen";


                    cervenka.stav = stav[0];
                    zelenka.stav = stav[1];
                    modrak.stav = stav[2];
                    zlutka.stav = stav[3];
                }
if ((aktivni == karta.cervena[14, 3]) | ((jmeno[x] == "Valeria")&(stav[x] ==
"odhalen") & (schopnosti[x] == 0)))
                {
                    masakr = new Boj(aktivni, ctyrka.Text,"0", cervenka.utocneC,
zelenka.utocneC, modrak.utocneC,
```

```
                zlutka.utocneC);
            }
```

4.8.5.1.2 Viktor

Viktorovou schopností je lečení svého zranění, pokud jeho útok byl úspěšný. AI se odhalí, pokud útok uspěl, aby využilo tuto schopnost.

```
if((aktivni != "modrak") & (jmeno[x] == "Viktor") & (masakr.vysledek == "Tvůj útok
byl úspěšný") & (stav[x] != "odhalen"))
            {
                stav[x] = "odhalen";

                cervenka.stav = stav[0];
                zelenka.stav = stav[1];
                modrak.stav = stav[2];
                zlutka.stav = stav[3];
            }

            if ((jmeno[x] == "Viktor") & (masakr.vysledek == "Tvůj útok byl
úspěšný") & (stav[x] == "odhalen") & (schopnosti[x] == 0))
            {
                zran[x] = zran[x] - 2;

                cervenka.zraneni = zran[0];
                zelenka.zraneni = zran[1];
                modrak.zraneni = zran[2];
                zlutka.zraneni = zran[3];
            }
```

4.8.5.2 Schopnosti postavy uživatele

Uživatel se odhaluje pomocí tlačítka Odhal se. Toto tlačítko načte jeho rasu do paměti jednotlivých AI a umožní mu při dalším stisknutí aktivovat schopnost jeho postavy.

```
if (modrak.stav == "zahalen")
            {
                modrak.stav = "odhalen";
                OdhSchop.Text = "Použij schopnost";
                cervenka.pametB = modrak.rasa;
                zelenka.pametB = modrak.rasa;
                zlutka.pametB = modrak.rasa;
                stavB.Text = modrak.stav;
                return;
            }
```

V rámci aktivace schopnosti se provede zjištění, kdo je cíl, kde je a v jakém stavu se nachází schopnost uživatele.

```
string[] cile = { "cervenka", "zelenka", "modrak", "zlutka" };
            string[] misto = { cervenka.misto, zelenka.misto, modrak.misto,
zlutka.misto };
            int[] schopnosti = { cervenka.schop, zelenka.schop, modrak.schop,
zlutka.schop };

            cil = "";
            x = 0;
            y = 0;
```

```
                mistoHP = 0;

                // reseni cile
                if (hr1.Checked)
                    cil = "cervenka";
                if (hr2.Checked)
                    cil = "zelenka";
                if (hr4.Checked)
                    cil = "zlutka";
```

Pokud uživatel již schopnost využil nebo mu byla zakázaná a stejně se ji pokusí použít, je na nemožnost takového konání upozorněn.

```
if ((OdhSchop.Text == "Použij schopnost") & (modrak.schop == 1))
            {
                MessageBox.Show("Buď jsi svou schopnost již použil, nebo ti byla
zakázána.", "Chyba", MessageBoxButtons.OK, MessageBoxIcon.Information);
                return;
            }
```

4.8.5.2.1 Ursana

Ursaninou schopností je lhát na věštby, je to jediná schopnost nevyžadující odhalení postavy. Uživateli je dáno na výběr, za jakou rasu se chce v případě, že je cílem zelené karty, vydávat. U typu B v případě karet dotazujících se na počáteční písmeno jména postavy je otázán, zda se chce nechat zranit či nikoliv.

Tato schopnost se aktivuje automaticky a spadá do kategorie pasivních schopností, ale její zpracování pro uživatele je odlišné od zpracování pro AI, proto je uvedena zde.

```
if (typK == "A")
                {
                    //jmeno, text, typ, rasa1, rasa2, moznost1, moznost2, stav
                    if (cil == "modrak")
                    {
                        if ((jmeno[y] == "Ursana") & (schopnosti[y] == 0))
                        {

                            UrsanaR ursanaR = new UrsanaR();
                            if (ursanaR.rasaU != null)
                            {
                                odkladiste = rasa[y];
                                rasa[y] = ursanaR.rasaU;
                            }

                        }
                    }

if (typK == "B")
                {
                    //jmeno, text, typ, rasa1, moznost1, stav
                    if (cil == "modrak")
                    {
                        if ((jmeno[y] == "Ursana") & (schopnosti[y] == 0))
                        {
                            if (rasa1 == "jmena1")
                            {
```

```
                                if (MessageBox.Show("Chceš se nechat věštbou
zranit?", "Ursana", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.No)
                                    moznost1 = "0";
                            }
                            else if (rasa1 == "jmena2")
                            {
                                if (MessageBox.Show("Chceš se nechat věštbou
zranit?", "Ursana", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.No)
                                    moznost1 = "0";
                            }
                            else
                            {
                                UrsanaR ursanaR = new UrsanaR();
                                if (ursanaR.rasaU != null)
                                {
                                    odkladiste = rasa[y];
                                    rasa[y] = ursanaR.rasaU;
                                }
                            }
                        }

                    }
if (typK == "C")
                {
                    //jmeno, text, typ, rasa1, moznost1, moznost2, stav

                    if (cil == "modrak")
                    {
                        if ((jmeno[y] == "Ursana") & (schopnosti[y] == 0))
                        {

                            UrsanaR ursanaR = new UrsanaR();
                            if (ursanaR.rasaU != null)
                            {
                                odkladiste = rasa[y];
                                rasa[y] = ursanaR.rasaU;
                            }
                        }
                    }
```

Kód třídy UrsanaR:

```
public partial class UrsanaR : Form
    {
        public UrsanaR()
        {
            InitializeComponent();
        }

        public string rasaU = null;

        private void OK_Click(object sender, EventArgs e)
        {
            if (t1.Checked)
            {
                rasaU = "upír";
            }

            if (t2.Checked)
```

```
            {
                rasaU = "vlkodlak";
            }

            if (t3.Checked)
            {
                rasaU = "člověk";
            }

            if(rasaU == null)
            {
                MessageBox.Show("Nevybral jsi žádnou rasu, tedy budeš odpovídat po
pravdě.", "Upozornění", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            }
        }
    }
```

4.8.5.2.2 Ultima

Schopnost umožňuje zranit cíl, který se nachází na území Hřbitov. Samozřejmostí jsou chybové hlášky, aby uživatel nemohl podvádět.

```
if ((modrak.jmeno == "Ultima") & (modrak.schop == 0) & (modrak.stav != "vyrazen"))
                    {
                        if ((cil == "") & (modrak.jmeno == "Ultima"))
                        {
                            MessageBox.Show("Nebyl zvolen cil schopnosti.", "Chyba",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                            return;
                        }

                        if (hr3.Checked)
                        {
                            Error error = new Error();
                            MessageBox.Show(error.chybka, "Chyba",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                            hr3.Checked = false;
                            return;
                        }

                        while (cil != cile[y])
                        {
                            y++;
                        }
                        if (misto[y] == "hrbitov")
                            mistoHP = 2;
                        else
                        {
                            MessageBox.Show("Vybraný cíl se nenachází na hřbitově.
Vyber jiný.", "Chyba", MessageBoxButtons.OK, MessageBoxIcon.Error);
                            return;
                        }
                    }
```

4.8.5.2.3 Fenrir a Gabrin

Zranění vybraného cíle. Probíhá ověření, zda uživatel hodil kostkami pro určení zranění a schopnost se mění na vyčerpanou.

```
if (((modrak.jmeno == "Fenrir") | (modrak.jmeno == "Gabrin")) & (modrak.schop == 0)
& (modrak.stav != "vyrazen"))
                    {

                        switch (modrak.jmeno)
                        {
                            case "Fenrir":
                                x = Convert.ToInt32(sestka.Text);
                                ctyrka.Text = "0";
                                break;
                            case "Gabrin":
                                x = Convert.ToInt32(ctyrka.Text);
                                sestka.Text = "0";
                                break;
                        }

                        if ((sestka.Text == "0") & (ctyrka.Text == "0"))
                        {
                            MessageBox.Show("Nebylo hozeno kostkou pro urceni
zranění cíle.", "Chyba", MessageBoxButtons.OK, MessageBoxIcon.Error);
                            return;
                        }

                        if ((cil != "") & (modrak.stav == "odhalen") & (modrak.schop
== 0))
                        {
                            mistoHP = x;
                            modrak.schop = 1;
                            MessageBox.Show("Cil " + cil + " byl zraněn za: " + x +
" životů.", "Výsledek", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        }
                    }
```

4.8.5.2.4 Freya

Ukazatel zranění vybraného hráče je změněn na hodnotu sedm.

```
if ((modrak.jmeno == "Freya") & (modrak.schop == 0) & (modrak.stav != "vyrazen"))
                    {
                        if ((cil != "") & (modrak.stav == "odhalen"))
                        {
                            switch (cil)
                            {
                                case "cervenka":
                                    cervenka.zraneni = 7;
                                    break;
                                case "zelenka":
                                    zelenka.zraneni = 7;
                                    break;
                                case "modrak":
                                    modrak.zraneni = 7;
                                    break;
                                case "zlutka":
                                    zlutka.zraneni = 7;
                                    break;
                            }
                            mistoHP = 0;
                            modrak.schop = 1;
                        }
                    }
```

4.8.5.2.5 Elektra

Vybranému cíli zakáže jeho schopnost, pokud již nebyla vyčerpána. Pokud byla vyčerpána, je uživatel na tuto skutečnost upozorněn a může vybrat jiný cíl.

```
if ((modrak.jmeno == "Elektra") & (modrak.schop == 0) & (modrak.stav != "vyrazen"))
                    {
                        y = 0;

                        while (cil != cile[y])
                        {
                            y++;
                        }

                        if (schopnosti[y] == 1)
                        {
                            MessageBox.Show("Zvolený cíl má již vyčerpanou
schopnost. Vyber někoho jiného.", "Chyba", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                            return;
                        }
                        if (schopnosti[y] == 0)
                        {
                            schopnosti[y] = 1;
                            modrak.schop = 1;
                        }

                        cervenka.schop = schopnosti[0];
                        zelenka.schop = schopnosti[1];
                        //modrak.schop = schopnosti[2];
                        zlutka.schop = schopnosti[3];
                    }
```

4.8.5.2.6 Viktor a Valeria

Tyto postavy mají pasivní schopnost, která se řeší sama ve fázi boje. Uživatel je na tuto skutečnost upozorněn.

```
if (fazovac.Text == "Boj")
                {
                    if ((modrak.jmeno == "Viktor") | (modrak.jmeno == "Valeria") &
(modrak.schop == 0))
                    {
                        MessageBox.Show("Tvá schopnost je aktivní stále. Nemusíš
mačkat toto tlačítko.", "Chyba", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        return;
                    }
                }
```

4.8.5.2.7 Wren

Pokud je uživatel s touto postavou napaden, může stiskem tlačítka zaútočit na aktivního hráče. Kód je totožný s bojem uživatele ve fázi kola Boj.

4.8.5.3 Schopnosti postavy AI

AI používá aktivní shcopnosti nejdříve v druhém herním kole. Je to z toho důvodu, aby hra nezačala tím, že se veškerá AI odhalí pro použití schopnosti na cíl bez jakékoliv znalosti o ostatních hráčích.

Uživatel je vždy upozorněn, kdo na koho používá schopnost.

4.8.5.3.1 Ursana

AI se v rámci hraní této postavy chová co nejjednodušeji. Dokud je postava zahalená, tváří se za všech okolností jako vlkodlak a pokud je odhalena, nenechá se jakkoliv zranit.

```
if (typK == "A")
{

      if ((cil != "modrak") & (jmeno[y] == "Ursana") & (schopnosti[y] == 0))
                    {
                        if (stav[y] == "zahalen")
                        {
                            odkladiste = rasa[y];
                            rasa[y] = "vlkodlak";
                        }
                        if (stav[y] == "odhalen")
                        {
                            odkladiste = rasa[y];
                            string[] rasy = { "upír", "vlkodlak", "člověk" };

                            for (int i = 0; i < 3; i++)
                            {
                                if ((rasy[i] != rasa1) & (rasy[i] != rasa2))
                                    rasa[y] = rasy[i];
                            }
                        }
                    }
if (typK == "B")
{
        if ((cil == "modrak") & (jmeno[y] == "Ursana") & (schopnosti[y] == 0))
                    {
                        if (stav[y] == "zahalen")
                        {
                            odkladiste = rasa[y];
                            rasa[y] = "vlkodlak";
                        }
                        if (stav[y] == "odhalen")
                        {
                            odkladiste = rasa[y];
                            string[] rasy = { "upír", "vlkodlak", "člověk" };

                            for (int i = 0; i < 3; i++)
                            {
                                if ((rasy[i] != rasa1) & (rasy[i] != rasa2))
                                    rasa[y] = rasy[i];
                            }
```

```
                        }
                        if ((rasa1 == "jmena1") | (rasa1 == "jmena2"))
                        {
                            moznost1 = "0";
                        }
                    }
if (typK == "C")
{
      if ((cil != "modrak") & (jmeno[y] == "Ursana") & (schopnosti[y] == 0))
                    {

                        if (stav[y] == "zahalen")
                        {
                            odkladiste = rasa[y];
                            rasa[y] = "vlkodlak";
                        }
                        if (stav[y] == "odhalen")
                        {
                            odkladiste = rasa[y];
                            string[] rasy = { "upír", "vlkodlak", "člověk" };

                            for (int i = 0; i < 3; i++)
                            {
                                if (rasy[i] != rasa1)
                                    rasa[y] = rasy[i];
                            }
                        }
```

4.8.5.3.2 Wren

Schopností Wren je protiútok. Pokud je napadena ve fázi Boje, odhalí se a zaútočí zpět na aktivního hráče.

```
if((cil == cile[y]) & (jmeno[y] == "Wren") & (schopnosti[y] == 0) & (stav[y] !=
"vyrazen"))
                {
                    if (stav[y] != "odhalen")
                    {
                        stav[y] = "odhalen";

                        pametR[y] = rasa[y];
                        pametG[y] = rasa[y];
                        pametY[y] = rasa[y];
                    }
                    cil = aktivni;
                    aktivni = hraci[y];
```

Následuje totožný kód, jako už byl použit pro boj AI ve fázi kola Boj. Cílem ale je právě aktivní hráč. Po skončení tohoto protiútoku se aktivním hráčem opět stává hráč, který Wren napadl.

```
                    aktivni = cil;
                }
```

4.8.5.3.3 Ultima

Pokud je jeden z cílů na začátku kola této postavy na míste Hřbitov, AI se odhalí a pomocí schopnosti danou postavu zraní.

```
if ((jmeno[x] == "Ultima") & (stav[x] != "vyrazen") & (schopnosti[x] == 0) &
(pocet_tahu > 8))
                    {

                        while (cil1 != cile[y])
                        {
                            y++;
                        }
                        if (misto[y] == "hrbitov")
                            cil = cil1;
                        else
                        {
                            while (cil1 != cile[y])
                            {
                                y++;
                            }
                            if (misto[y] == "hrbitov")
                                cil = cil2;
                            else
                                cil = "";
                        }

                        if ((cil != "") & (stav[x] == "odhalen") & (schopnosti[x] ==
0))
                            mistoHP = 3;

                        if ((cil != "") & (stav[x] != "odhalen") & (schopnosti[x] ==
0))
                        {
                            stav[x] = "odhalen";
                            mistoHP = 3;
                            pametR[x] = rasa[x];
                            pametG[x] = rasa[x];
                            pametY[x] = rasa[x];

                            if (cil == "modrak")
                                MessageBox.Show("Byl jsi zraněn schopností hráče " +
aktivni, "Schopnost", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                            else
                                MessageBox.Show("Hrac " + cil + " byl zasažen
schopností hráče " + aktivni, "Schopnost", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                        }
                    }
```

4.8.5.3.4 Elektra

AI hrající za postavu Elektra za svůj cíl volí hráče, který je označen jako cil1. Pokud tento hráč má již schopnost vyčerpanou, volí hráče označeného jako cil2.

```
if ((jmeno[x] == "Elektra") & (stav[x] != "vyrazen") & (schopnosti[x] == 0) &
(pocet_tahu > 12))
                    {

                        y = 0;
                        while (cil1 != cile[y])
                        {
                            y++;
                        }
```

```
                        if (schopnosti[y] == 0)
                            cil = cil1;
                        else
                        {
                            y = 0;
                            while (cil2 != cile[y])
                            {
                                y++;
                            }

                            if (schopnosti[y] == 0)
                                cil = cil2;
                            else
                                cil = "";
                        }

                        if ((cil != "") & (stav[x] != "odhalen") & (schopnosti[x] ==
0))
                        {
                            stav[x] = "odhalen";
                            pametR[x] = rasa[x];
                            pametG[x] = rasa[x];
                            pametY[x] = rasa[x];
                        }

                        if ((cil != "") & (stav[x] == "odhalen") & (schopnosti[x] ==
0))
                        {
                            schopnosti[y] = 1;
                            schopnosti[x] = 1;
                        }

                        if ((cil == "modrak") & (schopnosti[x] == 1))
                            MessageBox.Show("Byl jsi zraněn schopností hráče " +
aktivni, "Schopnost", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                        else
                            MessageBox.Show("Hrac " + cil + " byl zasažen schopností
hráče " + aktivni, "Schopnost", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    }
```

4.8.5.3.5 Fenrir a Gabrin

AI se odhaluje v případě, že zranění cíle přesahuje polovinu jeho životů nebo v případě, že samotná postava je takto zraněna.

```
if (((jmeno[x] == "Fenrir") | (jmeno[x] == "Gabrin")) & (schopnosti[x] == 0) &
(stav[x] != "vyrazen") & (pocet_tahu > 12))
                    {

                        //vyber cile s vetsim zranenim, pokud je stejne nacti cil1
                        cilZran = new Boj(cil1, cil2, cervenka.zraneni,
cervenka.zraneni, modrak.zraneni, zlutka.zraneni);
                        cil = cilZran.cil;

                        y = 0;

                        while (cil != cile[y])
                        {
```

```
                            y++;
                        }

                        switch (jmeno[x])
                        {
                            case "Fenrir":
                                kostka = new Kostka(6);
                                break;
                            case "Gabrin":
                                kostka = new Kostka(4);
                                break;
                        }
                        hpC = Convert.ToInt32(hp[y]);
                        hpA = Convert.ToInt32(hp[x]);

                        //odhalit se a pouzit schopnost kdyz cil ma mene jak
polovinu hp a nebo ja
                        if ((stav[x] != "odhalen") & ( ( (hpA - zran[x] < hpA/2) ) |
((hpC - zran[y] < hpC / 2))))
                        {
                            stav[x] = "odhalen";

                            pametR[x] = rasa[x];
                            pametG[x] = rasa[x];
                            pametY[x] = rasa[x];
                        }

                        if ((cil != "") & (stav[x] == "odhalen") & (schopnosti[x] ==
0))
                        {
                            mistoHP = kostka.UK;
                            schopnosti[x] = 1;

                            if (cil == "modrak")
                                MessageBox.Show("Byl jsi zraněn schopností hráče " +
aktivni, "Schopnost", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                            else
                                MessageBox.Show("Hrac " + cil + " byl zasažen
schopností hráče " + aktivni, "Schopnost", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                        }
                    }
```

4.8.5.3.6 Freya

Schopnost této vlkodlačky v rámci AI funguje podle stejných pravidel jako efekt modré karty Krvavý měsíc v rukách AI.

```
if ((jmeno[x] == "Freya") & (stav[x] != "vyrazen") & (schopnosti[x] == 0) &
(pocet_tahu > 12))
                    {
                        y = 0;
                        x = 0;

                        while (cil1 != cile[y])
                        {
                            y++;
                        }
```

```
                        while (cil2 != cile[x])
                        {
                            x++;
                        }

                        if ((zran[y] > zran[x]) & (zran[x] < 7))
                        {
                            cil = cil2;
                        }
                        else if ((zran[x] > zran[y]) & (zran[y] < 7))
                        {
                            cil = cil1;
                        }
                        else
                        {
                            x = 0;
                            while (aktivni != hraci[x])
                            {
                                x++;
                            }
                            if (zran[x] > 7)
                                cil = aktivni;
                            else
                                cil = "";
                        }

                        x = 0;
                        while (aktivni != hraci[x])
                        {
                            x++;
                        }

                        if ((cil != "") & (stav[x] != "odhalen") & (schopnosti[x] ==
0))
                        {


                            stav[x] = "odhalen";
                            pametR[x] = rasa[x];
                            pametG[x] = rasa[x];
                            pametY[x] = rasa[x];
                        }


                        if ((cil != "") & (stav[x] == "odhalen") & (schopnosti[x] ==
0))
                        {
                            switch (cil)
                            {
                                case "cervenka":
                                    cervenka.zraneni = 7;
                                    break;
                                case "zelenka":
                                    zelenka.zraneni = 7;
                                    break;
                                case "modrak":
                                    modrak.zraneni = 7;
                                    break;
                                case "zlutka":
```

```
                                    zlutka.zraneni = 7;
                                    break;
                                }
                                schopnosti[x] = 1;

                                if (cil == "modrak")
                                    MessageBox.Show("Byl jsi zraněn schopností hráče " +
aktivni, "Schopnost", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                                else if (cil != "")
                                    MessageBox.Show("Hrac " + cil + " byl zasažen
schopností hráče " + aktivni, "Schopnost", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                            }
                        }
```

5 Závěr

V rámci předložené diplomové práce byla vytvořena aplikace v prostředí Microsoft Studio 2017 s využitím jazyka C#, která obsahuje algoritmus pro postavu uživatele a dále univerzální algoritmus pro tři hráče, které ovládá AI.

Aplikace se řídí mírně obměněnými pravidly deskové hry Lovci noci, která jsou popsána v této diplomové práci.

AI, stejně jako mladý hráč, nedokáže používat ve hře všechny informace, které se dají získat mimo jeho vlastní kolo sledováním toho, jak hrají ostatní hráči a jak na sebe vzájemně reagují (léčení, útoky, cílování jednorázových karet atd.), a proto přirovnávám úroveň hráčské znalosti AI k chování desetiletého až jedenáctiletého dítěte. Toto hodnocení vychází z mých osobních dlouholetých zkušeností vedoucího kroužku deskových her v Klubu Kapsa.

Možnosti samotné deskové hry Lovci noci nejsou mojí aplikací zcela vyčerpány, a to i z důvodu, že pravidla umožňují více modifikací, než bylo možno použít kvůli omezení rozsahu diplomové práce.

Následné modifikace jsou možné nejen v oblasti počtu hráčů, ras postav a dále i možnost předem nedefinovaného počtu příslušníků jedné rasy ve hře. Tedy že při počtu např. čtyř hráčů se střetu nemusí účastnit dva Vlkodlaci proti dvěma Upírům, ale střet může probíhat v poměru 3:1 nebo, pokud hraje i rasa Lidé, mohou být poměry znatelně jiné, např. 1:1:2, 2:1:1, 1:0:3, 0:0:4.

Další možností je náhodné rozložení herního plánu před každou novou hrou, čímž se změní herní prostředí. Hráči jsou pak nuceni, na základě vybrané postavy, lehce přizpůsobit svoji herní strategii.

Další možné úpravy jsou v oblasti uživatelského rozhraní, které v této verzi aplikace nepatří mezi nejpřívětivější. Při vývoji této aplikace se počítalo s předpokladem, že uživatel je s původní deskovou hrou Lovci noci předem seznámen a důvěrně zná její pravidla a herní mechaniky.

Závěrem lze konstatovat, že před jakýmkoliv dalším rozvojem této aplikace je nutno provést refaktorování současného kódu, jelikož se, v různých fázích herního kola, opakuje stejný kód. Stejně tak některé herní karty jsou z pohledu algoritmu víceméně stejné a liší se pouze hodnotami např. zranění.

I když možných modifikací a směrů dalšího vývoje aplikace je několik, současná verze splňuje stanovené cíle zadané v této diplomové práci.

6 Bibliografie

1. VÁVROVÁ, Alena, Jarmila NOVOTNÁ, Marta VOLFOVÁ a Antonín JANČAŘÍK. *Hry ve vyučování matematice jako významná strategie vedoucí k rozvoji klíčových kompetencí žáků* [online]. Praha: JČMF, 2006. Dostupné také z: http://people.fjfi.cvut.cz/novotant/jarmila.novotna/D04%20Hry.pdf
2. KOLÁŘOVÁ, Marie. *Desková hra*. Brno, 2010. Diplomová práce. Masarykova Univerzita. Pedagogická fakulta.
3. NOWAK, Stanislav. *Formální popis deskových her*. Praha, 2011. Bakalářská práce. Univerzita karlova. matematicko-fyzikální fakulta.
4. TEDDIES. *Historie deskových her* [online]. Žamberk: Vše o hračkách, 2016. Dostupné z: http://www.vseohrackach.cz/clanky/historie-deskovych-her.
5. *Hra go. Historie* [online]. Godó, ©2019. Dostupné z: http://hra-go.cz/index.php?option=com_content&view=article&id=48&Itemid=55.
6. BERKA, Petr. *Dobývání znalostí z databází*. Praha: Academia, 2003. ISBN 80-200-1062-9.
7. TRENZ, Oldřich; FEJFAR, Jiří; POPELKA, Ondřej; KOLOMAZNÍK, Jan a Michael ŠTENCL. *Umělá inteligence. eLearningová opora k předmětu Umělá inteligence 1* [online]. Brno: Mendelova univerzita, Provozně ekonomická fakulta, 2010. Dostupné z: https://is.mendelu.cz/eknihovna/opory/index.pl?opora=2068.
8. MATĚJKA, Jan. *Alan Turing. Průběh Turingova testu* [online]. 2017. Dostupné z: https://janmatejkacom.wordpress.com/2017/04/30/prubeh-turingova-testu/.
9. MATĚJKA, Jan. *Alan Turing. Argument čínského pokoje* [online] 2017. Dostupné z: https://janmatejkacom.wordpress.com/2017/04/30/argument-cinskeho-pokoje/.
10. LOKVENCOVÁ, Iva. *Argument čínského pokoje –včera, dnes a zítra.* Brno, 2014. Bakalářská diplomová práce. Masarykova Univerzita. Filozofická Fakulta.
11. *Lovci noci* [online]. Praha: Mindok s.r.o., 2010. Dostupné z: http://www.mindok.cz/3/hry/hry-pro-narocne-5/lovci-noci-8595558300945-74.
12. *Shadow Hunters* [online]. Board Game Geek, 2009. Dostupné z: https://boardgamegeek.com/boardgame/24068/shadow-hunters.
13. *Lovci noci. Pravidla hry.* Praha: Mindok s.r.o., 2010.
14. IKEDA, Yasutaka. *Lovci noci - Almanach.* Praha: KOMOS Verlag, 2010.

15. PETZOLD, Charles. *Programování Microsoft Windows Forms v jazyce C#: [vytváříme uživatelské rozhraní aplikací]*. Brno: Computer Press, 2006. ISBN 80-251-1058-3.16.
16. BĚHÁLEK, Marek. *Programovací jazyk C#* [online]. Ostrava, Vysoká škola báňská, 2007. Dostupné z: http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text.pdf.

7 Přílohy

Lovci noci - Almanach
Lovci noci - Pravidla hry

7.1 Komunikace s firmou Mindok s.r.o.

7.1.1 Dotaz na firmu Mindok:

Dobrý den,
jelikož mě čeká diplomová práce a z nabízených témat jsem si zvolil téma Vývoj aplikace v jazyce C# hledám vhodné téma. Napadlo mě zpracovat deskovou hru Lovci noci, kterou jste u nás vydali.
Hru bych rád zpracoval jako hru jednoho hráče proti třem hráčům počítačovým. Z originální hry bych nepoužíval grafické zpracování, ale texty (jména postav, herní karty) a řekněme algoritmus hry (pravidla).
Rád bych na to znal Váš názor.

Předem děkuji za odpověď,
Bc. Jakub Riedl

7.1.2 Svolení od firmy Mindok:

My za českou verzi s tím vůbec nemáme problém a nepředpokládám žádný problém ani od původního majitele licence. Pokud se nejedná o komerční využití, je to úplně v pořádku.