PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Tahová hra Deus ex machina s podporou hry po síti

2012

Jiří Gerat

Anotace

V této práci jsem implementoval tahovou hru Deus ex machina pro více jak jednoho hráče v programovacím jazyce C#. Hra je inspirována deskovou hrou Dostihy a sázky a obsahuje možnost hry po síti. Výsledkem jsou dva programy, server a klient.

Děkuji Mgr. Petru Osičkovi za vedení této bakalářské práce.

Obsah

1. Úvod ke hře Deus ex machina **7**

2. Pravidla hry **8**
2.1. Cíl hry . . . 8
2.2. Výchozí situace . . . 8
2.3. Průběh hry . . . 8
2.4. Druhy a význam polí . . . 9
2.5. Konec hry . . . 11

3. Použité prostředky **12**
3.1. Microsoft XNA Game Studio . . . 12
3.1.1. Microsoft XNA Framework . . . 12
3.1.2. Content Pipeline . . . 12
3.1.3. Struktura . . . 13
3.2. Windows Installer XML . . . 13

4. Knihovna DeusExMachinaLogic **15**
4.1. LogicManager . . . 15
4.2. Tile . . . 16
4.3. Desk . . . 17
4.4. Player . . . 18
4.5. Referee . . . 19
4.6. CardCourier . . . 20
4.7. CardPredestination . . . 20
4.8. Card . . . 21
4.9. Cards . . . 22
4.10. FileManager . . . 22
4.11. Language . . . 23
4.12. NetworkMember . . . 23
4.13. SocketExtensions . . . 25

5. Klientská část **26**
5.1. DemGame . . . 26
5.2. GameManager . . . 27
5.3. Síťová komunikace . . . 27
5.4. GUI . . . 29
5.5. Ovládání . . . 30

6. Serverová část **31**
6.1. Síťová komunikace . . . 31

7. Postup pro sestavení programu **34**

Závěr **35**

Conclusions **36**

Reference **37**

A. Obsah přiloženého CD **38**

Seznam obrázků

1. Jednotlivé části Content Pipeline. 13
2. Stavový diagram tahu. 15
3. Průběh provedení akce. 20
4. Diagram herní struktury. 26
5. Posílání změn v daném tahu. 29

1. Úvod ke hře Deus ex machina

Úkolem této práce bylo vytvořit vlastní tahovou hru inspirovanou známou deskovou hrou *Dostihy a sázky*. Tato hra dostala jméno *Deus ex machina*.

Deus ex machina se odehrává v antickém městě. Hráči na začátku hry představují prosté kupce, kteří ovšem lační po moci a vlivu. Každý se tedy snaží ovládnout město a pomoci jim k tomu mají políčka na desce představující části města. Většina políček jsou bojovníci, kteří brání dané políčko a vybírají poplatky od každého, kdo není jejich majitelem, přičemž každý bojovník patří do nějaké skupiny více bojovníků stejné barvy. Pokud hráč vlastní celou skupinu jedné barvy, může si zvýšit zisky z poplatků od ostatních hráčů vylepšením svých bojovníku, to jest koupí lepšího vybavení.

V pokročilejší části hry se začnou ve městě objevovat zloději, které si hráč může najmout, aby okrádali ostatní protivníky, ovšem musí mít určitý vliv v daném teritoriu města. Ve hře zůstávají jen ti nejsilnější, takže pokud je nějaký hráč příliš slabý a nemá dostatečný vliv ve městě oproti ostatním kupcům, je vyřazen ze hry a musí hledat štěstí jinde.

Deus ex machina opravdu pracuje na stejných principech jako Dostihy a sázky. Ve spoustu ohledech ale tyto principy rozšiřuje a přidává nové prvky tak, aby byla hra pro hráče atraktivnější a záleželo trochu více na rozmyslu a trochu méně na náhodě. U her tohoto typu, tedy tahových her ovlivněných hody kostkou a působením náhodně vylosovaných karet, ale nelze prvek náhody vymítit úplně a není to ani žádoucí. S náhodnými prvky působí totiž hra živě (svojí nevyzpytatelností). Správným poměrem faktoru náhody a faktoru hráče tedy lze dosáhnout příjemné hratelnosti i pokud hraje neznalejší hráč této hry s hráčem znalejším.

2. Pravidla hry

Hráči jezdí dokola po jedné hrací desce představující antické město a provádí akce vyplývající z polí, na které stoupnou.

2.1. Cíl hry

Kupováním skupin bojovníků, zlodějů a vybavení získat co největší vliv ve městě a zruinovat tak protihráče.

2.2. Výchozí situace

Výchozí nastavení desky a hráčů:

- Každý hráč na začátku dostane 2990,- a jeden *žeton pohybu*. Žeton pohybu může hráč využít před svým hodem a přičíst tak nebo odečíst 1 od výsledného hodu. Pokud díky tomu zůstane na místě, provede akci na daném poli. Hráč může použít pouze jeden žeton pohybu najednou.
- Každý hráč začíná z prvního pole ve hře *Domov*.
- U deskové verze je třeba zamíchat balíčky karet *Posel* a *Předurčení* a umístit je na jejich vyznačené místo na hrací desce.

2.3. Průběh hry

Průběh hry je řízen následujícími body:

- Hráči se v tazích pravidelně střídají. Hráč musí táhnou pokud není dáno jinak.
- Pokud je hráč na tahu a má přidělené nějaké *žetony čekání*, jeden odevzdá a je přeskočen.
- Hráči postupují po hrací desce kupředu přes druhé políčko ve hře *Hilarion* o tolik polí, kolik padne na kostce.
- Padne-li hráči číslo 6, hází hráč znovu a hody se sčítají. Pokud hráč hodí 6 i v druhém hodu, přesune se na políčko *Nepřízeň bohů* a je považován za hráče v *nepřízni bohů*. Tah pro něj končí.
- Pokud je hráč v *nepřízni bohů*, na začátku každého kola může házet až třikrát a zkusit se tak vyprostit z nepřízně. Pokud hodí 6, je jeho status hráče v *nepřízni bohů* zrušen a může házet, jako by právě začínal kolo. Pokud 6 nehodí, zůstává hráč v nepřízni a tah pro něj končí.

- Vždy když hráč projede přes startovní pole *Domov* obdrží 400,- a jeden *žeton pohybu* (pokud není řečeno jinak, např. na kartě).
- Hráči mezi sebou nemají možnost obchodovat, pokud není stanoveno jinak.
- Veškeré poplatky, které hráči ve hře vzniknou je povinen ihned zaplatit nebo hru ukončit.
- Nemá-li hráč dostatek financí na plnění finančních závazků, je povinen odprodat bojovníky za cenu poloviční, jejich původní ceny na kartě, nebo zloděje a vybavení za cenu plnou. Pokud je prodávaný bojovník ve skupině, ve které má nějaký bojovník vybavení, musí hráč prodat nejdříve vybavení. Až poté může prodávat bojovníky.
- Každé pole má určitý význam a hráč může nebo musí ihned provést to, co z tohoto pole vyplývá.

2.4. Druhy a význam polí

Na hrací desce se vyskytují různé druhy polí s různými významy:

- ***Nepřízeň bohů*** - Hráč, který skončí svůj pohyb na tomto poli je považován za hráče v *nepřízni bohů*. Pokud už je nějaký hráč ve městě v nepřízni, je tímto osvobozen. V *nepřízni bohů* nemůže být více jak jeden hráč současně. Dokud je hráč v nepřízni, bojovníci, které vlastní, za něj nebojují a nedostává tak žádné poplatky od ostatních hráčů.
- ***Bojovník*** - Když se hráč zastaví na poli s bojovníkem a tento bojovník je dosud volný, může hráč bojovníka koupit. Pokud už bojovníka vlastní a vlastní i ostatní bojovníky z jeho skupiny, může (pokud není plně vybaven) vylepšit bojovníkovi zisky a koupit mu lepší vybavení. Hráč může koupit vždy jen jedno vybavení za jeden jeho tah. Pokud bojovník patří jinému hráči (a tento hráč není v *nepřízni bohů*), musí hráč, který se na tomto poli zastavil zaplatit poplatek daný bojovníkem a jeho vybavením. Pokud nemá hráč dostatek hotovosti, musí získat prostředky prodáním vybavení, bojovníků nebo zlodějů.

  Předkupní právo - Pokud jeden hráč vlastní alespoň půl skupiny bojovníků (tzn. u dvojčlenné skupiny alespoň jednoho a u trojčlenné skupiny alespoň dva), má na posledního ze skupiny *předkupní právo*. Pokud pak bude chtít koupit tohoto bojovníka s předkupním právem jiný hráč, musí za něj zaplatit o polovinu více. Při prodeji platí původní cena.

  Ucházení - Pokud dojde k situaci, kdy 2 hráči vlastní každý jednoho bojovníka ze stejné stáje, pak se o tohoto bojovníka *ucházejí* a žádný jiný hráč ho nemůže koupit.

- ***Žebrák / Zloděj*** - Hráč, který se zastavil na tomto poli, může využít služeb zloděje a najmout si ho, přičemž zisky plynou z poplatků za zloděje, ale jen pokud už vlastní alespoň jednu skupinu bojovníků z *teritoria zloděje*. Každý zloděj má své *teritorium*, což jsou skupiny bojovníků, které jsou na desce umístěny kolem zloděje (jedna skupina z každé strany). Pokud hráč nevlastní skupinu bojovníků z teritoria daného bojovníka, vidí na poli pouze *žebráka*, tím tah pro hráče končí.

  Poplatky za zloděje:

  - Hráč vlastní jednoho zloděje – ostatní hráči platí 100,-
  - Hráč vlastní více jak jednoho zloděje – ostatní hráči zaplatí 100 x N, kde N je počet zlodějů, které hráč vlastní.

- ***Výcvik, Aréna (Boost)*** – Když hráč zastaví na jednom z políček tzv. *boostu*, má možnost provést jednu z daných akcí na jiném poli na desce, i když na něm právě nestojí.

  Akce:

  - Pokud hráč nevlastní žádného bojovníka, nemá jak využít možnosti *boost* políčka.
  - Pokud má hráč na nějakého bojovníka *předkupní právo*, může si tohoto bojovníka koupit.
  - Pokud se hráč *uchází* o nějakého bojovníka, může tohoto bojovníka koupit a druhému hráči v ucházení tak zrušit jakýkoliv nárok na tohoto bojovníka.
  - Pokud hráči chybí k dokončení skupiny jeden bojovník, kterého vlastní jiný hráč, může si ho od něj koupit za jeho plnou cenu. Toto je jediný případ, kdy mezi sebou hráči obchodují.

- ***Předurčení, Posel*** – na těchto polích si hráč bere kartu z balíčku *Předurčení* nebo *Posel* a akce na kartě se ihned provede.

- ***Slavnosti*** – Hráč zaplatí 50,- za pořádání slavností.

- ***Bohoslužba*** – Hráč zaplatí 100,- za pořádání bohoslužby.

- ***Nařčení*** – Hráč obdrží jeden *žeton čekání*.

- ***Krčma*** – Hráč obdží *žeton otravy*. Pokud už jeden tento žeton má když se na tomto poli zastaví, vymění *žeton otravy* za *žeton čekání*. Také ztratí jeden *žeton pohybu*, pokud nějaký má.

2.5. Konec hry

- Vítězem se stává hráč, který zůstane jako poslední ve hře.
- Nemá-li hráč dostatek prostředů na zaplacení jakéhokoliv poplatku, hra pro něj končí.

3. Použité prostředky

Hra byla vytvořena v programovacím jazyku C# s pomocí knihoven *Microsoft XNA Framework* ve vývojovém prostředí *Microsoft XNA Game Studio 4.0*. Pro vytvoření instalačního souboru jsem použil WiX (*Windows Installer XML*) šablonu, jejímž výstupem je právě msi instalátor.

3.1. Microsoft XNA Game Studio

XNA Game Studio je vývojové prostředí určené pro vývoj 2D a 3D her. XNA Game Studio je rozšířením vývojového prostředí Visual Studio s podporou XNA Game Studio Framework a příslušných nástrojů. V rámci XNA Game Studio Framework jsou dodrženy návrhové vzory .NET Framework.

3.1.1. Microsoft XNA Framework

XNA Framework obsahuje knihovny tříd, určené pro vytváření her a skládá se z hlavních třech jmenných prostorů:

- ***Microsoft.Xna.Game*** - Microsoft.Xna.Game zahrnuje třídy a rozhraní pro řízení a manipulaci samotné hry.
- ***Microsoft.Xna.Pipeline*** - Microsoft.Xna.Pipeline poskytuje prostředky pro vytváření a používání vlastních Content Pipeline.
- ***Microsoft.Xna.Framework*** - Microsoft.Xna.Framework poskytuje funkcionalitu zejména v matematické a geometrické oblasti. Obsahuje třídy jako Vector, Rectangle, Point pro orientaci v 2D prostoru. Třída MathHelper poskytuje užitečné funkce pro matematické výpočty. Microsoft.Xna.Framework dále zahrnuje třídy pro přidávání a manipulaci se zvukem a hudbou, správu herního obsahu, typovou konverzi objektů a dat, práci s 2D nebo 3D grafikou, vstupy.

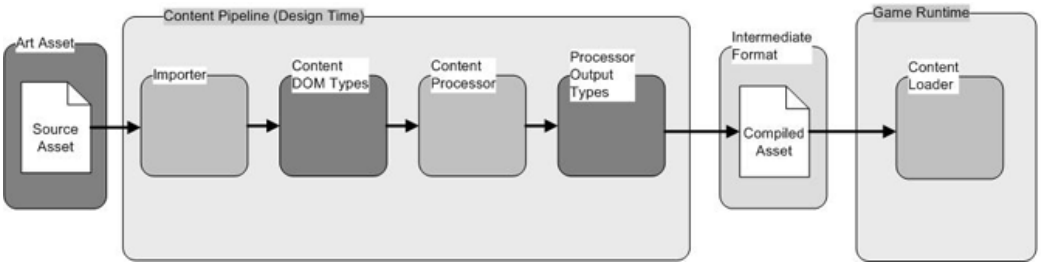
3.1.2. Content Pipeline

Content Pipeline je sada za sebou jdoucích procesů, které jsou aplikovány při kompilaci XNA hry, která načítá nějaký obsah. Na začátku je obsah ve svojí původní originální formě a postupně se transformuje do formy, která je přístupná z prostředí XNA Game Studio. Uspořádání činnosti lze vidět na 1. obrázku.

Content Pipeline tak zjednodušuje a optimalizuje manipulaci s obsahem.

```
Texture2D SpriteTexture = Game.Content.Load<Texture2D>("assetName");
```

Mezi podporované formáty patří např. 3D formáty (X, FBX), 2D formáty (BMP, JPG, PNG), XML soubory či audio formáty (WAV, MP3).



Obrázek 1. Jednotlivé části Content Pipeline.

3.1.3. Struktura

Centrální logika pro každou hru tvoří prostředí, kde se hra spustí a dále běží ve smyčce, dokud nejsou splněna ukončující kritéria.

Hra obsahuje následující základní metody:

- ***Game1()*** – Obecná inicializace
- ***LoadContent()*** – Nahrání (zpravidla grafických) zdrojů
- ***Run()*** – Vstup do herní smyčky.

  V každém kroku smyčky se provádí:

  - ***Update()*** – Čtení uživatelských vstupů, výpočty, testování podmínek pro ukončení hry
  - ***Draw()*** – Kód pro vykreslení grafického výstupu
- ***UnloadContent()*** – Uvolnění zdrojů

3.2. Windows Installer XML

Windows Installer XML je sada nástrojů pro vytváření instalačních balíčků fomátu msi z XML dokumentů.

Pomocí volně šiřitelné WiX šablony lze jednoduše vytvořit instalátor, který zkontroluje, zda jsou splněny požadavky pro bezproblémové spuštění hry vytvořené v prostředí XNA. V případě, že požadavky splněny nejsou, upozorní, nabídne odkaz či přímo nainstaluje potřebný sotfware.

Nevýhoda toho řešení spočívá v tom, že pomocí WiX nelze přímo iterovat ani vkládat celé složky s obsahem. Ve spojení s prostředím XNA to má velice nepraktické důsledky, jelikož XNA aplikace nevkládají svůj obsah do spustitelného souboru, ale mají ho celý (soubor po souboru) v Content složce. To znamená, že u projektů s četným obsahem musíme pro každou složku vložit párový tag ⟨Directory⟩ a pro každý soubor nepárový tag ⟨File/⟩. File tagy musí být navíc umístěny pro každou složku v párovém tagu ⟨Component⟩, kterému je třeba přidělit mimo jiné i nějaký GUID (globálně jedinečný identifikátor).

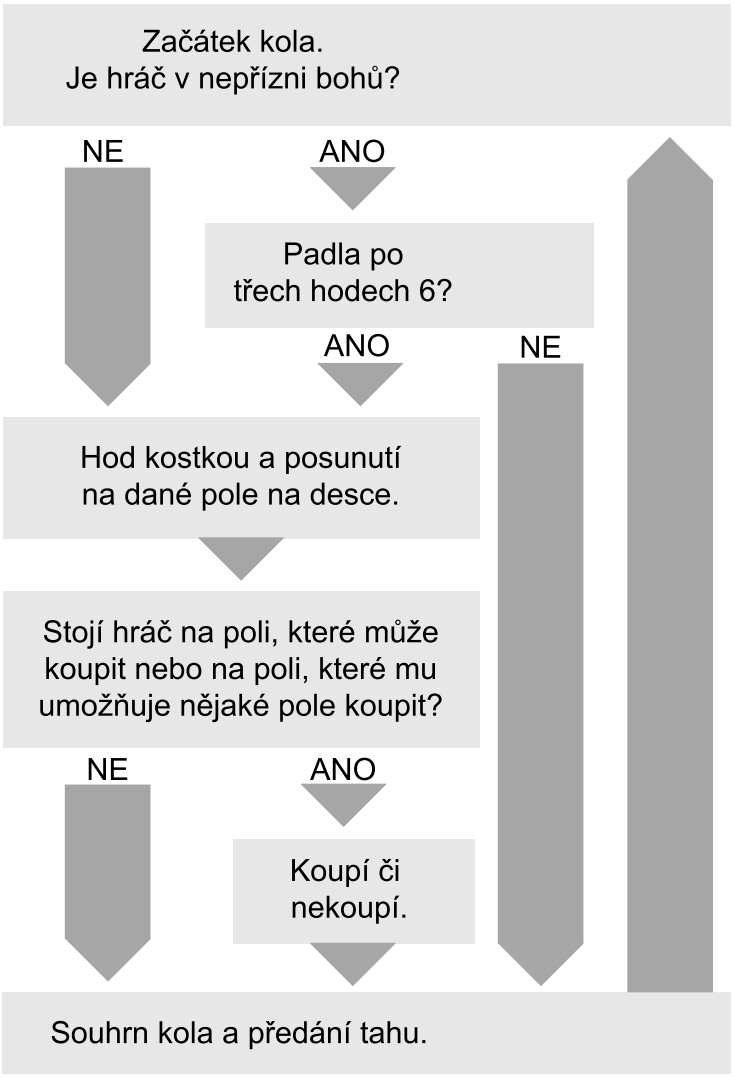
```
<Directory Id="INSTALLDIR" Name="DeusExMachina">
  <Component Id="MyComponent" Guid="guid" DiskId="1">
    <File Id="Executable" Name="fileName.exe" Source="path" />
  </Component>
</Directory>
```

Tuto nevýhodu lze řešit vytvořením jednoduché konzolové aplikace, která projde danou složku a vygeneruje její obsah do textového souboru v potřebné formě.

4. Knihovna DeusExMachinaLogic

Knihovna obsahuje třídy a metody, které jsou zodpovědné za správný chod logiky hry Deus ex machina dle zadaných pravidel.

Jak probíhá tah ukazuje následující diagram:



Obrázek 2. Stavový diagram tahu.

4.1. LogicManager

Třída Logic manager obsahuje instance základních tříd Player, Referee,

Desk, Cards, FileManager a Language pro zachycení stavu a nastavení hry. Konstuktor bere jako jediný argument instanci třídy Game (XNA.Framework) pro přístup k jednoduchému načtení obsahu desky a jazyků přes Content Pipelne.

Základní metody:

```
void NewGame(int playerCount, string[] nicks);
```

Tato metoda vytvoří novou hru tak, že vytvoří frontu hráčů *playersQueue* a naplní ji. Tato fronta určuje jednoznačně pořadí hráčů v dané hře. Jako argumenty bere metoda počet hráčů a pole řetězců, které představuje vybraná jména hráčů. Její síťová verze bere ještě jeden argument, který značí zda jde o volání serveru nebo klienta.

```
void NextPlayer();
```

Vezme prvního z fronty hráču, uloží si jeho instanci jako aktuálního hráče a zařadí ji na konec fronty.

```
void EndActualPlayer();
```

Vyřadí aktuálního hráče z fronty. Dále upraví desku tak, aby se všem polím, které hráč vlastnil, nastavila původní hodnota před koupí. Tato metoda se volá, když už hráč nemá peníze na zaplacení nebo pokud se rozhodl vzdát - tedy v konečné fázi daného hráče.

```
Player GetPlayerById(int id);
```

Praktická jednoduchá metoda, která vrací instanci hráče s daným id.

4.2. Tile

Třída představující jedno pole (dlaždici) hrací desky. Deklaruje výčtový typ *TileType*, označující druh pole.

```
enum TileType { Home, Fighter, PayEvent, WaitEvent,
   Predestination, Courier, Pub, Boost, Thief, Disfavor }
```

Část základních proměnných jako **int Id**, **string Name**, **TileType Type**, **int Group**, **int**[] **Fees**, **int Cost** a **int Value** jsou nastaveny jako veřejné, jelikož je využívá Xna Content pro jednoduché nahrátí obsahu do požadované struktury, v našem případě jednorozměrné pole typu *Tile*, přičemž názvy proměnných musí odpovídat názvům daných párových tagů v XML souboru.

Struktura XML souboru, ze kterého chceme načíst obsahy jednotlivých polí, pak vypadá takto:

```
<XnaContent>
  <Asset Type="DeusExMachinaLogic.Tile[]">
    <Item>
      <Id>2</Id>
      <Name>Hilarion</Name>
      <Type>Fighter</Type>
      <Group>1</Group>
      <Fees>10 60 140 440 580 700</Fees>
      <Cost>50</Cost>
      <Value>120</Value>
    </Item>
</Asset>
<XnaContent>
```

Každý *Item* tedy představuje jedno pole na desce a jména jeho atributů musí být shodné se jmény atributů třídy *Tile*.

Dále obsahuje dva seznamy **List⟨int⟩ presentPlayersId**, pro uložení id hráčů stojících na tomto poli a **List⟨int⟩ boostBuyPlayers**, pro uložení id hráčů, kteří se o toto pole uchází či mají předkupní právo - tedy akce, které můžou provést z pole typu *Boost*.

4.3. Desk

Tato třída vytváří hrací desku a realizuje ji ve struktuře jednorozměrného pole typu *Tile*.

Obsah tohoto pole **Tile[] tiles** se načíta z XML souboru přes *Xna Content*:

```
tiles = theGame.Content.Load<Tile[]>("XmlFiles/TilesConfig");
```

Po provedení *Content.Load⟨Tile[]⟩* se pole načte a počet prvků je dán strukturou vstupního XML souboru. Pokud struktura prvků *Item* tohoto souboru neodpovídá struktuře typu, do kterého se obsah nahrává (v našem případě třída *Tile*), nelze program zkompilovat a prostředí upozorní na chybu, takže se vylučují chyby špatné struktury za běhu programu. Po kompilaci je už soubor ve formátu *xnb* a nelze jeho XML strukturu editovat.

Metody:

```
void InitializeTiles();
```

Nastaví u každého prvku *Tile* z pole *tiles* výchozí hodnotu.

```
int ToDeskId(int value);
```

Vrátí číslo v rozmezí velikosti desky. Používá se hlavně k cyklení platných hodnot desky při postupném zvyšování o jedna.

```
Tile GetTileById(int id);
```

Jednoduchá metoda, která vrátí *Tile* daný dle vstupního id.

4.4. Player

Tato třída představuje hráče. Každý hráč má svůj identifikátor (*int id*) a jméno (*string nickname*). Dále obsahuje proměnné zachycující jeho peněžní stav, počet žetonů pohybu, počet žetonů opilost, počet žetonů čekání, aktuální pozici na desce, zda-li je hráč v nepřízni, či zda-li má možnost se z nepřízně vymanit.

Důležité metody:

```
void Buy(Tile tile);
void Sell(Tile tile);
```

Hráč může provádět různé akce. Ty nejdůležitější jsou kupování a prodávání bojovníků, zlodějů či vybavení. To zajišťují metody *Buy* a *Sell*, které berou jako argument dané pole na desce a provádějí přímou koupi bez kontroly hráčova nároku na toto pole. Vždy po koupi taky prochází desku a upravuje pole *ownedGroups*, aby bylo jasné, které skupiny hráč vlastní.

```
bool CanBuy(Tile tile);
bool CanUpgrade(Tile tile);
bool CanBoost(Tile tile);
```

Pro zjištění, zda může hráč toto pole koupit, vylepšit či provést na tomto poli akci *boostu*, slouží metody *CanBuy*, *CanUpgrade*, *CanBoost*.

```
bool HaveToPay(Tile tile);
```

V každém tahu v posledním kroku taky probíhá kontrola, zda hráč zaplatil své pohledávky na tomto poli, pomocí metody *HaveToPay*. Pohledávky se platí automaticky z peněz hráče, ale může nastat situace kdy hráč nemůže zaplatit v hotovosti a musí nejříve něco prodat. Opět bere jako argument dané pole typu *Tile*.

```
bool HaveToGet(Tile tile);
```

V každém tahu po skončení hráčova pohybu probíhá kontrola, zda má hráč obdržet nějakou částku. To nastává většinou u polí typu *Předurčení* a *Posel*.

4.5. Referee

V této třídě jsou uloženy základní informace o hře a o výchozích pravidlech. Obsahuje například hodnotu minimálního a maximální hodu na kostce, počáteční hotovost a počet žetonů pohybu hráče, id startovního pole, maximální počet hráčů ap.

Dále obsahuje atributy **bool**[] **chosenColors**, pro přehledné zaznamenání barvy hráče, a **string**[] **nicks** pro zaznamenání zvolených jmen.

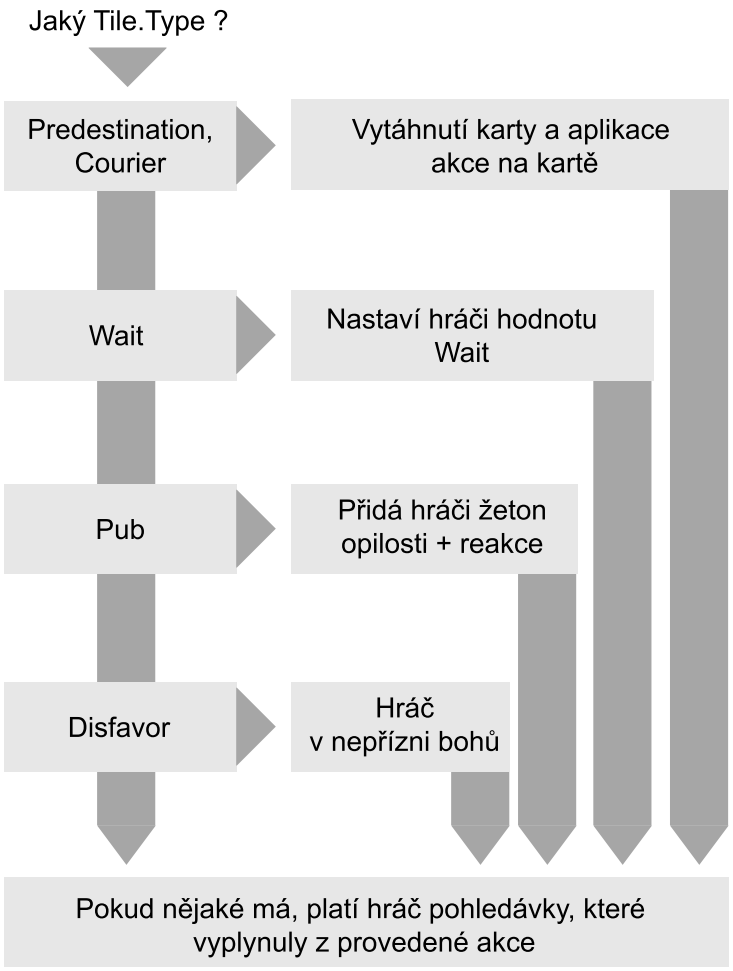
Důležité metody:

```
void Throw(int throw);
```

Metoda zařizuje hod kostkou a ukládá výsledky do dvou proměnných int *firstThrow* a *int secondThrow*. Pokud hráč hodí v prvním hodu 6, nastaví se *firstThrow* na danou hodnotu a vygeneruje číslo pro druhý hod. Pokud 6 nehodí, má secondThrow hodnotu 0 a podle této hodnoty taky potom GUI rozpoznává, zda-li je nutno ukázat i druhý hod.

```
void TileAction(Player player, Tile tile);
```

Velice důležitá metoda, která zařízuje provedení akce po skončení hráčova pohybu. Průběh ukazuje následující diagram:



Obrázek 3. Průběh provedení akce.

4.6. CardCourier

Třídá dědící ze třídy *Card*, na desce představuje kartu *Posel (Courier)*. Deklaruje výčtový typ *CourierType*, který označuje druh karty.

```
enum CourierType { Pay, Get, PayLevel }
```

Druhy těchto karet tedy obnáší pouze platby a v atributech si tak ukládá jen hodnotu, která se má vyplatit nebo být vyplacena.

4.7. CardPredestination

Třída karet typu *Předurčení (Predestination)*, která dědí ze třídy *Card*. Tato třídá už musí zvládat více funkcí než organizovat platby. Kromě deklarace výčto-

vého typu *PredestinationType*, který označuje druh karty, jenž může přikazovat přesun na desce, udílet žetony čekání nebo osvítit hráče možností zrušení nepřízně bohů:

```
enum PredestinationType
   { MoveBack, MoveForward, Move, Wait, FreeDisfavor }
```

Deklaruje také výčtový typ *MovementDestination*, který označuje druhy a místa přesunů, které karta rozkazuje:

```
enum MovementDestination { None, Standard,
   Thief, Home, Disfavor, Pub, Courier, LastFighter }
```

Obsahuje také definici struktury *ExtraMove*, která slouží k předávání pohybu navíc (děje se vždy po hráčově běžném pohybu) uživatelskému prostředí k vyhodnocení.

Metody:

```
ExtraMove GetExtraMove(LogicManager logic);
```

Slouží k vrácení aktuální *ExtraMove* hodnoty. Využívá přitom své metody *GetDestination*. *ExtraMove* hodnota obsahuje o kolik políček se má hráč posunout, na kterou stranu na desce a také jestli během toho obdržel nějakou extra hotovost.

```
int GetDestinationId(LogicManager logic);
```

Tato metoda vrací id pole, na kterém má hráč skončit dle přidělené karty. Využívá k tomu také ještě jedné metody z této třídy.

```
int FindNearest(LogicManager logic, Tile.TileType type);
```

Tato metoda vrací id výskytu nejližšího pole desky, které je typu *type*. To se využívá u karet, které přikazují pohyb (třeba zpět) na pole určitého typu.

4.8. Card

Z této třídy dědí třídy *CardPredestination* a *CardCourier*, přičemž sdružuje pouze společné atributy **int Id** a **TileType cardType**.

4.9. Cards

Třída *Cards* načíta karetní balíčky do jednorozměrných polí typu *PredestinationCard* a *CourierCard*. K načítání používá Xna Content a metodu *Load*.

```
predestinationCards = theGame.Content.
   Load<CardPredestination[]>("XmlFiles/PredestinationCards");

courierCards = theGame.Content.
   Load<CardCourier[]>("XmlFiles/CourierCards");
```

Dále obsahuje metodu *TakeCard*:

```
Card TakeCard(Tile.TileType type);
```

Jako argument bere typ pole, ze kterého byla tato akce zavolána, přičemž tuto akci může provést pouze pole typu *Courier* nebo *Predestination*. Pro daný typ poté vylosuje kartu vygenerováním náhodného indexu (pomocí třídy *Random*) v rozmezí daného pole a předá ji jako návratovou hodnotu.

4.10. FileManager

Třída *FileManager* poskytuje základní metody pro uložení a načtení herní situace.

Definuje strukturu *GameSituation*, která zachycuje přesný stav hry. Stav je zachycen **frontou hráčů**, **deskou**, **zvolenými barvami hráčů** - reprezentovanými polem *int[]* (každá barva má své id), **zvolenými jmény hráčů** - *string[]* a **aktuální fází**. Tato struktura je označena ve třídě jako serializovatelná *[Serializable]*.

Metody:

```
void SaveGame(LogicManager logic, Stream stream);
```

Jako argumety bere instanci třídy *LogicManager* pro získání aktuálního stavu hry a instanci třídy *Stream* s již vytvořeným souborem pro zápis.

Nejdříve si vytvoří serializovatelnou strukturu typu *GameSituation* a zaznamená do ní aktuální stav hry. Poté si vytvoří instanci třídy *BinaryFormatter* ze jménného prostoru *System.Runtime.Serialization* a pomocí její metody *Serialize* zapíše do souboru naši strukturu vyjadřující stav hry.

```
void LoadGame(string file, LogicManager logicManager);
```

Jako argumenty bere název souboru, ze kterého chceme načítat a instanci třídy *LogicManager* pro nastavení aktuálního stavu.

Do nového streamu načte zadaný soubor pro čtení a pomocí instance třídy *BinaryFormatter* a její metody *Deserialize* si uloží strukturu typu *GameSituation*. Z této struktury poté načte daný stav, kterou uloží do předané instance *logicManager*

4.11. Language

Třída *Language* slouží k načtení daného jazyka. Jazyky načíta z XML souboru opět pomocí Xna Content a metody *Load* do instance třídy *TextProvider*, kterou poté poskytuje.

Třída *TextProvider* obsahuje pouze proměnné typu *string* pro uložení frází, které se ve hře vyskytují. Její funkce je tedy jen poskytovat načtené fráze z XML souboru, který musí mít určitou strukturu. Tato struktura je dána proměnnými třídy *TextProvider*, jejichž názvy musí být shodné s párovými tagy v XML souboru.

```
<XnaContent>
  <Asset Type="DeusExMachinaLogic.TextProvider">
    <LocalGame>Lokální hra</LocalGame>
    <NetworkGame>Síťová hra</NetworkGame>
    .
    .
  </Asset>
</XnaContent>
```

Takový XML soubor pak načte do proměnné *LocalGame* řetězec *Lokální hra* a obdobně přiřadí zbytek obsahu. Názvy tagů v souboru musí být ve stejném pořadí jako jsou uvedeny v třídě *TextProvider*.

4.12. NetworkMember

Pro práci v síti slouží objektům využívající této knihovny třída *NetworkMember*. Z charakteru této hry lze vyvodit, že k realizaci komunikace herních událostí a postupů nepotřebujeme posílat nijak složitá data.

Stačí nám tak jednoduchá tečková separace příkazu a jeho parametrů v jednom řetězci:

```
"changeCmd.playerDisfavored.1."
```

Ostatní třídy, které chtějí komunikovat v síti z ní mohou tedy dědit a získat:

- Kompletní seznam jednotných názvů příkazů typu *string* pro práci v síti.
- Metody pro příjímaní a posílání síťových příkazů:

  ```
  void SendString(Socket receiver, string cmd);
  ```

  Pošle příjemci typu *Socket* příkaz *cmd*.

  ```
  string SendToAllConnected(string cmd, Socket[] clients);
  ```

  Pošle všem příjemcům z pole typu *Socket* příkaz *cmd*.

  ```
  string SendToAllExcept(Socket client, string cmd, Socket[] clients);
  ```

  Pošle všem příjemcům kromě příjemce daného argumentem *client* příkaz *cmd*.

  ```
  string ReceiveString(Socket sender);
  ```

  Čeká na zprávu od odesílatele typu *Socket*.

- Metody pro zpracování příkazů a jejich parametrů:

  ```
  string GetFirst(string cmd);
  ```

  Vrátí řetězec umístěný před první tečkou. Při tečce na vstupu vrací prádný řetězec.

  ```
  string GetExceptFirst(string cmd);
  ```

  Vrátí vše co je umístěno v řetězci za první tečkou. Při tečce na vstupu vrací prádný řetězec.

4.13. SocketExtensions

Tato statická třídá obsahuje metodu *IsConnected* pro zjišťování dostupnosti druhé strany typu *Socket*.

```
static bool IsConnected(this Socket socket);
```

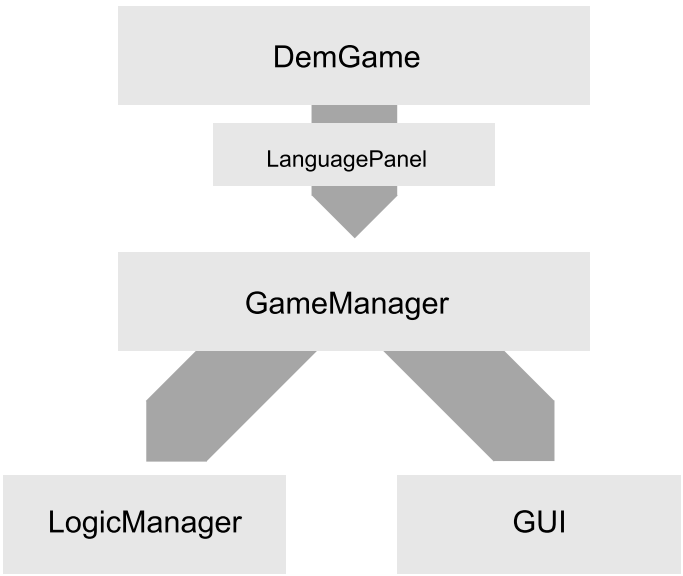
Využívá přitom odchycení vyjímky *SocketException*.

```
try
{
   return !(socket.Poll(1, SelectMode.SelectRead)
      && socket.Available == 0);
}
catch (SocketException)
{
   return false;
}
```

5. Klientská část

Klientská část má umožňovat hraní hry Deus ex machina na jednom počítači (lokální hra) i na více počítačích zároveň (síťová hra).

Hlavní třídou klientské části je třída *GameManager*, která zprostředkovává komunikaci mezi uživatelským prostředím a logikou hry, přičemž logiku získává z knihovny *DeusExMachinaLogic.dll* přes instanci její třídy LogicManager.



Obrázek 4. Diagram herní struktury.

5.1. DemGame

Herní okno vychází z třídy *DemGame*, jenž dědí z knihovny *XNA.Framework.Game*.

Hlavní funkce třídy *DemGame*:

- S využitím reference Windows.Forms vytváří instanci klasikého formuláře ze svého Window.Handle

  ```
  Form GameForm = (Form)Form.FromHandle(this.Window.Handle);
  ```

- Vytváří instanci třídy GameManager
- Zajišťuje vykreslování a aktualizaci instance GameManager metodami Draw() a Update()

5.2. GameManager

Jak už bylo řečeno, třída *GameManager* se stará o veškerou komunikaci mezi objekty hry. Instance této třídy se vytváří v konstruktoru třídy *DemGame*.

Hlavní funkce třídy *GameManager*:

- Zpracovává uživatelské vstupy jako pohyby a klikání myši nebo vstupy z klávesnice
- Obsahuje instanci třídy *LogicManager* - hlavní třídy logické knihovny *DeusExMachinaLogic.dll*
- Sdružuje a zpracovává instance třídy *Panel*, základní třídy grafického uživatelského prostředí aplikace

5.3. Síťová komunikace

Síťovou komunikaci klienta zařizuje třída *NetworkClient*. Ta je zodpovědná za předávání, příjímání a zpracovávání zpráv od serveru. K přenosu zpráv používá **Tcp protokol**. Strukturu zpráv a metody pro jejich posílání dědí ze třídy *NetworkMember*, kterou poskytuje logická knihovna *DeusExMachinaLogic.dll*.

Po zavolání metody *Connect* vytvoří klient nové vlákno, které v nekonečném cyklu příjímá zprávy od serveru, dokud nedostane *exitCmd* či sám nepřeruší spojení. Pokud přeruší spojení sám, odpojí se od serveru a jeho funkce v síti končí. O informování ostatních klientů se stará server, který v jednom ze svých vláken kontroluje, zda se klient neodpojil a případně na to reaguje. Klienti připojení k serveru se mohou nacházet postupně ve dvou fázích - ve fázi **Lobby** a ve fázi **Game**.

Ve fázi **Lobby** si každý připojený hráč volí barvu a jméno. Dále musí oznámit serveru, že je připraven ke hře. Komunikace probíhá následovně:

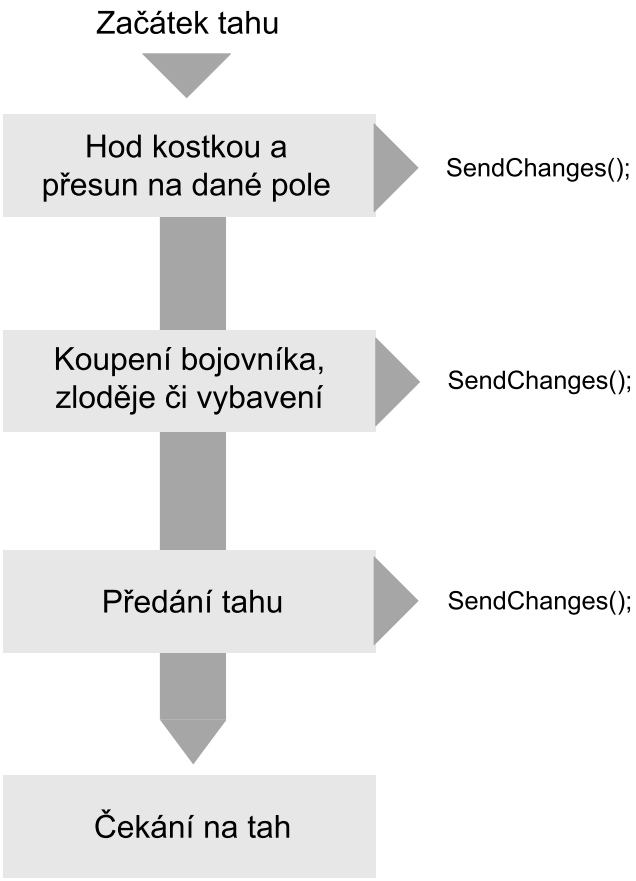
1. Klient se připojí k serveru
    - První připojený klient je označen jako *host*.
    - Pokud se připojí klient k serveru, kde byl už host zvolen, obdrží zprávu o aktualním výběru barev a jmen a o aktuálním počtu připojených a připravených hráčů.
2. Výběr barvy klienta
    - Odešle požadavek o výběru dané barvy serveru a čeká na potvrzení.

- Pokud se vrátí kladná odpověď, server úspěšné barvu zarezervoval a klient může pokračovat ve volbě jména.
- Pokud se vrátí záporná odpověď, tuto barvu si stihl zarezervovat jiný klient v síti a klientovi se znepřístupní výběr této barvy.

3. Výběr jména klienta
   - Po výběru a potvrzení jména pošle serveru zprávu o aktualizaci jména.
4. Oznámení připravenosti klienta
   - Klient posílá serveru zprávu o tom, že je připraven či připravenost ruší a čeká na potvrzení.
   - Po potvrzení si klient nastaví aktuální stav připravenosti (připraven/nepřipraven).
5. Konec výběru, začátek hry
   - Hru může začít pouze klient s označením *host*.
   - Pokud jsou připojeni a připraveni alespoň dva hráči, může host poslat požadavek o zahájení hry.
   - Jakmile klientovi přijde zpráva o zahájení hry, přechází do fáze **Game**.

Ve fázi **Game** si každý klient nastaví výchozí stav a čeká až mu server pošle zprávu, že je na tahu.

1. Klient čeká na předání tahu.
   - V tomto kroku může klient poslat serveru zprávu, že se vzdává, čímž se od serveru odpojí a hra pro něj končí.
   - Může se také přímo odpojit, čímž pro něj hra končí.
2. Klientovi byl předán tah.
   - V tomto kroku si klient ukládá změny, které na desce během jeho tahu provádí a vždy na konci určité fáze změny pošle serveru. Posílání zpráv v určitých fázích znázorňuje 5. obrázek.
   - Po poslední fázi posílá serveru zprávu o předání tahu a čeká až bude opět na řadě.
   - Klient může poslat zprávu, že se vzdává - odpojí se a hra pro něj končí.
3. Klient obdrží odpojení jiného klienta.
   - Pokud tento klient zůstal ve hře jediný, hra končí a klient oznámí výhru.
   - Jinak oznámí odpojení a čeká na tah nebo porkračuje ve svém tahu.



Obrázek 5. Posílání změn v daném tahu.

5.4. GUI

Grafické uživatelské prostředí klientské aplikace poskytuje kromě tříd základních komponentů *Button*, *Sprite*, *Text*, *TextBox* a *Dialog*, také třídy komponentů méně běžných, které upravují zobrazení herní situace, *Panel*, *Fader* a *DescriptionComponent*.

Panel představuje základní prvek GUI této hry. V panelu lze jednoduše přiřazovat akce jeho tlačítkům a každý panel má svoje označení, podle kterého se k němu přes instanci třídy *GameManager* můžeme dostat. Každý panel může obsahovat další panely, které se ukládají do seznamu jeho potomků.

Fader vytváří prvek, který má za úkol v dané chvíli schovat či ukázat některý prvek v okně postupným přechodem z neviditelného *faderu* do viditelného s barvou pozadí.

DescriptionComponent vytváří pro daný prvek text, který se po najetí

myši na daný prvek zobrazí u kurzoru.

5.5. Ovládání

Hra má poměrně intuitivní ovládání. Po startu klienta se zobrazí formulář pro výběr jazyka. Poté se dostaneme do hlavního menu, kde máme na výběr *Lokální hru* a *Síťovou hru*. Po načtení jazyka je pořád aktivní rám, který umožňuje přesouvat okno, minimalizovat, či ukončit aplikaci. Při ukončení následuje dialogové okno.

- **Lokální hra** - Po zvolení lokální hry se objeví obrazovka s možnými barvami hráčů fungujícími jako tlačítka. Dále je zde tlačítko *Zpět*, *Nahrát hru* a tlačítko *Hrej*, které se aktivuje po zvolení alespoň dvou hráčů pro hru. Po kliknutí na barvu si hráč zvolí jméno a potvrdí klávesou Enter. Tuto barvu může hráč zrušit kliknutím na křížek v pravém horním rohu barvy.

- **Síťová hra** - Po zvolení síťové hry se objeví obrazovka pro zadání ip adresy. Pokud ip adresu nezadáme, můžeme kliknout na tlačítko *Založit* a spustit tak server, ke kterému se rovnou připojíme. Pokud ip adresu zadáme, zkusí se klient připojit k dané adrese. Pokud toto připojení selže, objeví se dialogové okno. Po připojení k serveru se nám zobrazí podobná obrazovka jako u *Lokání hry* s tím rozdílem, že je zde tlačítko *Připraven*, které se aktivuje po výběru barvy a potvrzení jména. Toto tlačítko musí aktivovat každý z připojených uživatelů, aby hru mohl začít uživatel, který hru zakládal, zahájit.

- **Herní panel** - Po zahájení se objeví *Herní panel*. Vlevo nahoře vidíme postupně tlačítko zpět do menu, jméno hráče, hotovost hráče, počet žetonů pohybu, zobrazení žetonu opilosti, zobrazení hráče v nepřízni a odsazené tláčítko pro kapitulaci. Uprostřed okna je zobrazení desky a pod ní výčet bojovníků, které hráč vlastní. Po desce se může uživatel libovolně rozhlížet bočními šipkami. Hlavní částí je box, který je umístěn v pravé horní části obrazovky. V něm se provádí akce tahu. Pod tím boxem je box, který složí jako zobrazovač informací o poli, na které uživatel kliknul, přičemž může klikat na desce či ve výčtu svých bojovníků.

6. Serverová část

Serverová část tvoří jednoduchá konzolová aplikace, která zajišťuje připojení a sběr zpráv od klientů. Dále zachycuje odpojení klientů od serveru zodpovídá za informování ostatních připojených klientů. Pro ukládání aktuálního stavu hry si vytváří instanci třídy *LogicManager* z logické knihovny *DeusExMachinaLogic.dll*.

6.1. Síťová komunikace

O síťovou komunikaci se stará třída *Server*, která dědí z třídy *NetworkMember* logické knihovny *DeusExMachinaLogic.dll*. Ta poskytuje serveru:

- Dostupné příkazy pro komunikaci mezi členy síťového spojení
- Metody pro zpracování zpráv dané struktury
- Metody pro přijímání a odesílání zpráv v síti

Po spuštění aplikace se server nachází v jedné ze tří fází - **Waiting**, **Lobby** a **Game**.

Při fázi **Waiting**, pouze čeká na prvního klienta, který se připojí. Toho označí jako *hosta* a přechází do fáze **Lobby**.

Během fáze **Lobby** čeká na připojení dalších klientů. Při této fázi mohou nastat tyto situace:

- Připojení nového klienta
  - Při připojení nového klienta zvýší celkový počet připojených klientů
  - Informuje všechny ostatní připojené klienty o novém klientovi v síti
  - Zkontroluje, zda již není server plný. Pokud je server plný, nepřijímá nové spojení dokud se neuvolní místo.
- Klient žádá o rezervaci barvy
  - Server zjistí, zda si jiný hráč tuto barvu už nerezervoval.
  - Pokud je barva volná, rezervuje ji a pošle kladnou odpověď.
  - Pokud je barva již zabraná jiným klientem, odpoví záporně.
- Klient ruší výběr barvy
  - Server uvolní tuto barvu a pošle potvrzení.
  - Pokud byl hráč připraven, nastaví nepřipraven.

  - Nový stav pošle všem připojeným klientům.

- Klient posílá aktualizaci svého jména
  - Aktualizaci jména klienta server přeposílá všem ostatním připojeným klientům.
  - Dále si sám ukládá tuto aktualizaci, aby měl přehled o aktuálním stavu.

- Klient posílá zprávu o změné jeho připravenosti
  - Pokud je klient připraven, nastaví server u klienta, že je připraven a pošle mu potvrzení. Dále pošle aktualizaci všem ostatním připojeným klientům.
  - Pokud je klient není připraven, nastaví server u klienta, že není připraven a pošle mu potvrzení. Dále pošle aktualizaci všem ostatním připojeným klientům.

- Klient pošle zprávu o zahájení hry
  - Server si založí novou hru a pošle všem připojeným klientům zprávu o zahájení hry a kdo je na tahu.

- Server zachytí odpojení klienta
  - Pokud měl klient rezervovanou barvu, zruší rezervaci.
  - Pokud byl klient připraven, sníží hodnotu celkových připravených hráčů ve hře.
  - O vzniklých změnách informuje ostatní připojené klienty.
  - Pokud je klient, který se odpojuje označen jako *host*, aplikace pošle všem připojeným klientům exit zprávu a ukončí svou činnost.

Během fáze **Game** server hlavně zpracovává zprávy klientů o aktualizaci herního stavu.

- Obdžení změn ve hře od klienta
  - Server si nastaví změny ve své instanci logiky a přepošle změny ostatním připojeným klientům.

- Klient pošle zprávu o předání tahu
  - Server vybere dle daných pravidel dalšího hráče, který má provést tah a pošle zprávu jeho klientovi.

- Klient se vzdává nebo se odpojuje
  - Tak jako tak už server nepočítá s komunikací s tímto klientem, sníží počet přípojených hráčů a informuje o tom ostatní připojené klienty.

7. Postup pro sestavení programu

Pro sestavení programu (server a klient) je potřeba Microsoft Visual Studio 2010 a XNA Game Studio 4.0. V tomto programu se otevře soubor *DeusExMachinaXNA.sln*. Poté je možné stisknutím kombinace kláves CTRL+SHIFT+B programy sestavit. Stisknutím klávesy F5 se programy sestaví a spustí se klientská část.

Závěr

V programovacím jazyku C# a prostředí XNA Game Studio jsem vytvořil tahovou hru Deus ex machina, která je inspirována známou českou deskovou hrou Dostihy a sázky. Ta je tvořena klientskou a serverovou částí a umožňuje hraní této hry po síti. Tato hra má oproti hře Dostihy a sázky změněná pravidla, čímž jsem chtěl docílit lepší hratelnosti a zajímavějšího průběhu.

Hra by šla dále vyvíjet a obohatit o další uživatelské funkce, které by zlepšily atraktivitu hry pro koncového uživatele.

Conclusions

I managed to create a turn-based game called Deus ex machina in C# language and XNA Game Studio environment. The game is inspired by well-known czech game Dostihy a sázky. Application includes client and server part and these parts allow network play of this game. This game has a bit different rules than game Dostihy a sázky and it should contribute to better gaming experience and more interesting continuance.

Development of the game could continue with adding more interesting user interface fuctions, which would be more attractive for user.

Reference

[1] *http://www.deskovehry.info* [online]. 2009 [cit. 2012-08-11]. Kompletní popis pravidel hry Dostihy a sázky. Dostupné z WWW: `http://www.deskovehry.info/pravidla/dostihysazky.htm`.

[2] *MSDN* [online]. 2012 [cit. 2012-08-13]. Content Pipeline. Dostupné z WWW: `http://msdn.microsoft.com/en-us/library/bb447745.aspx`.

[3] *xnadevelopment.com* [online]. 2012 [cit. 2012-08-11]. Content Pipeline. Dostupné z WWW: `http://www.xnadevelopment.com/tutorials.shtml`.

A. Obsah přiloženého CD

`bin/`

Instalátor DeusExMachinaSetup.msi potřebný ke správné instalaci.

`doc/`

Dokumentace práce ve formátu PDF, vytvořená dle závazného stylu KI PřF pro diplomové práce, včetně všech příloh, a všechny soubory nutné pro bezproblémové vygenerování PDF souboru dokumentace.

`src/`

Kompletní zdrojové texty programu Deus ex machina se všemi potřebnými zdrojovými texty a soubory pro vytvoření spustitelných verzí programu.

`readme.txt`

Instrukce pro instalaci a spuštění programu Deus ex machina, včetně požadavků pro jeho provoz.

Navíc CD obsahuje:

`install/`

Instalátor Microsoft XNA Framework 4.0, který je nutný pro provoz programu.

U veškerých odjinud převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro materiály, u kterých toto není splněno, je uveden jejich zdroj (webová adresa) v textu dokumentace práce nebo v souboru readme.txt.