VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

Inteligentní řízení ve strategických hrách

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. DAVID KNOTEK
AUTHOR

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

Inteligentní řízení ve strategických hrách

Intelligent control in strategy games

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. DAVID KNOTEK
AUTHOR

VEDOUCÍ PRÁCE ING. HRUBÝ MARTIN, PH.D.
SUPERVISOR

BRNO 2008

Zadání

Licenční smlouva

Abstrakt

Cílem práce je vytvořit jednoduchou počítačovou hru s výraznými prvky umělé inteligence. Umělá inteligence bude v projektu zastoupena ve třech úrovních, manažerské (globální), hráčské (lokální) a asistentské (pomocné). V další fázi projektu se zaměřím na grafickou stránku, 2D simulátor zápasů, uživatelské rozhraní a následné nasazení hry do provozu. Hra bude implementována v jazyce C++, který poskytuje dostatečný programátorský komfort a jeho velkou výhodou je i multiplatformnost.

Klíčová slova

Strategická počítačová hra, umělá inteligence, hledání cesty, C++, simulace, objektově-orientované programování

David Knotek :
Inteligentní řízení ve strategických hrách, **diplomová práce, Brno, FIT VUT v Brně,** 2008

Abstract

The target of this work is a creation of simple strategic game with distinct signs of artificial intelligence. Artificial intelligence will be in project represented on three levels, managerial (global), robotically (local) and assist (associated). In the next phase of the project will be the main attention targeted on graphical side of game, 2D simulator of play, graphical interface a resulting deployment the game. The game will be implemented in language C++, which provide sufficient comfort, and may be used multi-platformly.

Keywords

Strategic computer game, artificial intelligence, path finding, C++, simulation, objects oriented programming

David Knotek :

Intelligent control in strategy games, **master's thesis, Brno, FIT VUT in Brno,** 2008

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Martina Hrubého, Ph.D. a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Brně dne 19.5.2008 David Knotek ……

Na tomto místě bych rád poděkoval vedoucímu mé diplomové práce Ing. Martinu Hrubému Ph.D. za zájem, čas a úsilí, které věnoval mé práci a pomohl mi v řadě složitých rozhodnutí.

Obsah

1. Úvod

Člověk je tvor hravý. Není proto divu, že vše, co vytvoří pro ulehčení svého života, se snaží využít i pro svoji zábavu. Jakmile vyrobil první automobil, objevily se první automobilové soutěže. Dnes je jich stovky, či spíše tisíce. Ne jinak tomu bylo a je s výpočetní technikou. Sotva se objevil první počítač, vznikla první počítačová hra. Zdokonalováním počítačového hardwaru zaznamenaly svůj velký technologický vývoj i hry. Čím dokonalejší technologie, tím vyšší požadavky na hry a tak jsme se dostali od jednoduchých textových až k dnešním, někdy téměř od reality nerozpoznatelným, hrám.

Počítačové hry se dělí na různé typy a žánry. Jedním z prvních druhů byly hry textové („textovky“) a logické, jejichž hlavní výhodou byla na tu dobu malá hardwarová náročnost. S technologickým vývojem docházelo ke spojování těchto žánrů, vznikaly adventury, v nichž počítačový svět již nebyl textový, ale grafický. Přidáváním soubojů s nepřáteli vznikaly RPG (Role Playing game) hry a dungeony. Mezi velice populární dnes řadíme hry akční. Logické úkoly zde z velké části chybí, jelikož cílem je většinou zabít co největší počet nepřátel. K trénování různých činností, které člověk vykonává v běžném životě, vznikly simulátory, které se snaží co nejvěrněji přiblížit skutečnosti.

Na závěr jsem si nechal hry strategické, které patří k mým nejoblíbenějším. Svět v nich vidíte v 2D nebo 3D pohledu. Dají se rozdělit do tří kategorií. Na hry tahové (turn-based), strategie v reálném čase (real-time) a manažerské. V manažerské strategii je cílem vybudovat co nejlepší město, nemocnici či dopravní společnost. V dalších dvou typech jde o poražení nepřítele. K dispozici máte různé druhy jednotek, které můžete zlepšovat, opravovat nebo vymýšlet nové. K tomu samozřejmě potřebujete různé suroviny, které je třeba těžit. Oba typy jsou si velice podobné, jediný rozdíl je v tom, že u tahových her hrajete s jednotkami postupně, jsou omezeny pohybovými body a soupeři se ve hře střídají. V real-time strategiích ovládáte vše naráz a důraz je kladen zejména na rychlé ovládání a rozhodování.

Právě strategická počítačová hra manažerské typu (*fotbalový manažer*) je tématem mé práce. Tvorba hry se zaměřuje především na otázku umělé inteligence na několika úrovních. Funkčnost inteligence je demonstrována ukázkovým příkladem formou 2D simulace zápasu dvou fotbalových týmů.

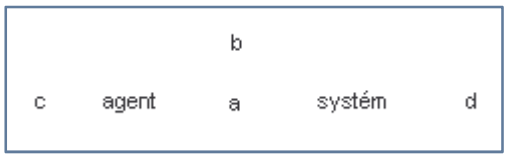
2. Teorie her

2.1. Teorie her jako formální teorie racionálního rozhodování

Moderní společnost je spjatá s potřebou koordinace aktivit velkého množství často odlišných subjektů. Její osou se staly formální organizace a její fungování má podobu formální racionality. Aktéři (skupiny i jednotlivci), kteří se v rámci moderní společnosti dostávají do formálních vztahů, tak mohou být modelováni jako racionální hráči určitých her. *„Teorie her se zabývá rozhodováním jedinců, kteří jsou v nějakém vzájemném vztahu.“* [1]

Jak je možné rozumět termínům racionálního rozhodování v té nejobecnější rovině? Racionální jednání zde přesně odpovídá Paretovu termínu *logické jednání*, kde jednající předem promýšlí adekvátní prostředky k dosažení cíle. V této souvislosti hovoří Pareto o vědomém propojení prostředků a cílů, kdy prostředky vedou k cíli nejen subjektivně, ale i objektivně.

Při pohledu na jednoho jednajícího lze uvažovat tzv. *„kybernetický“* model, kde je *agent* právě oním racionálním subjektem (aktérem), který provádí zpracování informace a rozhodnutí, a *systém*, jenž je chápán jako okolí agenta. Systém má nějaké stavy, jimž působí na rozhodnutí agenta. Jednání agenta je směřováno k systému, v jehož rámci mohou být další agenti. Jednotlivé vztahy mezi systémem (b), agentem (a) a *„jinými“* vlivy (c) na rozhodnutí agenta a pro agenta neznáme výstupy systému (d) (obrázek č. 1).



(Obrázek č. 1 - *Schéma Agent vs. Systém*)

2.2. Modely racionálního rozhodování

Modelů racionálního rozhodování existuje značné množství. Vždy se jedná o modely založené na aplikaci formálních věd. Mezi ty, které využívají především formální logiku, by patřily různé druhy analýz usuzování (usuzování za neúplné informace, nemonotónní usuzování, logika defaultů, epistémické logiky apod.).

Teorii her je aplikovaná matematická disciplína, jejíž uplatnění proniká téměř všemi humanitními vědami. Vyjdu-li z obrázku 1, dívá se teorie her na agenta jako na *hráče*, který umí posoudit důsledky své volby (tahu) v podobě nějaké hodnoty *výhry*. Volby, které jsou agentovi k dispozici, jsou jeho herními *strategiemi* a on je schopen racionálně posuzovat výhry a vytvořit si preferenční vztahy mezi strategiemi. Za základní otázku teorie her bych pak mohl použít formulaci pocházející od Johna von Neumanna z roku 1926 :

„Hraje-li n hráčů hru H, jak má hrát konkrétní hráč i, aby dosáhl pro sebe nejvýhodnějšího výsledku?“

Protože jsem výsledek hry pro konkrétního hráče definoval jako jeho výhru (hodnotu výhry), skrývá se ona racionalita hráče v jeho snaze maximalizovat zisk v hodnotě výhry. O tuto maximalizaci se snaží každý hráč, proto se někdy o hrách hovoří jako o *konfliktních situacích*.

Primárními předpoklady teorie her se tak stávají *individualismus*, *racionalita* a *vzájemná závislost* aktérů. Individualismus tu nesmí být zaměňován s izolacionizmem. Když si uvědomím, že je zde zmíněná vzájemná závislost, kdy jednání aktérů ovlivňuje ostatní jednající, i když třeba jen tím, jaká je výše výhry. Na druhé straně, i v případech, kdy se pracuje s možnou dohodou dvou a více aktérů, lze považovat každého hráče za jednotku, která jedná v duchu vlastního zájmu. V tomto smyslu se neberou do úvahy skryté motivy jednání. U kooperativních situací se tedy nevytváří koalice na základě skrytých preferencí, ale s cílem maximalizovat zisk, respektive minimalizovat ztráty.

Racionální aktér jedná ve svém vlastním zájmu. Zde by se hodil termín *instrumentální racionalita*, což je právě ona schopnost jednat na základě posouzení důsledků své volby a preferencí mezi těmito důsledky. Je zřetelné, že snaha o zjednodušení lidského jednání obsahuje v tomto druhu racionality souhru prostředků i cílů. Z vnějšího pohledu, zbaveného ohledu na skryté motivace jednání, je důležitý pouze *výsledek* jednání, důležitý je faktický dopad na ostatní účastníky hry.

2.3. Počátky teorie her

Bez nadsázky by se dalo říci, že hazardní hry stály v pozadí rozvoje matematické pravděpodobnosti v 17. století (Blaise Pascal, Pierre Fermat). Na druhé straně se mohutný rozvoj matematických metod zasloužil o to, že se matematika stala plodnou oblastí i při studiu deskových her (šachy, dáma apod.). Již na počátku 20. století se někteří významní matematikové věnovali hledání optimálních postupů (strategií), tj. takových postupů, které maximalizují zisk, resp. minimalizují ztráty. Za první matematiky, kteří do této oblasti vstoupili, by asi byli považováni zejména Emile Borel a dále i Ernst Zermelo a Hugo Steinhause. Tím, kdo by však mohl být prohlášen za prvního teoretika na poli ekonomie a matematické teorie her je Antoine Augustin Cournot (1801–1877). Ve svém textu (1838) se zabýval jednorázovým soutěžením dvou firem na trhu a poskytl řešení v podobě rovnovážných strategií.

I v šachu by teoreticky bylo možné sestavit konečnou tabulku (matici) strategií pro oba hráče. Složitost hry však tkví v tom, že je těchto strategií příliš mnoho. Původem maďarský matematik působící od 30. let v Princetonu John von Neumann (1903–1957) se již ve 20. letech 20. století věnoval hledání optimálních strategií pro hry, které lze zaznamenat právě takovou tabulkou.

Za počátek aplikací teorie her v sociálních vědách bývá považována kniha von Neumanna a Oskara Morgensterna *Theory of Games and Economic Behavior (*1944), která shrnuje a doplňuje tehdejší výsledky v teorii her a upozorňuje na příbuznost analýz konfliktních situací v ekonomii a analýz strategických her. Touto obsáhlou prací se matematická teorie her prvně objevuje jako

samostatná disciplína aplikované matematiky. Podle von Neumanna a Morgensterna si neklade za cíl predikovat, jakým výsledkem skončí určitá situace, ale nabízí její analýzu s případnou možností nalezení optimální strategie. Přes mnohá zjednodušení, které jsem již zmínil, má teorie her, podle obou autorů, i ve své formální podobě mnoho společného se skutečným životem. Také v běžném životě zjišťujeme údaje o tom, co lze získat nebo ztratit, jaká jsou pravidla pro aktéry určitých situací a za jakých podmínek do nich můžeme vstupovat. Teorie her tak pomáhá pochopit termíny jako jsou *užitek* a *zájem*. Analýza možností při rozhodování, kde se berou do úvahy vlastní i spoluhráčovy tahy, umožňuje znázornit různé konfliktní a kooperativní situace. Významným prvkem, s nímž teorie her také pracuje, je rozhodování za určitého stupně informace, což je opět rys běžného každodenního rozhodování.

2.4. Základní pojmy teorie her

Jaké jsou obecné předpoklady teorie her :

1. Hráči jsou racionální.
2. Všichni účastníci hry znají pravidla a ta se v průběhu jedné hry nemění.
3. Hráči mají přehled o hodnotách ve hře a znají výši zisků a ztrát.

Když vyslovíme slovo *hra*, většinou nás asi napadne některá z deskových her. Ty jsou nám známy zejména jako posloupnost tahů, kdy má každý hráč určitou informaci o stavu hry v jejím průběhu. Existují samozřejmě i „*jednotahové*“ hry, ale i u nich nás může zajímat např. opakování, které umožní sledovat reakce protihráčů. Zkoumáním her v této podobě se zabývá d*ynamická teorie her.*

Pro objasnění termínu *statické teorie* je nutno zavést obecný model – *hru v normálním tvaru*. *Hrou v normálním tvaru* budeme rozumět trojici množin :

$\{\{1,2,..,n\},\{S_1,..,Sn\},\{Z_1,..,Zn\}\}$, kde

$\{1,2,..,n\}$ je množina hráčů,

$\{S_1,..,Sn\}$ je množina prostorů strategií a

$\{Z_1,..,Zn\}$ je množina výplatních funkcí hráčů

Jednotliví hráči jsou očíslováni přirozenými čísly, pro model je totiž podstatné hráče odlišit a znát jejich počet. Rozhodně platí, že máme alespoň dva hráče. Každému hráči *i* náleží *strategie* obsažené v příslušném S_i. Strategie je úplný popis jak odehrát hru a každému hráči poskytuje představu o návaznosti na kroky jeho spoluhráčů. Při hraní hry v normálním tvaru si každý hráč zvolí určitou strategii x_i a sada všech zvolených strategií (všech hráčů) dává příslušnou hodnotu výplatní funkce $Z_i(x_1,..,x_n)$ pro hráče *i* (výplatní funkce jsou definovány na kartézském součinu prostoru strategií $S_1(x_1..S_n)$. Všichni hráči znají strategie své i svých spoluhráčů a znají všechny hodnoty výplatních funkcí. Hra se hraje tak, že každý hráč si zvolí strategii nezávisle na ostatních a všechny

volby hráčů jsou zveřejněny současně. Příkladem může být známá hra *kámen-nůžky-papír* : Hru hrají dva hráči a oba mají stejný prostor strategií – mohou volit K (kámen), N (nůžky) nebo P (papír). Výplatní funkce lze zaznamenat do matice (tabulka č. 1) :

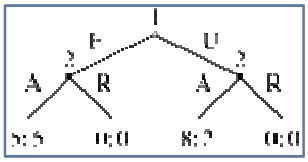
	2. Hráč			
1. Hráč		K	N	P
	K	0	1	-1
	N	-1	0	1
	P	1	-1	0

(Tabulka č. 1 - *Matice kámen, nůžky, papír*)

V tabulce jsou vypsány pouze hodnoty výplatní funkce 1. hráče, protože jde o *hru s nulovým součtem*, což v tomto případě znamená $Z_1(x,y) = -Z_2(x,y)$, kde x je volba 1. hráče, y volba 2. hráče. Ten, kdo vyhraje podle pravidel hry (tj. např. volí K a protihráč N), získá 1 (jednotek), prohraje-li, odevzdá 1 (jednotek) druhému hráči. Při remíze jsou výplaty nulové.

U her v *normální formě* se předpokládá, že hráči vybírají tahy zároveň, nebo alespoň nevědí, který tah vybral protihráč. Pokud hráči mohou znát tahy protihráče, mluvíme většinou o tzv. *extenzivní formě*.

Extenzivní forma hry bývá používána k formalizaci her, ve kterých hraje roli pořadí tahů. Hry jsou prezentovány jako stromy (obrázek č. 2). Každý uzel zde reprezentuje místo, ve kterém některý z hráčů vybírá tah. Hráč, který vybírá tah, je určen číslem napsaným v uzlu. Hrany reprezentují možné tahy hráče. Zisk pro jednotlivé hráče je specifikován v listu stromu.



(Obrázek č. 2 - *Extenzivní forma hry*)

Ve hře na obrázku jsou dva hráči. Hráč 1 vybírá první a má na výběr buď F nebo U. Hráč 2 vidí tah hráče 1 a poté vybere buďto A nebo R. Předpokládejme, že hráč 1 vybere variantu U a hráč 2 poté vybere A. Potom zisk prvního hráče je 8 a zisk druhého hráče je 2.

Extenzivní forma může zobrazit i situaci, kdy hráči vybírají tahy zároveň a také hry s neúplnou informací. Pokud hráč neví, v kterém z několika stavů se nachází, zakreslí se okolo těchto stavů kružnice.

Bez ohledu na konkrétní způsob reprezentace můžeme ještě rozlišovat hry podle následujících kritérií :

- **Počet hráčů -** Bývá zvykem předpokládat, že na hře participují alespoň dva účastníci. Obvykle se hovoří o hrách s konečným počtem hráčů.

- **Racionalita hráčů** - U účastníků hry lze rozlišovat dvě krajní pozice. Na jedné straně je tzv. „inteligentní“ hráč, který se chová v duchu racionality, jak jsme ji dříve zmínili, a na straně druhé je „hráč“, který své tahy volí náhodně, chová se jako náhodný mechanismus. (Pro racionálního hráče se obvykle používá právě termín *agent*, pro náhodný mechanismus se používá spíše termínů *příroda* nebo *prostředí.*)
- **Strategie -** Hry s *konečnými* a *nekonečnými strategiemi*. Hra *kámen-nůžky-papír* je hrou s konečnými strategiemi. Pokud by hráč volil např. reálné číslo z nějakého intervalu, šlo by o hru s nekonečnou strategií. Sem patří hry, kde se ve strategiích objevuje volba časového bodu („správné načasování“).
- **Výhra -** Hry s *konstantním* a *nekonstantním součtem*. Pro hry s konstantním součtem platí, že pro každou volbu strategií všech hráčů je součet výplatních funkcí všech hráčů konstantní.
- **Spolupráce** Hry *kooperativní* a *nekooperativní*. U nekooperativních her se předpokládá, že hráči nemohou vytvářet koalice ani si nějak doplňovat informace o hře domluvou. Může zde existovat překážka v komunikaci daná charakterem prostředí, v němž se hra odehrává, nebo to může být přímo zakázáno (předpis, zákon).
- **Počet tahů -** V tomto případě lze rozlišit hry *strategické*, které jsou reprezentovány např. ve zmíněném normálním tvaru a pomocí matice výplat, a hry *tahové*, jež bývají reprezentovány grafem ve tvaru stromu. Obě reprezentace lze kombinovat např. v závislosti na dostupnosti informace.

Hra *kámen-nůžky-papír* je konečná, strategická, nekooperativní hra dvou (racionálních) hráčů s konstantním součtem ($Z_1(x,y) + Z_2(x,y) = k$, kde $k = 0$). U her s konstantním součtem, kde $k \neq 0$, lze k považovat za jistý „vklad“ do hry, který však nemá pro teoretické úvahy nad těmito hrami žádný zvláštní význam a je možno využít analýzy her s nulovým součtem.

2.5. Optimální řešení

Na studium her se lze dívat ze dvou hledisek. *Normativní* hledisko se zabývá nejvýhodnějším jednáním v dané hře. *Deskriptivní* hledisko se zaměřuje na chování konkrétních hráčů. Je pochopitelné, že matematicky orientovaná teorie her se soustředí na hledisko normativní. Jejím zájmem je hledání odpovědí na výše uvedenou základní otázku teorie her. Do centra zájmu se tak dostává výplatní funkce. Ta dokáže nejen znázornit uspořádání (*preference*) výher a proher, ale navíc i ukazuje *o kolik* je něco výhodnější, resp. nevýhodnější. Několikrát jsme již řekli, že racionální hráči budou posuzovat zejména toto hledisko. Na následujícím příkladě velmi jednoduché jednotahové hry si ukážeme, jak budou postupovat racionální hráči při hledání pro ně nejlepšího řešení.

Mějme dva hráče. Každý z nich má dvě karty, 1. hráč má ♣5 a ♥2 a 2. hráč má ♣5 a ♥3. Na povel musí oba hráči ukázat jednu z karet. Dojde-li ke shodě v „barvě“, dostane 1. hráč (od druhého)

absolutní hodnotu z rozdílu hodnot ukázaných karet. Pokud se ukázané karty liší, tak ten, kdo měl vyšší hodnotu, dostane součet hodnot ukázaných karet. [2]

Jde opět o nekooperativní hru dvou hráčů s nulovým součtem. Hodnoty výplatní funkce jsou v následující tabulce (tabulka č. 2) :

	2. Hráč			
1. Hráč		♣5	♥3	
	♣5	0	8	(a)
	♥3	-7	1	(b)
		(c)	(d)	

(Tabulka č. 2 - *Nekooperativní hra dvou hráčů*)

Jak hrát tuto hru, abychom získali co nejvíce a současně se co nejméně poškodili při tahu protivníka? Z pozice 1. hráče je zcela nevýhodný spodní řádek (b). Při libovolné volbě 2. hráče je vyšší zisk na horním řádku (a) říkáme, že (b) je *dominován* řádkem (a). To znamená, že v pozici prvního hráče je třeba volit strategii ♣5. Pro 2. hráče je zcela nevýhodné volit sloupec (d), vždy přijde o 8 nebo 1 (jednotek). Sloupec (d) je dominován sloupcem (c). Druhý hráč bude volit strategii ♣5. Pro oba hráče jsme takto získali „nejlepší" volby strategií, aby v rámci pravidel hry a s ohledem na volby protihráče získali „co nejvíce" a ztratili „co nejméně". Těmito úvahami jsme našli *optimální strategie*, což znamená, že pokud se kterýkoli z hráčů od této své strategie odchýlí, může ztratit. Optimální strategie určily na matici výplat tzv. *sedlový bod*. Formálněji řečeno, sedlový bod v matici výplat určuje strategie x_0 (1. hráče) a y_0 (2.hráče) tak, že pokud si 1. hráč zvolí libovolnou strategii x, ale 2. hráč se drží optima, 1. hráč si nepolepší, tj. $Z_1(x,y_0) \leq Z_1(x_0,y_0)$. Obdobně to platí pro 2. hráče. Ve výplatních funkcích 1. hráče to znamená, že odchylka 2. hráče od strategie y_0 může přilepšit 1. hráči, $Z_1(x_0,y_0) \leq Z_1(x_0,y)$. O hodnotě výplatní funkce $Z_1(x_0,y_0)$ se hovoří jako o *ceně hry*. Optimální strategie x_0,y_0 jsou označovány termínem *Nashova rovnováha*, podle jména amerického matematika Johna Nashe.

V druhém příkladě jsem ukázal, jak lze snadno „odřezat" dominované strategie. Obecně však tento postup nemusí fungovat, protože sedlových bodů může být na matici výplat více, a dokonce tam nemusí být žádný. Když se podívám na matici výplat hry *kámen-nůžky-papír*, zjistím, že nemá žádný sedlový bod, a tedy ani Nashovu rovnováhu mezi svými strategiemi. Optimální řešení však existuje. Berme v úvahu, že hru *kámen-nůžky-papír* hrajeme mnohokrát za sebou. Kdybychom si z nějakého důvodu oblíbili např. strategii „kámen" (K), tak si toho náš (racionální) protihráč všimne a využije toho k vlastní výhře. Otázku, kterou si tedy každý racionální hráč položí, zní : „*S jakou pravděpodobností mám hrát své strategie, abych nepomohl soupeři k výhře tím, že pro něj bude výhodné hrát některou svou strategii?"* Zde je na první pohled jasné, že každá ze strategií této hry by měla být volena stejně často, resp. se stejnou pravděpodobností. Každý hráč volí strategii K s pravděpodobností ⅓ a se stejnou pravděpodobností i strategie ostatní (N a P). Teď již to nejsou *ryzí*

strategie (K, N, P), ale tzv. strategie *smíšené*, což jsou vektory pravděpodobností volby příslušné ryzí strategie.

Obecně, a opět trochu formálněji, má 1. hráč k dispozici množinu smíšených strategií S_1= {**x** = <P_1(K), P_1(N), P_1(P)> : kde P_1 (..) je pravděpodobnost volby příslušné strategie 1. hráčem a P_1(K) + P_1(N) + P_1(P) = } a 2. hráč S_2= {**y** = < P_2(K), P_2(N), P_2(P)> : kde P_2(..) je pravděpodobnost volby příslušné strategie 2. hráčem a P_2(K) + P_2(N) + P_2(P) = 1}.

U hry *kámen-nůžky-papír* jsou omezeny vektory **x** = <⅓,⅓,⅓> pro 1. hráče a **y** = <⅓,⅓,⅓> pro 2. hráče, protože taková volba pravděpodobností hraní ryzích strategií zaručí, že protihráč nemá k dispozici jednoznačně výhodnou volbu své ryzí strategie.

U her (dvou hráčů s konstantním součtem) platí věta, že *existuje alespoň jedna Nashova rovnováha* smíšených strategií a důkaz věty pochází od von Neumanna [1928]. Normativní pohled na teorii her je právě hledáním takových optimálních řešení. Již jsem zmínil, že termín *Nashova rovnováha* nese jméno matematika Johna Nashe. Jeho mimořádný příspěvek k teorii her pochází z roku 1951, kdy v článku *Non-Cooperative Games* zavedl pojem *rovnovážného bodu* a dokázal zobecnění uvedené věty pro všechny konečné nekooperativní hry.

Pro variantu nekonečné nekooperativní hry dvou hráčů není znám univerzální postup k nalezení optimálního řešení. Dokonce je dokázáno, že optimální řešení nemusí existovat. Dá se však najít pro některé typy výplatních funkcí.

2.6. Individuální a skupinová racionalita

Na začátku tohoto textu jsem ukazoval antagonistický konflikt dvou aktérů, tj. nekooperativní hru dvou hráčů s nulovým součtem. Zmínil jsme též, že hry s nulovým součtem lze použít pro analýzu her s konstantním součtem. Ve společenských vědách se však často setkáváme s jednáním, které nemusí mít charakter hry s konstantním součtem. Snad nejznámějším příkladem je tzv. „vězňovo dilema“. Jde asi o nejčastěji diskutovaný příklad v textech o teorii her.

Při vyšetřování závažného zločinu jsou uvězněni dva vážně podezřelí. Protože jsou důkazy neúplné a oběma lze prokázat jen např. krádež auta, které bylo ke spáchání zločinu použito, dostanou oba podezřelí (nezávisle) následující nabídku. Pokud se podezřelý přizná a tím umožní usvědčit svého komplice (tj. druhého podezřelého), bude mu trest významně snížen či dokonce prominut. Možnosti jsou uvedeny v tabulce (tabulka č. 3), kde dvojice <a,b> znamená počet let, která stráví ve vězení 1. podezřelý (a) a 2. podezřelý (b).

	2. Podezřelý		
1. Podezřelý		přiznání (P)	nepřiznání (N)
	přiznání (P)	<6,6>	<0,10>
	nepřiznání (N)	<10,0>	<1,1>

(Tabulka č. 3 - *Vězňovo dilema*)

Vězňovo dilema je nekooperativní hra dvou hráčů s nekonstantním součtem. Použiji-li znalosti z kapitoly **2.5 *Optimální řešení***, zjistím, že pro oba hráče je strategie N dominována strategií P a tato strategie tak tvoří Nashovu rovnováhu. I když by bylo pro oba mnohem výhodnější, kdyby se drželi strategie N. Bod 2,2 sice není rovnovážný, ale byl by jistě preferován každým účastníkem hry, jde o tzv. *paretovské optimum*. *Vězňovo dilema* je pěknou ilustrací rozdílu ve skupinové a individuální strategii.

Verze *vězňova dilematu* se stala již v 50. letech vzorovou hrou pro modelování individuální a skupinové racionality. Později se objevují různé varianty této hry i při studiu dlouhodobého (evolučního) „chování“ strategií v rámci populace.

3. Rozdělení projektu na etapy

Hlavním cílem projektu je vytvoření počítačové hry (fotbalového manažéra) s jistými prvky umělé inteligence. V návrhu projektu bude popsána kompletní realizace od inteligenčního jádra hry, přes hraní zápasů, plánování strategií, chodu celého týmu až po grafické uživatelské rozhranní. Projekt bude následně demonstrován na 2D simulaci zápasu mezi dvěma počítačem řízenými hráči. Jeho framework však umožňuje připojení v podstatě neomezeného množství hráčů jak člověkem, tak strojem řízených, kteří mezi sebou mohou vzájemně zápasit a vést svůj tým.

Umělá inteligence, která tvoří jádro projektu, je zastoupena na třech různých úrovních. *Manažerská (globální inteligence)* - řízení hry na úrovni manažera týmu, *asistentská (pomocná inteligence)* - rady asistenta týmu při tvorbě sestav, řízení financí a zázemí týmu a *hráčská* - inteligence na nejnižší úrovni, úrovni hráčů jako samostatně myslících strojů. Právě tato část projektu se mi jeví jako hlavním přínosem pro hry tohoto typu, kombinace globální inteligence s inteligencí samostatných hráčů.

Jakmile bude dokončeno herní jádro, nastane další fáze projektu, což bude grafika hry. Poslední fází bude nasazení hry do provozu a zvažování jejího uplatnění, případně další úpravy hry.

Projekt lze tedy rozdělit do tří fází :

- **Tvorba Modulu rozhodování (umělé inteligence)**
- **Tvorba Grafického modulu**
- **Nasazení hry do provozu a testování**

3.1. Modul rozhodování

Jak již bylo zmíněno, inteligence je ve hře zastoupena ve třech formách.

První, tou nejzákladnější, je inteligence *manažerská*, zastávající funkci na globální úrovni hry, úrovni hráčů (manažérů týmu). Manažéři řídí automaticky svůj tým, plánují zápasy, nastavují sestavy hráčů, formace, strategie, trénují hráče, starají se o finanční a přestupovou politiku týmu, zázemí, zaměstnance týmu, tedy řídí vše potřebné jako v reálném fotbalovém světě.

Ve hře bude rozlišeno několik obtížnostních úrovní :

Obtížnostní úrovně hry				
Úroveň	Easy	Normal	Hard	World Champion
%	60	80	100	140

(Tabulka č. 4 - *Obtížnostní úrovně hry*)

Popis parametrů :

- **Easy (Lehká)**

  Úroveň pro úplného začátečníka. Herní inteligence bude velmi zjednodušená. Nebude docházet v podstatě k žádné přestupové politice, tým bude hrát s velmi podobnou formací, nastavení strategie bude pouze týmové, trénink bude veden pouze na 60%, takže hráči nebudou plně vyvíjet své schopnosti.

- **Normal (Normální)**

  Úroveň pro běžného hráče. Ovládání týmu bude na solidní úrovni. AI bude řídit přestupy, trénink na 80%, formace nastavovat s využitím předchozího zápasu a nastavovat strategii pro klíčového hráče sestavy.

- **Hard (Těžká)**

  Úroveň pro zdatného manažera. Trénink hráčů bude probíhat na 100%, bude tedy využit maximální hráčův potenciál, při volbě formace bude docházet ke kalkulacím s využitím již odehraných zápasů a pomoci asistenta manažera, nastavování strategie bude individuální pro většinu hráčů s definováním jasného cíle týmu pro daný zápas. Přestupová politika bude probíhat po celý kalendářní rok, bude docházet ke skautingu a aktivnímu řízení financí aby klub dobře prosperoval a mohl nakupovat nové lepší hráče a rozšiřovat a zlepšovat zázemí týmu.

- **World Champion (Světová, velmi těžká)**

  Úroveň pro profesionála. Veškerá herní inteligence bude v plné síle a bude řízena v poměru 140% vůči normálnímu stavu. Bude tedy docházet k jakési nadlidskosti, predikcím a výpočtům, které běžný hráč nemá k dispozici. Budou například využity i skryté vlastnosti jednotlivých hráčů a bonusy jak finanční, tak v podobě zázemí týmu. Bude velmi těžké dlouhodobě vést svůj tým k úspěchům.

Druhou formou inteligence je výpomoc asistenta trenéra při volbě herní formace a strategie při přípravě k zápasů. Asistent zvolí nejlépe vyhovující formaci v závislosti na již odehraných zápasech a zápasech, které bude tým v nejbližší době hrát a nominuje hráče, kteří se budou do této formace svými dovednostmi nejlépe hodit. Manažer může využít i volbu nastavení strategie týmu i jednotlivým hráčům. Asistent se bude podílet i na řízení financí klubu a dohlížet na nákupy a prodeje hráčů, které bude hodnotit ze svého pohledu.

Velká většina současných fotbalových i jiných manažerů používá obě z předešlých forem inteligence ve svém herním jádře. Proto by mým přínosem k počítačovým hrám nebylo nic nového. Přemýšlel jsem tedy, čím se odlišit a čím posunout herní myšlení o kousek dál. Při zápasech ovlivňuje dění na hřišti formace a strategie jednotlivých hráčů. Jejich chování je však předem dané podle nějakého algoritmu, kterým se hra simuluje a již předem je interně znám výsledek. Ten mohou ovlivnit pouze manažeři svými zásahy do strategie týmu během hry nebo střídáním některých z hráčů.

Co kdyby i samotní hráči měli možnost v průběhu hry ovlivňovat dění na hřišti svou individualitou a svými schopnosti? Vystupovali by zde jako samostatně myslící bytosti (roboti) a tím by dávali hře dostatečnou rozmanitost a nepředvídatelnost. Jejich chování však nebude úplně nahodilé, bude ohraničeno určitými týmovými pravidly nebo předem definovanou strategií, ale jejich skutečný vliv pro tým bude velmi individuální a bude ovlivněn jejich atributy, formou, umístěním v sestavě, morálkou a samozřejmě určitým náhodným faktorem.V podstatě se dá říct, že hráč by se měl ve stejném zápase za úplně stejných podmínek chovat jinak, tak jak to chodí v reálném světě kolem nás.

3.2. Grafický modul

Grafika není hlavní součástí tohoto projektu. V první fázi bude kladen důraz na umělou inteligenci hry a grafika bude podána pouze formou textových výpisů na standardní výstup případně jednoduchý grafický 2D simulátor zápasů zobrazující dění hry.

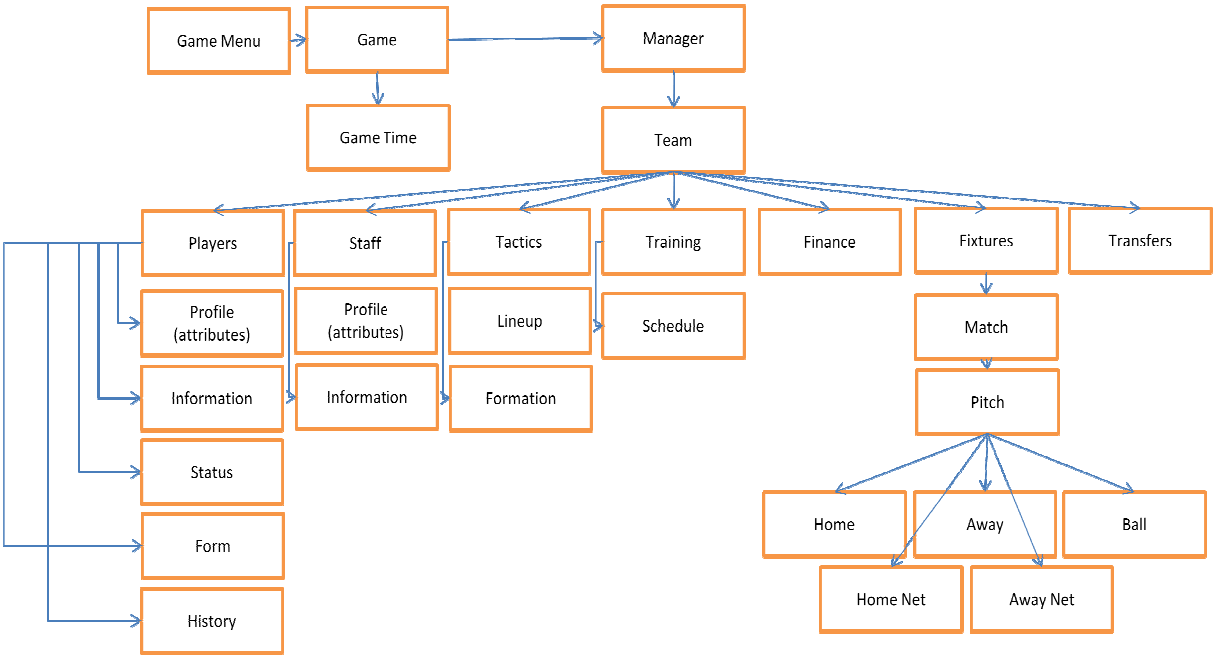
3.3. Nasazení hry do provozu

Hlavní náplní této fáze bude testování hry a návrh jejího nasazení a porovnávání úspěšnosti sestavených algoritmů v konfrontaci jiných. V současné době vidím dva způsoby využití hry :

- **Desktop aplikace**
  Hra by byla použita jako běžná pc hra s grafickým uživatelským rozhraní s možností hrát on-line případně ovládat více manažeru v rámci jednoho počítače.
- **Webová aplikace**
  Pro část webové aplikace by byla použita pouze první část projektu (umělá inteligence), na kterou by bylo navázáno webové rozhraní s datovým modelem. Byla by tak vytvořena on-line hra pro řadu fotbalových fandů, kteří by mohli sledovat dění svého týmu v 2D simulaci a ovládat svůj tým přes web.

4. Návrh projektu

Základní model celého projektu hry je zobrazen na obrázku. č 3. Jednotlivé buňky představují základní třídy a spojení jejich vzájemnou hierarchii. Tento model je zjednodušený, představuje pouze hlavní prvky hry.



(Obrázek č. 3 - *Projektový model*)

4.1. Modul Game Menu

Herní menu nebude v první fázi projektu implementováno. Půjde pouze o konzolovou aplikaci, která demonstruje hlavní náplň celého projektu a to simulaci funkčnosti umělé inteligence. Tento základ by mohl být nahrazen grafickým uživatelským rozhraním ve způsobu webové aplikace.

4.2. Modul Game

Hra je základní třídou, kterou vše začíná. Herních typů je hned několik. Hráč bude mít možnost volit mezi tzv. „Quick Game“ (Rychlou hru), kdy bude jako manažer ovládat již zavedený tým v polovině sezóny, hra bude situována pouze na jednu ligu. Databáze týmů a hráčů bude poměrně malá, bude to ideální typ hry pro nové hráče nebo pro demo hry. Druhým typem bude běžná „Dynasty Mode“ (Manažerská kariéra), kde hráč na začátku navolí zemi, za kterou bude hrát a poté zvolí tým dané země, který bude z pozice manažera řídit. Tento typ hry začíná běžnou předsezónní přípravou. Je potřeba nastavit veškeré náležitosti pro chod celého týmu od začátku. Posledním typem hry bude „Match Game“ (Zápasová hra), kdy sice budete vystupovat jako manažer týmu, ale nebudete moci tým jako celek řídit, jediné o co zde půjde je sezóna a zápasy, tedy formace, strategie a hraní zápasů, bez

jakéhokoliv ovládání zázemí týmu, zázemí bude řídit týmový asistent a to automaticky. Každý z typů hry bude jinak časově a zkušenostně náročný a proto očekávám, že každý hráč si najde zálibu v tom svém.

4.3. Modul Game Time

Hra bude postavena na principu simulace v reálném čase s možností zastavování herního času v důležitých okamžicích, které si vyžadují zásah nebo pozornost manažera týmu. Půjde tedy o typický kalendář, který bude představovat manažérův denní chléb a podle kterého se bude orientovat.

4.4. Modul Manager

Manažer týmu je hlavní postavou hry. Hráč, který hru hraje se musí do této role vcítit a chovat se, jako by jím byl sám. Manažerova kariéra je dlouhá. Může vystřídat několik týmů nebo být převážnou část své kariéry oddaný pouze jednomu klubu.

V počátku hry (manažerovy kariéry) je potřeba vybrat tým, který bude vést. Ve vedení mu napomáhají již předem daní zaměstnanci týmu jako asistent manažera, trenéři, lékaři, skauti a jiný personál nezbytný pro perfektní chod mužstva.

Způsob jakým manažer svůj tým vede závisí především na hráči, který hru hraje. Neméně významnou úlohu pro výsledek hry mají i manažerské vlastnosti, které se samozřejmě mohou v průběhu hry měnit a výrazně tak ovlivňovat celý tým. Těchto vlastností je celkem 10 a mohou nabývat hodnot od 0 do 99 (maximum). Čím vyšší je číslo dané vlastnosti, tím více je manažer v této schopnosti kvalitnější a tím lépe bude moci svůj tým vést. Seznam všech deseti vlastností je následující :

- **Coaching goalkeepers** - trénování brankářů
- **Coaching outfields** - trénování hráčů v poli
- **Coaching youngers** - trénování mladých hráčů
- **Tactics** - taktika
- **Management** - vedení týmu
- **Adaptability** - přizpůsobivost
- **Strictness** - přísnost
- **Motivation** - motivace
- **Scouting** - hledání talentů
- **Determination** - rozhodování

4.5. Modul Team

Tým je skupina lidí (hráčů a zaměstnanců týmu), kteří mají stejný zájem - vyhrávat. Všichni mají jednoho společného kapitána - manažera. Ten odpovídá za celkový chod týmu - trénink, taktiku, zápasy, finance, přestupní politiku a jiné další nezbytnosti. Cílem každého týmu je vyhrávat, hrát tu nejvyšší možnou soutěž a být první. Pokud dokáže manažer plnit tato či jiná očekávání týmového vedení, je vše v naprostém pořádku, jeli tomu jinak hledá se kámen úrazu a jak bývá zvykem, odnese to právě kapitán.

4.6. Modul Player

Každý tým se skládá především z hráčů, kteří jsou základními stavebními kameny. Každý hráč má v týmu určitou roli, kterou musí plnit a pro kterou je svými schopnostmi předurčen. Základní dělení hráčů v týmu podle pozic na hřišti je následující :

- **GK** – goalkeeper - brankář
- **DEF** – defender - obránce
- **MID** – midfielder - záložník
- **ATT** – attacker - útočník

Vedle pozice, na které je hráč schopen hrát je důležitá i strana :

- **C** – center - střed
- **L** – left - levá
- **R** – right - pravá

Hráčova schopnost zahrát dobře je dána nejen jeho umístěním na hřišti, ale také jeho schopnostmi (atributy). Hráči mají celkem 13 atributů, z nichž každý je důležitý na určitou pozici podle předem definovaných vah. Atributy mohou nabývat hodnot od 0 do 99 (99 je maximum, nejlepší hodnota) :

- **Acceleration** – zrychlení
- **Pace** – rychlost
- **Passing** – nahrávky
- **Shooting** – zakončování
- **Stamina** – výdrž hráče, kondice
- **Tackling** – schopnost odebírat soupeři míč

- **Dribbling** – vedení míče
- **Marking** – osobní obrana
- **Creativity** – tvořivost
- **Positioning** – schopnost se dobře postavit
- **Handling** – chytání míče
- **Anticipation** – předvídavost
- **Coordinating defence** – schopnost řídit obranu

4.7. Modul Staff

O hráče se musí někdo starat a připravovat jim zázemí. Existuje několik typů zaměstnanců a každý z nich má jinou funkci :

- **Assistant manager** (asistent manažera)
  Náplní asistenta je dělat černou práci za manažera týmu, pomáhat mu při určování taktiky a tvorbě sestavy při zápasech. Právě asistent trenéra a jeho schopnosti sestavit nejvhodnější formaci je jedna z forem umělé inteligence o níž bude psáno dále. Asistent je také klíčovým členem pro řízení financí týmu, stará se o sponzory, stadion a spokojenost fanoušků.
- **Coach** (trenér)
  Hlavní funkcí trenéra je trénink. Čím je trenér zdatnější, tím je jeho trénink efektivnější. Specialitou každého trenéra je způsob hry a formaci, kterou preferuje. Toto je schopen svým hráčům naordinovat s vysokou kvalitou a proto je při výběru trenéra důležité vybírat podle jeho zaměření. Pokud manažer týmu rád preferuje například styl 4-4-2A, je potřeba vybrat trenéra, který preferuje tuto formaci a zároveň útočnou hru. Jeho přínos pro hráče bude efektivnější.
- **Scout** (skaut)
  Skaut může být použit k několika účelům. Jeho primární funkcí je hledání vhodných talentů pro tým a pro pozdější nákup hráčů do týmu. Může však zároveň hodnotit i hráče vlastního týmu, jejich přístup k tréninku, potenciál růstu a tipovat případné nasazení do sestavy. Další velmi významnou funkcí skauta je sledovat následujícího soupeře a dávat o něm před zápasem zprávy.
- **Psycho** (lékař)
  Lékař se stará o zdraví všech členů. Je nezbytný pro rychlejší rekonvalescenci hráčů.

Všichni tito zaměstnanci mají své atributy stejně jako hráči (v rozsahu 0..99). Čím větší hodnotu vlastnosti má, tím lépe dokáže pracovat.

4.8. Modul Tactics

Taktika je klíčovou záležitostí hry každého týmu. Obvykle je důležitější než samotní jednotlivci. Pokud i průměrní hráči dokáži plnit trenérovy příkazy, leckdy bývají mnohem hůře porazitelní než skupina „frajírků“, kteří si dělají na hřišti co se jim zlíbí.

Základním taktickým prvkem je rozestavení hráčů na hřišti (formation). Každý manažer preferuje vlastní formaci, které nejvíce věří a dokáže ji svým hráčům nejvíce podat. Každá formace je jinak účinná na určitý typ hráčů a také jinak účinná v závislosti na soupeřově formaci. Proto zde vzniká poměrně závažný problém - jakou formaci na jaký zápas zvolit?

Manažeři mohou volit mezi těmito formacemi :

- **4-4-2** - normální formace
- **4-4-2A** - útočná formace
- **4-4-2D** - obranná formace
- **4-2-4** - extrémně útočná formace
- **4-3-3** - velmi útočná formace
- **3-5-2** - vyvážená formace
- **3-4-3** - útočná formace
- **5-4-1** - velmi obranná formace
- **5-3-2** - obranná formace
- **5-3-2A** - normální formace
- **5-3-2D** - velmi obranná formace

Formace jsou zapsány ve tvaru, kde první číslo značí počet obránců, druhé počet záložníků a třetí počet útočníků. Brankář je do formace počítán automaticky. Příznak A znamená útočnou a D defenzivní strategii.

Jakmile každý manažer zvolí formaci pro zápas v závislosti na předchozích znalostech a zkušenostech, musí nastavit týmovou a hráčskou taktiku a současně umístit jednotlivé hráče do sestavy.

Sestava se skládá z 11 hráčů (1 brankář a 10 hráčů) a několika náhradníků. Počet náhradníků závisí na typu zápasu, který se hraje. Přátelské zápasy mají neomezený počet náhradníku a neomezený počet střídání, na zápasy ligové a mezistátní může být nominováno maximálně 5 náhradníků a z toho jen 3 mohou být v zápase použiti.

Týmová strategie ovlivňuje chování celého týmu a měli by se podle ní řídit všichni hráči na hřišti. Strategie může být nastavena podle parametrů (tabulka č. 5).

Team instruction						
Passing	Mixed	Short	Direct	Long		
Tackling	Easy	Normal	Hard			
Mentality	Ultra Defensive	Defensive	Normal	Attacking	Gung Ho	Time Wasting
Closing Down	Stand Off	Own Half Only	Always			
Counter Attack	Yes	No				
Men Behind Ball	Yes	No				

(Tabulka č.5 - *Týmové instrukce*)

Popis parametrů :

- **Passing** (Nahrávky)

  Parametr určuje převládající způsob nahrávek, o který se budou hráči snažit.

  Mixed - smíšené, Short - krátké, Direct - střední, Long - dlouhé
- **Tackling** (Obírání míče)

  Parametr určuje způsob obírání míče, o který se budou hráči snažit. Easy - lehké, Normal - normální, Hard - neustálé
- **Mentality** (Smýšlení)

  Parametr určuje vyváženost týmu v souvislosti s útočnou a obrannou fází. Ultra Defensive - velmi defenzivní, Defensive - defenzivní, Normal - normální, Attacking - útočná, Gung Ho - velmi útočná, Time Wasting - zdržování hry
- **Closing Down** (Tlak na míč)

  Parametr určuje způsob tlaku na míč. Stand Off - žádný tlak na míč, Own Half Only - pouze na vlastní polovině, Always - po celém hřišti
- **Counter Attack** (Protiútoky)

  Parametr určuje zda bude tým používat protiútoky.
- **Men Behind Ball** (Muži za míčem)

  Parametr určuje zda bude tým bránit všemi hráči za míčem.

Jednotliví hráči se mohou chovat podle týmové strategie nebo podle jejich individuální strategie. Individuální strategie předčí týmovou a hráč se tedy může svým chováním na hřišti poměrně výrazně lišit. Při této volbě je potřeba dávat poměrně velký pozor, aby nedocházelo k nepředvídatelným událostem a k netýmovému chování hráčů na hřišti. Hráčské instrukce ukazuje tabulka č.6 :

Player instruction							
Passing	Team	Mixed	Short	Direct	Long		
Tackling	Team	Easy	Normal	Hard			
Mentality	Team	Ultra Defensive	Defensive	Normal	Attacking	Gung Ho	Time Wasting
Closing Down	Team	Stand Off	Own Half Only	Always			
Marking	Zonal	Man					

(Tabulka č.6 - *Hráčské instrukce*)

Popis parametrů :

- **Passing** (Nahrávky)
  Parametr určuje převládající způsob nahrávek, o který se bude hráč snažit.
  Team - týmová (default), Mixed - smíšené, Short - krátké, Direct - střední, Long - dlouhé
- **Tackling** (Obírání míče)
  Parametr určuje způsob obírání míče o který se bude hráč snažit.
  Team - týmová (default), Easy - lehké, Normal - normální, Hard - neustálé
- **Mentality** (Smýšlení)
  Parametr určuje vyváženost hráče v souvislosti s útočnou a obranou fází.
  Team - týmová (default), Ultra Defensive - velmi defenzivní, Defensive - defenzivní,
  Normal - normální, Attacking - útočná, Gung Ho - velmi útočná, Time Wasting - zdržování hry
- **Closing Down** (Tlak na míč)
  Parametr určuje způsob tlaku na míč. Team - týmová (default), Stand Off - žádný tlak na míč,
  Own Half Only - pouze na vlastní polovině, Always - po celém hřišti
- **Marking** (Osobní obrana)
  Parametr určuje jakou obranu bude hráč používat.
  Zonal - zónová obrana, Man - osobní obrana

4.9. Modul Training

Trénink je nezbytnou a často i nepříjemnou součásti kvalitního výkonu. Pokud tým kvalitně trénuje, je předpoklad, že bude hrát dobře, pokud netrénuje hrát dobře určitě nebude. Bez tréninku prostě nelze žádný sport na žádné úrovni kvalitně vykonávat. A proto i ve fotbalovém manažeru má své opodstatněné místo.

Nastavení způsobu tréninku bude velmi různorodé a bude záležet na manažerovi, jak tvrdý trénink svým svěřencům naordinuje. Uživatel bude mít možnost nastavit svým hráčům celý tréninkový plán. Celý den bude rozdělen po hodinách do oken a každý trenér může navolit svému týmu kompletní denní program. Tímto způsobem pak může sestavit plán na celý týden a to z celkem 7 typů kondičních sad.

- **Fitness** (kondiční) - trénink kondice hráče
- **Aerobic** - trénink pohybu hráče
- **Sprint work** - trénink rychlosti
- **Weight training** - trénink síly
- **Base run** - trénink vytrvalosti
- **Core stability** - trénink výdrže
- **Tactical** (taktické) - trénink taktiky
- **Changing play** - trénink změny tempa hry
- **Counter Attack** - trénink protiútoků
- **Defending phase** - trénink obranné hry
- **Attacking phase** - trénink útočné hry
- **Shadow play** - trénink stínování protihráčů
- **Tackling** (obranné) - trénink odebírání míče
- **Pressuring** - trénink tlaku na míč
- **Defending 1v1 2v2 3v3** - trénink obranné hry
- **Defending phase** - trénink obranné fáze hry
- **Passing** (nahrávky) - trénink nahrávek
- **Passing drills** - trénink nahrávek
- **Crossing drills** - trénink centrů
- **Keeping ball** - trénink chytání (pouze pro brankáře)
- **Crossing praktice** - trénink centrů do vápna
- **Shooting** (střelba) - trénink střelby
- **Shooting drills** - trénink střelby
- **Finnishing drills** - trénink zakončování
- **Attacking 3v2** - trénink útočné fáze hry
- **Set-pieces** (standardní situace) - trénink přímých kopů na bránu
- **Defensive set play** - trénink defenzivních přímých kopů na bránu
- **Attacking set play** - trénink útočných přímých kopů na bránu
- **Rest** (odpočinek) - odpočinek, volný program

Každá z těchto sad obsahuje speciální tréninkové metody, kde každá ovlivňuje (pozitivně i negativně) určité atributy hráče. Konkrétní ovlivnění atributů není v tuto chvíli přesně definováno, nebude však jednoduché nastavit vše tak, aby zlepšování (zhoršování) bylo přiměřené a reálné v časovém měřítku (hráč by se neměl za jednu sezónu příliš zlepšit nebo zhoršit).

4.10. Modul Finance

Co by to bylo za řízení klubu, kdyby manažer nemohl ovlivňovat ekonomiku týmu. Finance týmu budou ovlivňovat dva faktory, příjmy a výdaje. Mezi příjmy týmu můžeme počítat příjem z prodaných vstupenek, televizních přenosů, obchodování týmu, odměn týmu, prodeje hráčů a sponzorských darů. Výdaji mohou být nákupy hráčů, platy hráčů a zaměstnanců týmu, poplatky, údržba stadionu, pokuty nebo úroky z půjček. Rozdíl mezi výdaji a příjmy dělá zisk týmu, který se bude každým rokem zúčtovávat. Na začátku každé sezóny bude týmu nastaven roční „budget" (rozpočet), se kterým musí vystačit na nákupy hráčů. Pokud tým během sezóny vydělá, bude rozpočet další sezónu vyšší.

Dalším důležitým bodem, který je s financemi spojený je zázemí a stadionu týmu. Každý z týmů má na začátku zázemí a stadion (kapacitu) na jiné úrovni a je pouze na prosperitě manažera, jak s tím dokáže naložit.

4.11. Modul Match

Okamžikem pravdy celého týmu bývá zápas. Zde tým ukazuje co natrénoval, jak jsou na tom jeho hráči po fyzické, psychické i technické stránce. Třída „Match" bude simulovat zápas mezi dvěma týmy (domácí a hosté). Bude uchovávat veškeré technické údaje o zápase, góly, střely na bránu, rohy, auty, fauly, žluté karty, červené karty, výpočet držení míče, statistiky o pozicích míče v určitých oblastech hřiště, statistiky nahrávek, střel, prostě vše co k fotbalu patří. Tyto údaje budou spravovány samotnou třídou „Match" nebo pomocnými třídami. Jednou z hlavních pomocných tříd je třída „Pitch" (Hřiště).

4.12. Modul Pitch

Hřiště je místem, kde jsou umístění hráči obou týmu, rozhodčí a míč. Jejich vzájemný pohyb, umístění a chování bude právě třída Pitch obstarávat. Bude zároveň základním zdrojem pro 2D simulaci a vstup do grafické části hry.

Jsou v zásadě dva způsoby, jak ukládat pozice objektů na hřišti. Prvním, jednodušším způsobem, je rozdělit hřiště do dvourozměrného systému dlaždic (dlaždicová mapa se používá v řadě strategických her), kde každý objekt bude umístěn na jednu dlaždici protože dlaždice reprezentuje minimální prostor objektu.

Druhým, poměrně komplikovanějším je absolutní umístění objektů do prostoru o předem definovaném rozměru. Zde však vyvstává problém kolize objektů, je nutné předem určit jaký daný objekt zabírá prostor, aby nedocházelo k překrývání objektů.

Oba způsoby jsou následně mapovány na obrazovku o určitém rozměru pixelů. V tuto chvíli bych chtěl upozornit na to, že nejsem pevně rozhodnut, na jakém principu bude hřiště pracovat a jak

bude rozděleno. Předpokládám však, že zvolím první variantu vzhledem k její jednodušší implementaci.

5. Implementace modelu

Celá hra je implementována v jazyce C++. C++ jsem zvolil hned z několika důvodů. Prvním a v podstatě nejdůležitějším požadavkem, byla rychlost. Z předchozích zkušeností z programování počítačových her se mi neosvědčil jazyk Java, ve kterém bylo matematické i grafické zpracování při větším počtu objektů ve scéně příliš pomalé, především z důvodu interpretace předkompilovaného kódu jazyka. Dalším požadavkem byla multiplatformnost, která je v moderních aplikacích příjemnou podmínkou, což jazyk C++ splňuje a poměrně důležitou podmínkou byl objektově orientovaný přístup k programování, na kterém je celý model hry postaven.

5.1. Modul rozhodování

5.1.1. Inteligence Asistentská

Druhá forma inteligence, tedy inteligence *asistentská*, se mi v projektu podařila naimplementovat v podstatě kompletním způsobem. Hlavním rozhodovacím problémem byla přesná volba typu inteligence, tedy specifikace, zda jde o řešení hry s nulovým či nenulovým součtem. Z předchozího teoretického úvodu plyne, že hry s nulovým součtem mají ve výsledné matici zisků součet všech hodnot roven nule. Jde tedy o jakési rozdělování pomyslného koláče, kdy každý hráč svou volbou ukrojí část koláče pro sebe. U her s nenulovým součtem se předpokládá že hodnoty v matici zisků jsou neporovnatelné a jsou tedy buďto absolutními nebo relativními zisky.

Inteligence *asistentská* je typem hry s nenulovým součtem o dvou hráčích. V prvé řadě je potřeba pro oba hráče vytvořit *tabulku zisků*. Ta je v projektu naimplementována jako dvourozměrné pole struktur s cenovými hodnotami pro oba hráče. Velikost pole závisí na počtu formací, mezi kterými mohou manažéři volit. Vznikne tak tabulka podobná tabulce č. 7.

Formace	4-4-2	4-4-2A	4-4-2 D	…
4-4-2	77/87	77/67	77/45	…
4-4-2A	98/87	…	…	…
4-4-2 D	78/87	…	…	…
4-2-4	…	…	…	…
…	…	…	…	…

(Tabulka č. 7 - *Tabulka zisků*)

Kde se však jednotlivé hodnoty tabulky berou a jak se počítají? Klíčovou úlohu zde hraje funkce nastavující jednotlivým hráčům hodnoty všech pozic, na kterých mohou na hřišti hrát, podle jejich atributů. Důležitá je zde také pozice hráče, kterou má předurčenu. Pokud je počítána hodnota pozice, na které hráč neumí hrát, je jeho přínos pro tým na této pozici zvolen na 0, tedy

pravděpodobně nebude na toto místo vůbec volen. Následující příklad uvádí způsob výpočtu pro pozici ATT_CL (levý střední útočník).

ATT_CL = (acceleration * 0.2) + (pace * 0.3) + (shooting * 0.3) + (stamina * 0.1) + (dribbling * 0.1))

Z příkladu je patrné, že výsledná hodnota se počítá jako součet poměru atributů, které jsou pro pozici vhodné. Jiné atributy a jiným procentuálním vyjádřením jsou vhodné na pozici útočníka a jiné zase na pozici brankáře, ostatně jak tomu v reálném fotbalu bývá. Vztahy pro výpočet jednotlivých pozic uvádí tabulka č. 8.

Player attributes (0-99)	Attackers	Midfielders	Defenders	Goalkeepers
Acceleration	20%	15%	15%	15%
Pace	30%	15%	15%	
Passing		25%		
Shooting	30%			
Stamina	10%	20%	5%	5%
Tackling			25%	
Dribbling	10%			
Marking			25%	
Creativity		25%		
Positioning			15%	20%
Handling				30%
Anticipation				20%
Coord. Defence				10%
	100%	100%	100%	100%

(Tabulka č.8 - *Užitek atributů pro pozice na hřišti*)

Dalším krokem je vybrání hráče, který se na volenou pozici svými užitky nejvíce hodí. Tento hráč je umístěn do pomyslné sestavy. Tímto způsobem jsou obsaženy všechny pozice formace a výsledný užitek celé formace je dán součtem všech dílčích užitků hráčů ve formaci dělený počtem hráčů formace.

užitek formace = suma(užitek hráčů) / suma(počet hráčů)

Uvedeným způsobem jsou vypočítány hodnoty pro oba hráče a všechny potencionální pozice a následně vloženy do tabulky zisků.

```
// Naplnění tabulky strategií
SStrategy costTable[FORMATION_COUNT][FORMATION_COUNT]
Nastav hodnoty hračů na jednotlivé pozice
for ( pro všechny strategie ) {
  Cena[all] = 0
  for ( pro všechny hráče ) {
    Vyber pozici dané strategie
    Nalezni nejlepšího hráče pro danou pozici
    Cena[i] = Cena[i] + CenaHrace(pozice)
  }
  Cena = Cena/PocetHracu
  Vložení ceny do tabulky zisků
}
```

Jakým způsobem vyberu ze získané tabulky zisků tu nejvhodnější strategii pro oba hráče? Zde vycházím z teoretického základu, popsaného v druhé kapitole, který ukazuje, že nalezení nejvhodnější strategie pro oba hráče se shoduje s nalezením Nashova ekvilibria. Nashův algoritmus hledá tzv. *best response* strategie pro oba hráče. Pokud je nalezena pouze jedna strategie, jde o *ryzí strategii*, kterou oba hráči volí v jakémkoliv případě, pokud je *best response* strategií více, jedná se o ekvilibrium ve *smíšených strategiích* (viz. výše).

```
// Nalezení optimální strategie
Strategy[all] = -1
StrategyMax[all] = -MAXINT
for ( pro všechny strategie ) {
  for ( pro všechny hráče ) {
    if (table[i][i] > StrategyMax[j]) {
      StrategyMax[j] = table[i][i]
      Strategy[i] = i
    }
  }
}

// Smíšené strategie
if( je vice nez jedna strategie stejne hodnoty ) {
  Výběr strategie ve smíšených strategiích
}
```

Díky výše zmíněným algoritmům jsou získány nejen strategie pro oba hráče, ale zároveň i hráči, kteří se nejlépe hodí na určité posty. Manažer tak má zvolenu tu nejlepší možnou sestavu, kterou by mohl sestavit.

Je pravdou, že v reálném světě je volba naprosto ideální sestavy v podstatě nemožná. Nelze totiž objektivně matematicky popsat vlastnosti všech hráčů, a to především jejich aktuální formy, mentální připravenosti, zdraví, motivace na každý zápas a jiných těžko popsatelných individuálních atributů.

5.1.2. Inteligence Hráčská

Posledním typem inteligence je inteligence *hráčská*. Musím říct, že tato část projektu mě stála pravděpodobně největší úsilí a také mi vzala nejvíce času. Chovaní jednotlivých hráčů není

jednoduché popsat. Je opravdu nespočet možností jejich chování a to i v omezeném prostoru s omezenými možnostmi.

Jádrem simulace hry je třída *Match* (zápas). Tato třída obsahuje hlavní algoritmus (*Match::Play()*), který vytvoří hřiště pro simulaci utkání, hřiště inicializuje (*Pitch::Init()*) a v opakujících se cyklech provadí update hřište (*Pitch::Update()*) a vykreslení jeho objektů na obrazovku (*Pitch::DrawScene()*).

Hlavní algoritmus simaulace zápasu *Match::Play()*

```
// Funkce spousti simuluje deni na hristi
void Match::Play() {
  InitTime();
  pitch->Init();
  pitch->DrawScene(time);

  // Herni cyklus
  bool done = false;
  while (done == false) {
    SDL_Event event;
    // Rutina pro uzavreni okna
    while( SDL_PollEvent(&event) ) {
      if( event.type == SDL_QUIT ) done = true;
    }

    // Update hriste
    pitch->Update();
    // Vykresleni sceny
    pitch->DrawScene(time);
    // Update casu hry
    UpdateTime();
    // Uspani
    Sleep(SLEEP);
  }
}
```

Inicializace hřiště *Pitch::Init()*

```
// Inicializace hriste
void Pitch::Init() {
  // Vymazani hriste
  ClearField();
  // Vymazani tymu
  ClearTeams();
  // Pocatecni pozice balonu
  InitBall();
  // Pocatecni rozestaveni tymu HOME
  InitHome();
  // Pocatecni rozestaveni tymu AWAY
  InitAway();
  // Inicializace grafiky
  InitGraphics();
}
```

Update hřište Pitch::Update()

```
// Funkce provadi update deni na hristi
void Pitch::Update() {
  UpdateBall();
  if (TestGoal()) return;
  UpdateTeam(home);
  UpdateTeam(away);
}
```

Vykreslování na obrazovku *Pitch::DrawScene()*

```
// Hlavni funkce pro vykreslovani sceny na obrazovku
void Pitch::DrawScene(int time) {
  // Zamknuti obrazovky
  graphics.Lock(graphics.GetScreen());
  // Smazani obrazovky
  graphics.ClearScreen(graphics.GetScreen());
  // Vykresleni hriste
  DrawField(graphics.GetScreen());
  // Vykresleni skore
  DrawScore(graphics.GetScreen());
  // Vykresleni casu
  DrawTime(graphics.GetScreen(), time);
  // Vykresleni objektu
  DrawObjects(graphics.GetScreen());
  // Odemknuti obrazovky
  graphics.Unlock(graphics.GetScreen());
  // Buffer na obrazovku
  SDL_Flip(graphics.GetScreen());
}
```

Chování hráčů je popsáno algoritmy ve třídě *Pitch*. Tato třída se stará především o veškeré dění na hřišti, od inicializace hřiště, přes popis a chování objektů, až po samotné vykreslení dění na obrazovku, pro něž je právě tato třída zdrojem dat. Chování hráčů, které jsem navrhl, lze jednoduše popsat formou stromové struktury (obrázek č. X), podle níž se hrači ve svých stavech rozhodují (chovají).

Vstupním bodem struktury je tým a jeho vlastnosti, které jsou simulací voleb manažéra týmu. Pokud manažer preferuje pro tento zápas například častou střelbu na branů nebo dává přednost vedení útoků spíše po křídlech než středem hřiště, hráči se budou rozhodovat podle jeho (tedy týmových) příkazů.

V kořeni rozhodovacího stromu je větvení dle držení míče. Je jasné, že v jednom herním cyklu budou využity obě větve stromu. Tým držící míč bude obsloužen funkcí *WithBall()*, tým bez míče funkcí *WithoutBall()*. Ve funkce *WithoutBall()* se vybere hráč nejblíže míči a funkci *GoToTheBall()* obstará jeho posun. Ostatní hráči týmu jsou předáni funkci *UpdatePlayer()*, která se postará o update pozice každého hráče dle jeho individuálního nastavení. Rozhodnutí, že pouze první hráč bude následovat míč je zde pouze demonstrativní a implementačně jednodušší, optimální by bylo rozhodování opravdu každého hráče individuálně, podle jeho aktuálního stavu na hřišti a jeho předurčené strategie. Každý hráč by se tak, dle svého a týmového rozestavení, sám rozhodl, zda je pro něj v tuto chvíli výhodné bojovat o míč nebo zaujmout jinou ze strategií. Nespornou výhodou tohoto (jednoduššího) způsobu je fakt, že nemůže docházet k patové situaci, kdy míč je v místě hřište, kde o něj neprojeví zájem žádný z hostujícího nebo domácího týmu, protože vždy minimálně jeden hráč z každého týmu se bude snažit míč získat.

Daleko komplikovanější je funkce *WithBall()*. V této funkci hraje klíčovou roli hráč s míčem. Jeho rozhodování obsluhuje funkce *DoWithTheBall()*, která rozhoduje v první řadě na základě hráčovi individuální strategie nebo týmové (defaultní) strategie. Pokud má hráč manažerem předordinovánu individuální strategii, podle které se má na hřišti chovat, je jeho volba poměrně jasná. Většina hráčů se

však obvykle chová týmově. Jejich rozhodování je pak určeno tím, co jim jejich manažer zvolí. Na obrázku č. X je ukázána jedna z možností rozhodnutí dle chování hráčů s míčem. Strategie *SHOOTING* předpokládá, že tým se bude snažit střílet na bránu z každě možné pozice, strategie *PASSING* určuje týmu snahu držet míč co nejdéle na svých kopačkách a *MOVING* značí snahu hráčů držet míč co nejvíce u nohy a předejít tak stracení míče špatnou nahrávkou. Implicitní strategií každého týmu je strategie *STANDING*, kdy všichni hráči si drží své pozice a kombinují případně střílí pouze v nezbytných případech. Je to typ opravdu konzervativní strategie. Každá z těchto strategií provádí v určitém sledu několich předdefinovaných rutin :

- **TryShoot** (Threshold)

  Funkce simuluje střelbu na bránu, podle prahu Threshold vypočte, zda je pro hráče efektivní vystřelit z pozice, ve které se nachází. Pokud je střelba prahem umožněna, vypočte ideální místo pro umístění střely vzhledem k poloze střílejícího hráče, brankáře soupeřova mužstva a poloze vlastních i soupeřových hráčů. Důležité je i nastavení rychlosti střely, která může významě ovlivnit, zda bude míč brankářem chycen, či nikoliv. Rychlost je vypočtena podle následujícího vztahu :

  (Player->Shooting * max(100-Distance,1))/100

  Ze vztrahu vyplývá, že vedle atributu hráčovy schopnosti střílet je důležitá i jeho poloha na hřišti, která výrazně snižuje rychlost střelby. Opět by se i zde dalo uvažovat o možném vylepšení a to především z fyzikálního hlediska, kdy rychlost míče by nebyla uvažována jako konstantní, ale klesající v závislosti na tření a gravitaci, které rychlost výrazně ovlivňují. Mohl by zde potom vzniknout faktor, který způsobí, že střela na bránu ani nedoletí. Z důvodů jednodušší implementace jsem však zvolil variantu s konstantní rýchlostí střely.
- ***TryMove()***

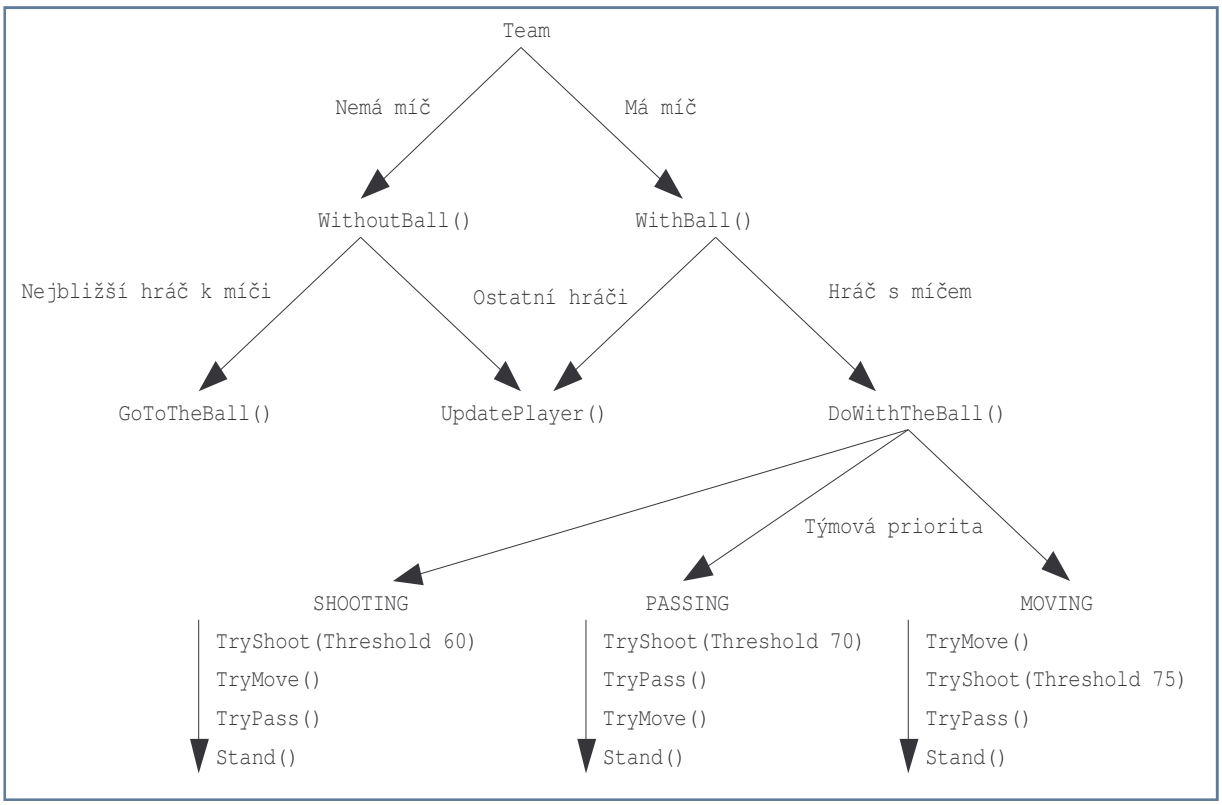
  Funkce simuluje pohyb hráče s míčem k soupeřově bráně.
- ***TryPass()***

  Funkce simuluje nahrávku míče spoluhráčovi. V této funkci je důležitá především volba správného spoluhráče, kterému je nahrávka adresována. Algoritmus počitá i s možností, že hráč může být obsazen protihráčem (v jeho blízkém okolí se vyskytuje protihráč). Tento hráč je pak vyřazen ze seznamu potencionálních adeptů pro přihrávku, aby nedocházelo ke zbytečně častým ztrátám míče. Kromě podmínky volnosti je při volbě důležitá pozice a vzdálenost hráče, hráč který je příliž blízko nebo příliž daleko nebude pravděpodobně zvolen, stejnětak hráč který je daleko od soupeřovi brány. Při volbě jsou tedy upřednostněny tyto možnosti s následujícími vahami :

  TODO dodělat váhy
- **Stand()**

  Funkce simulující nicnedělání hráče.

Každá z těchto rutin je typu bool, tedy pokud neuspěje, testuje se další v pořadí. Poslední rutina *Stand()* je provedena v nejhorším případě kdykoliv.



(Obrázek č. 5 - *Rozhodovací strom hráče na hřišti*)

5.1.3. Inteligence Manažerská

Manažerská forma inteligence není v tuto chvíli implementována z důvodu nižší priority této části projektu. Implementace se zaměřuje především na část asistentskou a hráčskou, která tvoří jádro hry. V poslední fázi projektu, bude současně s herním menu implementována inteligence manažérská.

5.1.4. Algoritmus hledání cesty, A* (A star)

Hledání cesty lze omezit na zodpovězení otázky : *„Jak se dostanu z bodu A do bodu B?“*. Obecně platí, že cesta z jednoho místa na jiné může mít několik různých řešení, v ideálním případě však požadujeme řešení naplňující následující cíle :

Jak se dostat z bodu A do bodu B.

Jak se vyhnout překážkám na cestě.

Jak najít nejkratší možnou cestu.

Jak příslušnou cestu najít rychle.

Některé algoritmy hledání cesty neřeší žádné z uvedených problémů, jiné zase řeší všechny. Zvolil jsem algoritmus A* jehož hlavní výhodou je poměrně lehká implementace a v celku dobrá výpočetní rychlost.

Abych mohl použít algoritmus A* , musel jsem interpretovat prostředí, po kterém se objekty pohybují jako graf, jinými slovy rozdělit jej na hrany a uzly a určit co představují. Uzly představují dlaždice a hrany jsou přechody mezi nimi. Výsledná cesta se pak počítá podle vzorce $\boldsymbol{F = G + H}$, kde G je cena ze současného místa do místa vedlejšího a H je heuristika, tedy předpokládaná cena do cíle, více na internetové stránce [3].

Jak již jsem v návrhu hry zmínil, hřiště *Pitch* je implementováno jako dlaždicová mapa o rozměrech 210x110 dlaždic. Hrací plocha hřiště představuje rozměr 200x100, po obvodu této plochy je 5 dlaždic, které reprezentují aut. Díky tomuto návrhu je i okolí hřiště hráči dostupné a hra tak vypadá realističtěji. Každá dlaždice je jeden objekt *AStarNode*, obsahující informace o sobě a svém okolí. Důležité jsou především údaje o poloze dlaždice, její ceně a seznamu jejich sousedů.

Algoritmus A* je v projektu implementován funkcí *FindPath* ve třídě *AStarSearch*. Funkce nalezne cestu ze startovního uzlu A do cílového uzlu B. Oproti originálnímu algoritmu A*, jsem si své hledání cesty lehce upravil, podle vlastních výpočetních a výsledkových nároků. Modifikace spočívá ve vložení dvou parametrů *depth* a *checkAccessible*.

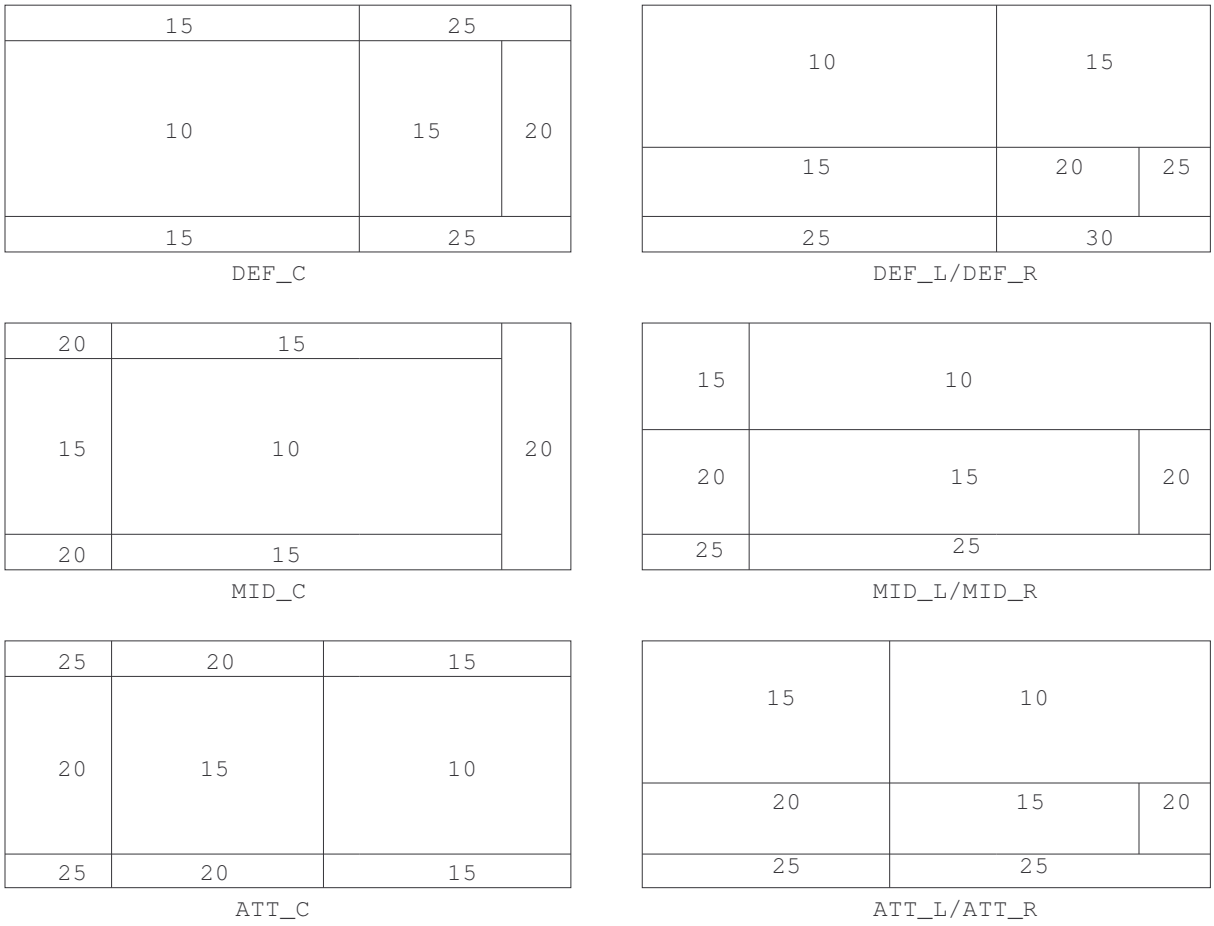
První z nich je kontrola zanoření *depth*, který udává počet cyklů algoritmu A*. Hlavní přínos je v tom, že algoritmus nehledá cestu až do konce, ale naznačí pouze omezený počet prvních kroků, kterými se hráč vydá. Výpočet celé cesty by byl poměrně časově náročný a hráč ji pravděpodobně v dalším herním cyklu bude přehodnocovat podle nových podmínek na hřišti. Tato časová úspora se navíc zvýrazní i faktem, že objektů, pro které se je tento algoritmus klíčový při výpočtu cesty, je v každém herním kroku celá řada a proto zavedení parametru *depth* je výrazným rychlostním zlepšením.

Algoritmus A* je ve hře používán nejen pro výpočet cesty hráčů, ale také pro určení směru míče *Ball*, který představuje samotný objekt hřiště. Při výpočtu cesty míče, na rozdíl od hráčů, nebereme v úvahu překážky, které se mohou v cestě objektu vyskytnou. Proto jsem zavedl druhý parametr *checkAccessible*, který indikuje zda algoritmus nalezení cesty *FindPath* má, či nemá brát překážky v zřetel. Jeden, z mého pohledu, nejdůležitějších algoritmů hry popisuje následující úryvek kódu :

```
// Funkce vyhleda optimalni cestu ze startovniho "startNode" do ciloveho "goalNode"
// bodu
// double "depth" Maximalni hlouba zanoreni, pocet iteraci cyklu
// bool "checkAccessible" Kontrola dostupnosti okolnich bodu
// return Cestu (seznam bodu)
list<AStarNode> AStarSearch::FindPath(AStarNode startNode, AStarNode goalNode,
                                      double depth, bool checkAccessible) {
  list<AStarNode> pathList;
  double currDepth = depth;
  double depthMax = 1;
```

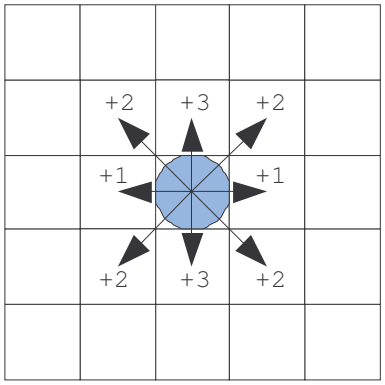
```
  // Pokud je start zaroven cil
  if (startNode.CompareNodes(goalNode)) return pathList;
  startNode.SetCostFromStart(0);
  startNode.SetEstimatedCostToGoal(startNode.GetEstimatedCostToGoal(goalNode));
  startNode.SetParentNode(Point(NULL, NULL));
  openList.Add(startNode);

  // Hlavni cyklus
  while (!openList.IsEmpty()) {
    AStarNode node;
    // Vypocet kroku
    double step = min(currDepth,depthMax);
    // Kontrola zda v open listu neni goal node
    if (openList.Contain(goalNode)) node = openList.Get(goalNode.GetTile());
      // Odstraneni prvniho uzlu z open listu
    else node = openList.PopFront();
    // Je uzel koncovym uzlem?
    if ((node.CompareNodes(goalNode)) || (currDepth <= 0))
      return ConstructPath(node);
    // Je uzel v okoli ciloveho uzlu?
    if (node.GetTile().CompareAround(goalNode.GetTile(), 1)) {
      closeList.Add(node);
      goalNode.SetParentNode(node.GetTile());
      return ConstructPath(goalNode);
    }
    // Nalezeni sousedu daneho uzlu
    list<AStarNode> neighbors = GetNeighbors(node, step, checkAccessible);
    // Pro kazdeho souseda
    list<AStarNode>::iterator i;
    for (i=neighbors.begin(); i!=neighbors.end(); i++) {
      AStarNode neighbor = *i;
      // Test zda je uzel dostupny
      if (!pitch->IsFieldEmpty(neighbor.GetTile())) continue;
      bool isOpen = openList.Contain(neighbor);
      bool isClosed = closeList.Contain(neighbor);
      float costFromStart = node.GetCostFromStart() + neighbor.GetTotalCost();
      // Vlozeni do seznamu
      if ((!isOpen && !isClosed) || (costFromStart < neighbor.GetCostFromStart()))
        neighbor.SetParentNode(node.GetTile());
        neighbor.SetCostFromStart(costFromStart);
        neighbor.SetEstimatedCostToGoal(neighbor.GetEstimatedCostToGoal(goalNode));
        if (isClosed) closeList.Remove(neighbor);
        if (!isOpen) openList.Add(neighbor);
      }
    }
    // Vlozeni do closeListu
    closeList.Add(node);
    // Update kroku
    currDepth = currDepth - step;
  }
  // Cesta nenalezena
  return pathList;
}
```

Při výpočtu nejkratší cesty je důležité ohodnotit dlaždice, po kterých objekt postupuje. Výpočet se liší pro dva typy objektů. Pro míč *Ball* je cena všech dlaždic konstantní. Pro hráče *Player*, není hodnota na první pohled zřejmá. Při výpočtu záleží na pozici, kterou hráč na hřišti zaujímá, na typu hráče (brankář, obránce, záložník, útočník) a na straně, na které hraje. Ohodnocení pro jednotlivé pozice je zobrazeno na obrázku č. 6.



(Obrázek č. 6 - *Ohodnocení dlaždic pro pozice*)

Vedle ohodnocení dlaždic pro konkrétní typ hráče je důležitý také směr, kterým se hráč pohybuje. Pokud se pohybuje *horizontálně* je cena za jeho pohyb rovna 1 bodu, *diagonálně* 2 body a *vertikálně* 3 body (obrázek č. 7). Tento způsob jsem navrhl především z toho důvodu, aby se hráči snažili pohybovat spíše směrem k oběma brankám nežli do stran k autům. Pro míč toto pravidlo samozřejmě neplatí.



(Obrázek č. 7 - *Ohodnocení směru cesty*)

Činnost algoritmu bych stručně popsal takto :

Předpokládejme, že hráč (střední záložník MID_C) se snaží najít cestu z bodu A [90,55] do bodu B [15,50], kde se nachází míč. Vstupem algoritmu jsou dva body, startovní bod A a cílový bod B. Dále je to parametr *depth*, který je roven rychlosti hráčova pohybu (viz. výše), předpokládejme, že bude nabývat hodnoty 1.85 a parametr *checkAccessible*, který je v tomto případě nastaven na *true* (bude se testovat dostupnost okolních dlaždic, aby nedocházelo ke kolizím s jinými hráči). Algoritmus v první cyklu nalezne všechny dostupné sousedy bodu A a vybere z nich ten, jehož cena podle vzorce **F = G + H** (viz. výše) bude nejnižší, tedy v tomto případě bod [89,55]. Celková cena pro tento bod bude rovna F = 11 + 750 = 761. V druhém kroku bude algoritmus postupovat obdobně a vybere bod [88,55], což odpovídá ceně F = 11 + 740 = 751. V třetím kroku se algoritmus zastaví, protože dojde k překročení limitu počtu cyklů, který je dán parametrem *depth.* Funkce tedy cestu z bodu [90,55] do bodu [88,55].

Tento příklad byl opravdu velmi jednoduchý, nebylo v něm uvažováno řada faktorů, které mohou na při hledání cesty nastat (přechod z různě oceněných pásem, obcházení jiného hráče a jiné…).

5.2. Grafický modul

Grafika je ve hře opravdu okrajovou záležitostí. V první fázi projektu tvořilo grafiku pouze několik textových výpisů na obrazovku, aby bylo při ladění hry vidět, co se ve hře děje. Po domluvě se svým vedoucím diplomové práce, Ing. Martinem Hrubým, Ph.D., jsem do projektu implementoval jednoduchý zobrazovací modul 2D hřiště.

O vytvoření 2D modelu se rovnoměrně starají dvě třídy. Třída *Graphics*, která obsahuje základní funkčnost pro kreslení na obrazovku a třída *Pitch*, která je hlavním zdrojem informací pro vykreslování.

Třída *Graphics* využívá ke své činnosti volně dostupnou multi-platformní knihovnu SDL (Simple Directmedia Layer), která umožňuje nízkoúrovňový přístup ke zvuku, klávesnici, myši, joysticku, 3D hardwaru skrze OpenGL a 2D video framebufferu. Knihovna se používá především pro pro zvukový MPEG software, emulátory, a mnoho počítačových her, například *Civilization : Call To Power.* Knihovna SDL je napsána v jazyce C, ale běžně pracuje s jazykem C++ a je napojena na řadu dalších jazyků včetně jazyku Ada, C#, D, Eiffel, Java, Lisp, Pascal, Perl, PHP, Python, SmallTalk a další. SDL je distribuováno pod licencí GNU LGPL verze 2, je tedy volně dostupná i pro komerční využití.

6. Testování

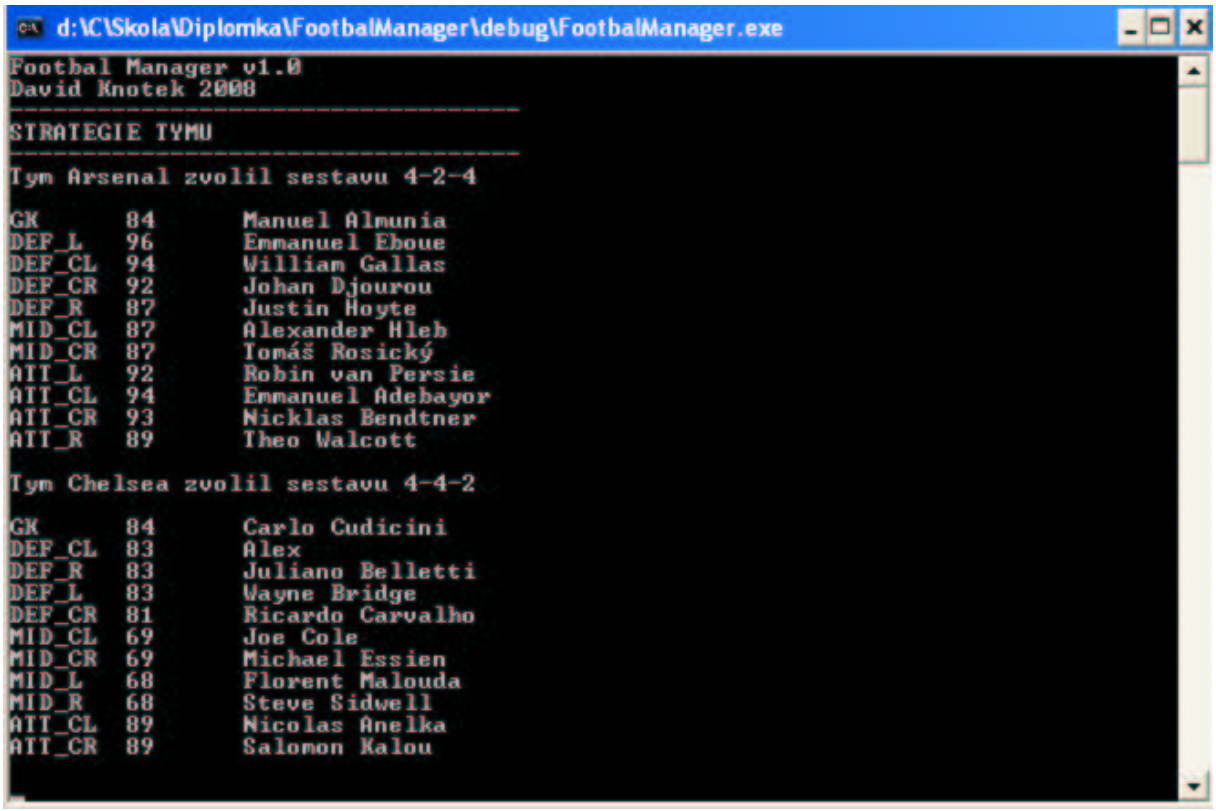
Být herním programátorem může být zábavná věc - ale hlavně je to dřina. Není to práce ani trochu jednoduchá, člověk musí zvládat složitou matematiku (zejména algebru), rozumět umělé inteligenci a tykat si s nejmodernějšími technologiemi. Testování obvykle bývá jednou z nejméně příjemných, ale zároveň také jednou z nejdůležitějších fází vývoje hry. Mělo by zabírat minimálně 30% času práce.

Při vývoji hry existuje řada verzí (fází dokončení hry). *Alfa* verze je hra dokončená a to taková, která by měla jít dohrát. Může však obsahovat chyby a proto nesmí opustit vývojové studio. *Beta* verze je již funkční hra připravená k testování a ladění posledních chybiček. Když jsou všechny odstraněny nebo vyprší deadline od vydavatele, vychází originální *release* verze, která je již určená k prodeji. Tento projekt samozřejmě není projektem tak rozsáhlým, aby bylo nutné dodržovat přesnou koncepci vývoje her. Je však dobré projekt rozdělit do určitých fází, které se konceptu blíží, aby měl vývoj hry určitý řád.

Při testování hry se zaměřuji především na dvě oblasti hlavního vývoje hry, inteligence *manažerské* a *hráčské*. Nyní podrobně rozvedu dva ukázkové příklady hry s rozdílným nastavením parametrů. Popíšu podrobné chování, které se v implementaci odehrává, aby byla zřejmá její funkčnost.

Vstupním bodem programu je třída *Main*. Její jedinou úlohou je načíst argumenty příkazové řádky a vytvořit třídu *Game*, která představuje hru. Po vytvoření instance třídy Game je možné hru spustit funkcí *Run()*. Funkce Run() obsahuje hlavní tělo programu, které je sestaveno pro ukázku simulace hry. Jsou zde vytvořeny 2 týmy, domácí Arsenál a hosté z Chelsea, kteří se utkají v přátelském zápase. Po vytvoření a inicializaci obou týmů dochází volbě sestavy pomocí asistenta manažéra. V tomto bodě do hry vstupuje inteligence *asistentská*, kterou představuje především funkce *AutoPickSquad(team).* Funkce očekává jako vstupní parametr tým protivníka, který hraje při volbě strategie důležitou roli.

Algoritmus výběru nejvhodnější strategie byl popsán v kapitole 5.1.1. Je založen na tvorbě tabulky všech dostupných strategií a poté výběr optimální strategie pro oba týmy. Tabulka strategií je vytvořena na základě hráčů, které má trenér k dispozici. Pokud tedy spustíme program, asistent týmu Arsenálu zvolí pro zápas strategii 4-2-4. Tato strategie předpokládá hodně útočný fotbal založený na kvalitních útočnících. Pokud se podíváme na sestavu Arsenálu, mělo by zde být hodně kvalitních útočníků, proto byla vybrána strategie 4-2-4. Již při prvním pohledu na obrázek č.8 je zřejmé, že hráči hostujícího týmu mají atributy horší než hráči domácích. Proto také asistent zvolil poněkud defenzivnější strategii 4-4-2, která zakládá především na silné obraně a záloze.



(Obrázek č. 8 – *Sestavy týmů*)

Nyní otestuji, kteří hráči byli vybraní do sestavy a co bych musel upravit, aby asistent zvolil jinou strategii? Pojďme otestovat například hráče na pozici ATT_L týmu domácích (Robin van Persie). Je hodnota 92 opravdu správná a tedy správná i jeho pozice?

Vzorec pro výpočet na pozici ATT_L :

```
(int)(ATT_L = att*left*((acceleration*0.2)+(pace*0.3)
            +(shooting*0.3)+(stamina*0.1)+(dribbling*0.1)));
```

Po dosazení hodnot [1]:

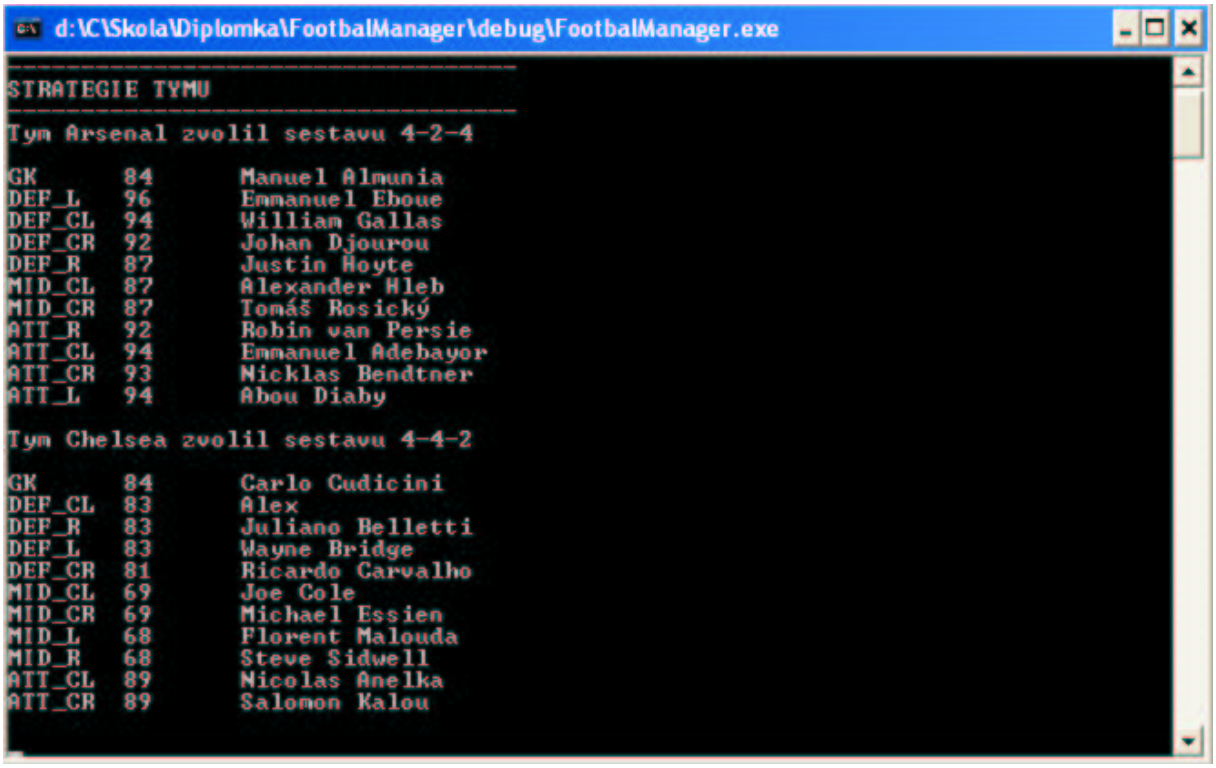
```
(int)(ATT_L = 1*1*((90*0.2)+(95*0.3)+(97*0.3)+(89*0.1)+(79*0.1))) =
          = 1*1*(18+28.5+29.1+8.9+7.9) =
          = 92.4
```

Výsledek po zaokrouhlení tedy odpovídá vypočtené hodnotě.

Proč je ale vybrán Robin van Persie zrovna na tuto pozici? Proč není vybrán například na pozici ATT_R, kde je jeho hodnota také 92? Výběr hráče na pozici totiž nezáleží pouze na jeho hodnotě, ale také na pořadí, ve kterém se na pozice vybírá. Pozice ATT_L se ve formaci 4-2-4 vybírá dříve, než pozice ATT_R, proto je na tuto pozici vybrán hráč s vyšší hodnotou.

[1] Konfigurace hodnot pro hráče jsou uloženy v textovém souboru Arsenal.txt resp. Chelsea.txt

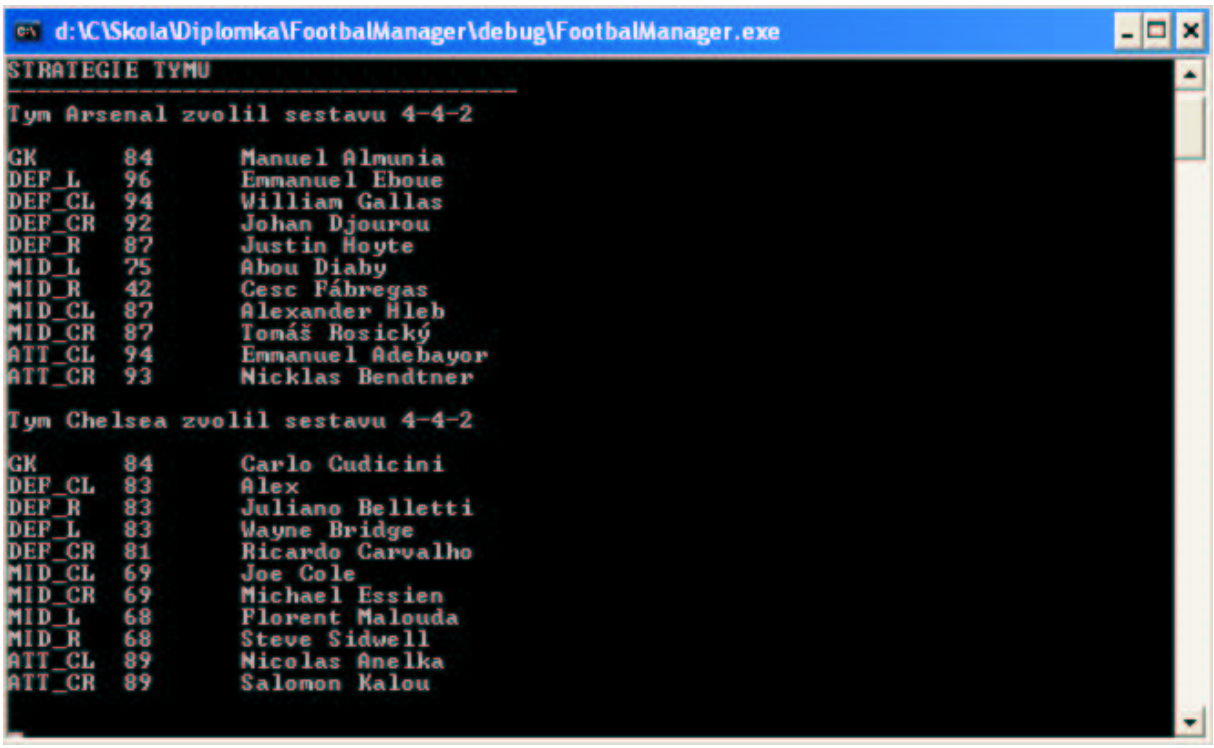
Co by se muselo stát, aby byl na pozici vybrán jiný hráč? Pokud se podrobněji podíváme na soupisku domácího týmu zjistíme, že není jiný hráč, který by mohl hrát na pozici ATT_L (není v ní žádný útočník, který umí hrát na levé straně). Přidáme-li do soupisky dalšího hráče (Abou Diaby) a nastavíme jeho hodnoty tak, aby byl schopen kvalitně hrát na požadované pozici, výsledná sestava bude odlišná (viz. obrázek č.9).



(Obrázek č. 9 – *Sestavy týmů po změně hráčů*)

Nyní jsem otestoval, že přidání, odebráním, či úpravou hráče lze modifikovat sestavu týmu, kterou asistent vybere. Toto chování se opravdu shoduje s realitou. Dejme tomu, že by celosvětově známý hráč David Backham přestoupil do týmu FC Boby Brno, také by se pravděpodobně díky jeho atributům měnila sestava týmu. Algoritmus lze tedy považovat za správný.

Nyní změním hráče týmu Arsenálu tak, aby asistent vybral jinou strategii než 4-2-4. Odeberu tedy většinu kvalitních útočníků a ponechám pouze dva (Emmanuel Adebayor a Niclas Bendtner). Z výsledku na obrázku č. 10 je patrné, že asistent manažera zvolil strategii 4-4-2, tedy strategii odlišnou.



(Obrázek č. 10 – *Sestavy týmů po odebrání útočníků*)

Poté, co asistenti obou týmů zvolí své nejlepší formace, dojde k vytvoření instance třídy *Match* (zápasu). Třída požaduje dva vstupní parametry, jimiž jsou dva týmy, domácí a hosté, kteří se utkají v zápase. Hlavní funkcí třídy Match je funkce *Play()*. Ta obsahuje hlavní smyčku hry na hřišti viz kapitola 5.1.2. Po inicializaci herního času, hřiště a 2D simulační scény, vstupuje program do hlavní smyčky, kde se opakují čtyři funkce :

- *pitch->Update()*

  Funkce provádí hřiště a všech objektů na něm podle herního času
- *pitch->DrawScene(time)*

  Funkce kreslí na obrazovku
- *UpdateTime()*

  Funkce provádí update zápasového času. I sekunda zápasu představuje jeden iterační cyklus hry, kdy jsou provedeny tyto čtyři funkce. Stav herního času a tedy i zápasu je zobrazen na 2D hřišti.
- *Sleep(SLEEP)*

  Funkce uspí vlákno na čas SLEEP (pro plynulý běh hry)

Inteligence hráčská je prováděna právě funkcí *Update()*. Tělo funkce je popsáno v kapitole 5.1.2 a pohyb hráčů na hřišti řídí rozhodovací strom na obrázku č. 5. Protože je funkce poměrně rozsáhlá a její výsledek je jen částečně předvídatelný (z důvodů faktoru náhody), popíšu zde pouze prvních par okamžiků, které se odehrávají po úvodním hvizdu rozhodčího.

Po inicializaci hřiště jsou hráči umístěni na hrací plochu a míč umístěn do středu hřiště. Dle rozhodovacího stromu, jsou nyní hráči týmu ve stavu *WithouBall(team)*, je proto algoritmicky vybrán *FindNearestPlayer(team)* hráč nejblíže k míči. Ten má snahu míč získat, je tedy volána funkce *GoToTheBall(team, player)*, která má za úkol zpracovat pohyb hráče směrem k míči (viz. kapitola 5.1.4 Algoritmus hledání cesty, A*). Chování ostatních hráčů ve stavu bez míče je STAND. Tento cyklus se opakuje do té doby, než jeden z týmů nezíská míč pod svou kontrolu. V tu chvíli začne tým s míčem ovládat funkce *WithBall(team)*. Hráč s míčem má na výběr z několika možností rozhodnutí (viz. obrázek č. 5). Pro každou s možností, které jsou zpracovány v určitém sledu (dle priority týmu), je vypočtena pravděpodobnost úspěchu a hráč pak zvolí (i s určitou mírou faktoru náhody) zda možnost provede či nikoliv. Pokud se rozhodne například pro nahrávku, algoritmus funkce *TryPassTheBall(team, player)* vybere potencionálního spoluhráče a tomu nahrávku adresuje. Zde musím upozornit na to, že cílové místo nahrávky je ovlivněno jak hráčovými atributy, tak faktorem náhody. Stav simulačního programu po 39 iteracích je uveden na obrázku č. 11

Zdlouhavé popisování výpočtů algoritmů však není náplní této dokumentace, proto těm, kteří mají zájem více proniknout do způsobů výpočtů, doporučuji krokovat program.

7. Ovládání programu

7.1. Překlad programu

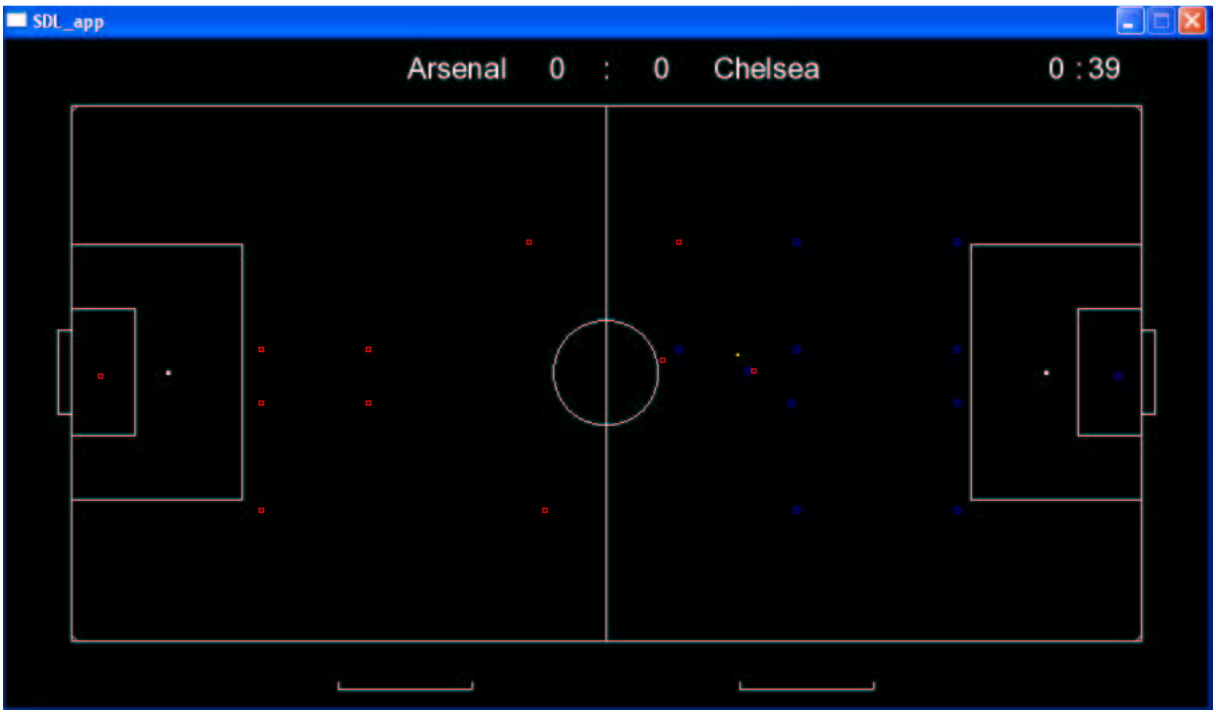
Program je přeložitelný několika způsoby. Jedním z nich je otevřít celý projekt v jednom z mnoha vývojových prostředí (VC++, DevC++, Visual Studio, …) a v prostředí jej přeložit. Další možností je použít Makefile soubor, který pomocí příkazu *make all* vytvoří spustitelný soubor. Poslední, asi nejpracnější, možností je překládat projekt postupně pomocí příkazové řádky, který však z praktických důvodů nedoporučuji.

7.2. Spouštění programu

Po přeložení a sestavení programu je vytvořen spustitelný soubor. Pro samotné spuštění je nutné mít ve stejném adresáři několik souborů. Tři dynamické knihovny *zlib1.dll, sdl_ttf.dll, libfreetype-6.dll*, soubory se sestavami týmů *Arsenal.txt* a *Chelsea.txt* a soubor se znakovou sadou *Arial.ttf*. Pokud nebude jeden z těchto souborů programem dostupný, skončí program chybou.

Po spuštěni programu jsou vytvořeny dva týmy, *Arsenal - domácí* a *Chelsea - hosté*. Oba týmy zvolí své nejlepší strategie a utkají se v simulovaném zápase. Program je pouhou demonstrací funkčnosti algoritmů umělé inteligence, jeho grafické a uživatelské rozhranní není náplní této práce a ani není realizovatelné v časových možnostech jedince v rozsahu projektu

7.3. Ukázka programu



(Obrázek č. 11 - *2D simulace zápasu*)

8. Závěr

Podařilo se mi vytvořit kostru strategické počítačové hry zaměřené na umělou inteligenci. Inteligence hry je prezentována formou 2D simulace zápasu, na níž jsou ukázány dvě ze tří forem inteligencí (*asistentská a hráčská*), které jsou v projektu implementovány. Je samozřejmé, že zdaleka ne všechny části projektu jsou vypracovány do nejmenšího detailu, snažil jsem se vytvořit ukázku toho, jak by hra měla vypadat a fungovat. Vytvořil jsem základní model, podle něhož lze hra dále vyvíjet, a dokončit tak do podoby hratelné pro koncové uživatele, případně konkurenceschopné v souboji jiných umělých inteligencí tohoto stylu. Pro kompletní dokončení hry by bylo zapotřebí více studentů, především z důvodů značných možností umělé inteligence.

Oblast her založených na umělé inteligenci, je oblíbenou oblastí mého zájmu a proto bych velice rád rozvíjel své znalosti v navazujícím doktorském studiu, kde bych mohl pokračovat na tom, co jsem začal.

Závěrem bych chtěl velice poděkovat Ing. Martinu Hrubému, Ph.D., který mi s touto prací významně pomohl a vydal mě tím správným směrem.

9. Seznam použitých zdrojů

[1] Graham Romp 1997: 1

[2] Chobot a Turnovcová 1980: 37

[3] SDL knihovna http://www.libsdl.org

[4] Wikipedie, otevřená encyklopedie http://cs.wikipedia.org/wiki/Teorie_her

[5] Bakalářská práce, Strategická počítačová hra, David Knotek 2006

[6] Portál o programování her http://www.gamedev.net/

[7] Portál o programování her v C++ http://www.cppgameprogramming.com/

[8] Visual C++ Developer Center http://msdn.microsoft.com/en-us/visualc/default.aspx

[9] Portál o programování her http://www.gamasutra.com/

10. Seznam příloh

[1] Excelová tabulka Values.xlsx obsahující hodnoty pro výpočet hráčských, týmových a inteligenčních vztahů

[2] Zdrojový kód programu FootbalManager