PrsiCardGame

Zadání

Navrhněte konzolový projekt, do kterého zaimplementujete karetní hru prší
(pravidla: http://karetnihry.blogspot.cz/2010/05/prsi-pravidla.html).
Hrát budou hráči střídající se u jedné konzole.
Požadovaným výstupem je funkční konzolová aplikace hry prší pro dva hráče.

Řešení z pohledu uživatele aplikace

Ovládání a input uživatele

Vzhledem k omezenosti konzolových aplikací se veškerá navigace a ovládání v aplikaci provádí pomocí číselných kláves – aplikace zobrazí položky, které uživatel může zvolit, a ten tak následně provede stisknutím příslušné klávesy na klávesnici. Výjimky z tohoto pravidla jsou tři – první je potvrzení volby nebo výběr kladné odpovědi pomocí klávesy [y], druhou je zadání svého jména, třetí pak vybrání příslušné karty v samotné hře, kde je kromě stisknutí klávesy s příslušným číslem ještě potřeba svou volbu potvrdit stisknutím klávesy enter. Uživatelský vstup je ochráněn proti chybnému zadání, takže by uživatel neměl být schopný svou chybou způsobit pád nebo nezamýšlené chování aplikace.

Navigace

Po spuštění se zobrazí hlavní menu, ve kterém má uživatel na výběr ze čtyř voleb:

1. *New game* – zahájí samotnou partii prší
2. *Tutorial* – seznámí uživatele s ovládáním a možnostmi aplikace
3. *Options* – umožní uživateli aktivovat nebo deaktivovat možnost přebíjení karty Eso
4. *Quit* – ukončí aplikaci

New game:

Před zahájením hry jsou uživatelé postupně vyzváni, aby zadali svá jména. To umožní uživatelům rozpoznat, který hráč je na řadě, aniž by si museli pamatovat, zda jsou označeni jako *player 1,* nebo *player 2*. Po zadání obou jmen hra začíná.

Hra

V každém tahu jsou hráči zobrazeny informace o:

- čísle kola – dává uživatelům jednoduchou a přesnou informaci o tom, jak dlouho již soupeří,
- hráči na tahu – pomáhá uživatelům v orientaci ve hře při omezených možnostech konzolové aplikace,
- poslední provedené akci – pomáhá uživateli udržovat si přehled o vývoji partie,
- poslední hrané kartě – na kterou musí položit jednu ze svých karet, mají-li k tomu možnost,
- speciálním efektu poslední hrané karty – byla-li poslední kartou karta se speciálním efektem
- očíslované kartě/očíslovaných kartách v ruce – ze kterých mohou hráči hrát svůj tah
- možnosti líznout kartu nebo neudělat nic – je-li to v tomto tahu povolené

Uživatel zadáním čísla zvolí svůj tah a potvrdí klávesou enter. Je-li tento tah v rámci pravidel, je mu umožněn. V případě výběru svrška je vyzván k výběru barvy. Na tah se nyní dostává druhý hráč, který má stejné možnosti. Po odehrání prvního hráče v prvním kole však může být hráč ve svém tahu omezen speciální akcí karty, kterou hrál jeho spoluhráč, tyto karty jsou:

- eso – pokud je to povolené a zároveň hráč má jiné eso, je mu umožněno ho přebít, v opačném případě nedělá v tomto tahu nic

- svršek – mění barvu karty
- sedmička – hráč jej musí přebít jinou sedmičkou nebo si líznout požadovaný počet karet
- poslední protihráčova karta – je-li soupeřova ruka prázdná, ale hráč drží v rukách červenou sedmičku, kterou je možné legálně zahrát, je vyzván, aby ji zahrál a vrátil protihráče do hry

Pokud se jeden z hráčů dokázal zbavit karet a druhý ho tam nemůže legálně vrátit, hra oznámí vítěze a po stisknutí libovolné klávesy vrátí uživatele do menu.

Technické provedení

Program

Vstupní bod aplikace slouží pouze k úvodnímu nastavení konzole, vytvoření instance třídy menu a zavolání jeho metody *ShowMenu()*, která zobrazí úvodní menu.
Alternativně by bylo možné základní menu vytvořit v této třídě, já osobně však preferuji držet metodu *Main()* co nejjednodušší a na každý delší kód vytvořit třídu, která se o tuto část stará.

Menu

Při konstrukci si vytvoří a uloží instanci třídy GamePlay jako private field, která drží základní data nutná pro hru.
Public metoda *ShowMenu()*, která je volána z třídy *Program,* obsahuje cyklus *while*, který stále dokola zobrazuje hlavní menu v případě, že si uživatel nevybere žádnou z voleb, které mu *ShowMenu()* nabízí, nebo se do něj vrátí z jiné části programu. Cyklus je přerušen až ukončením programu.
Volby *Start new game a options* uživatele přenesou do jiných tříd, aniž by došlo k přerušení cyklu.
Volba *Quit* ukončí program po potvrzení uživatele.
Volba *Tutorial* zavolá private metodu *Tutorial()*, která uživateli napoví, jak s aplikací zacházet.
Alternativně by nebylo nutné metodu *Tutorial()* vůbec vytvářet, avšak držet kód v case switche co nejkratší dle mého názoru přispívá k přehlednosti celé metody.
Místo switche by bylo možné použít kombinaci statementů if(), else if() a else, switch však působí čistěji, což také přispívá k čitelnosti kódu. Ze stejného důvodu jsem ho využil i na dalších místech aplikace.
Nekonečný cyklus while() by mohl být nahrazen while() cyklem s podmínkou, která nabude pravdivosti, když uživatel zvolí možnost Quit. Program by pak byl ukončen tím, že by dospěl na konec kódu na rozdíl od současného stavu, kdy končí zavoláním metody Exit() na třídě Environment.
Nekonečný cyklus while() by šel nahradit i nekonečným cyklem for(), dle mého jde však o horší volbu, která by čtení kódu znesnadnila. Ze stejného důvodu je tento cyklus použit i ve třídě *GamePlay*.

GameOptions

Jednoduchá statická třída sloužící k uložení nastavení. Nastavení obsahuje jedinou položku – povolit přebití esa, nebo jej zakázat. Informace je uložena jako *option.txt* do složky se spouštěcím souborem.
K výběru hodnoty opět slouží switch(), k samotnému uložení je určena private metoda *SaveOption(bool overrulable)*, jejím použitím je omezeno opakované psaní kódu na minimum.
Metoda využívá StreamWriter k zapsání hodnoty *overrulable*, kterou dostala v parametru.
Na třídě se ještě nachází public metoda *LoadOption()*, která načte hodnotu ze zmiňovaného souboru, pokud existuje. Pokud ne, vrátí výchozí hodnotu, která je nastavená na true.
Soubor option.txt přímo ve složce se spouštěcím souborem považuji u reálných projektů za bad practice, v případě takto malé ukázkové aplikace jsem si toto zjednodušení dovolil.
Místo while() jsem tentokrát k zajištění opakování využil statement default, který v případě neplatného inputu od uživatele metodu zavolá znovu (rekurze). V případě většího množství chybných inputů se do paměti dostane větší množství těchto spuštěných metod. Mohlo by tak dojít k zahlcení paměti, v tomto případě by však uživatel musel nějakou dobu facerollovat po klávesnici, než by se v paměti nahromadil přehršel dat (netestováno).

Třída je statická, protože se nepředpokládá vytváření instancí, ani její funkce nevyžaduje držení jakýchkoli dat.

Card

Ve stejném souboru jsou definované enumy *Suit* s jednotlivými barvami karet a *Name* s jednotlivými hodnotami karet.
Třída *Card* má dvě public properties – *CardSuit* typu *Suit* držící barvu karty a *CardName* typu *Name* držící hodnotu karty. Karta by měla po vytvoření vždy zůstat stejná, proto má settery pro *CardSuit* a *CardName* nastavené jako private a o jejich naplnění se stará konstruktor.
Protože karta bez hodnoty a barvy není úplná, vyžaduje konstruktor jejich hodnoty jako parametry. Po jejich uložení vloží konstruktor do fieldu cardSuit znaky reprezentující barvu a hodnotu příslušné karty.
Public metoda *Draw()* slouží k vykreslení karty. Pomocí Console.WriteLine() vytvoří obrys karty a do středu vloží data z field *cardString*.

Player

Třída obsahuje data a logiku sloužící k obsluze hráče v těsné spolupráci s třídou *GamePlay*.
Properties:

1. *Nick* – slouží k uložení jména hráče. Nastavuje se během konstrukce, takže má setter jako private.
2. *Hand* – je kolekce karet v hráčově ruce. V případě, že je prázdná, hráč vyhrává. Setter je public kvůli úvodnímu rozdávání karet probíhajícímu ve třídě *GamePlay*.

Fields:

1. *gamePlay* – slouží k uložení *GamePlaye*, kterému se budou posílat výsledky uživatelského inputu
2. *overrulableAce* – bool sloužící k uložení momentálního nastavení pravidla o přebíjení es

Konstruktor bere v parametrech *GamePlay* pro field *gamePlay* a *nick* hráče, také dochází k iniciaci *Hand* a načtení hodnoty pro field *overrulableAce* pomocí *LoadOption()* na statické třídě *GameOption*.
Ve třídě se dále nachází pět public metod, které jsou volané z *gamePlaye*. Ty se starají o úkony související s kartou, která byla zahrána jako poslední, případně s aktuálním stavem hry (více v části o třídě *GamePlay*.
Tyto metody si volají na pomoc některé private metody. *PlayNormal()* a ActiveJack() volají metodu *Play()*, která si jako parametr bere hratelné karty, které jsou v kolekci Hand, a po provedení dalších úkonů (pomocí dalších private metod i vlastních statementů) pak odešle do *gamePlaye* informaci o výsledku.
Zbylé metody *Play()* nepoužívají a samy si volají další pomocné metody, těmi jsou *DrawUI()* sloužící k vykreslení uživatelského rozhraní při tahu a *ManageUserInput()*, který řeší input od uživatele v různých situacích. Obě tyto metody dostávají potřebná data (hratelné karty, povolení lízat a bool zda 0 od uživatele přikazuje nedělat nic nebo lízat, případně není povolené ani jedno) v parametrech.

GamePlay

Hlavní třída starající se o samotné hraní. Ukládá data nutná pro hladký běh partie a dává instancím třídy *Player* informace o poslední hrané kartě, jejím speciálním efektu a skutečnosti, zda tento efekt již pozbyl účinku. Také přijímá od instancí třídy *Player* data o jí hrané kartě.
fields:

1. *deck* – kolekce všech karet
2. *shufffledDeck* – zamíchaná kolekce všech karet, ze které se rozdává a ze které si hráči lížou
3. *justPlayed* – udržuje informaci o tom, jaká karta byla právě někým hrána. Pokud speciální efekt karty již vypršel, ale zároveň další karta ještě nebyla odehrána, nastaví se na null
4. *usedCards* – kolekce všech karet, které hráči odehráli, v případě, že dojdou karty v *shuffledDeck*, se *usedCards* přesunou tam a znovu ji tak naplní

5. *sevenAmounts* – drží informaci o tom, kolik karet si má líznout hráč, proti kterému byla odehrána sedmička, v případě přebití hodnota narůstá vždy o dva. Po dolízání všech karet jedním z hráčů se hodnota vrátí na 0
6. *suitToChange* – po použití svrška se sem ukládá informace o tom, na jakou barvu uživatel svrška změnil
7. *recentAction* – drží si informaci o poslední události ve hře, případně hráči, který ji podnikl
8. *random* – využívá se pro míchání karet při přípravě hry
9. *winner* – poté, co se jeden z hráčů dokáže zbavit karet, se sem uloží pointer na jeho instanci

Konstruktor iniciuje fields a generuje karty.
Hlavní metodou, jejímž zavoláním začíná hra, je *Play()*. Ta se postará o úkony, které je nutné učinit před začátkem hry (míchání karet, nicky hráčů, rozdání karet apod.) a zahájí hru. Využíváno je opět nekonečné metody while, která končí, až když jeden z hráčů má jisté vítězství.
Tato metoda využívá několika pomocných private metod, které se starají o logiku hry. Jako první se zavolá *DrawBasicUI()*, která do konzole vykreslí uživatelské rozhraní.
Další na řadě je *PlayerManagement()* starající se o zavolání správné metody a předání správných dat hráči, který je na řadě. Jde o potenciálně nejsložitější metodu aplikace, ale opatřil jsem ji množstvím komentářů, které by měly její fungování ujasnit. Ještě zdůrazním, že pro pochopení celkové logiky aplikace je nutné na tuto metodu nahlížet ve spojení s metodami třídy hráč, které jsou odsud volány, a také public metodami na *gamePlay* starajícími se o data získaná zpět od hráče (ty jsou řazené až po pomocných private metodách).
Třetí na řadě je kontrola, zda hráč svým posledním tahem již nevyhrál. Pomocná private metoda *CheckWinningConditions()* zkontroluje, zda hráč drží nějaké karty, a pokud ne, vrátí true. V tom případě následuje kontrola, zda druhý hráč drží srdcovou sedmičku a může ji legálně využít, aby vítězného soupeře vrátil do hry. Pokud ne, je příkazem break přerušen cyklus a následuje oslavný text vítězi. V opačném případě hra pokračuje a soupeř je v příštím tahu nucen využít svou srdcovou sedmičku. Hra následně pokračuje běžným způsobem dál.

Závěr

PrsiCardGame dle mého názoru dokáže udržovat naplňování pravidel hry prší a umožňuje vše, co umožňuje fyzická verze hry, včetně možnosti využít přebíjení es, nebo jej zakázat. Uživatelské rozhraní je omezené možnostmi konzole. Spokojil jsem se se změnou barvy pozadí na DarkGreen, tedy barvy, která připomíná stůl v kasinu, pro lepší čitelnost na tomto pozadí jsem písmo nastavil na zelené.
Co by se dalo zlepšit? Přestože je konzole omezující, jistých vylepšení by se dosáhnout dalo. Asi nejméně spokojený jsem s uživatelským rozhraním a jeho vykreslováním. Každá třída si vykresluje svoji trošku a ač to v tomto základním podání funguje, jsem si vědom toho, že by bylo lepší karty v ruce vykreslovat vedle sebe (alespoň po čtyřech) nebo obarvit znaky na kartách dle reálných barev, což by mohlo pomoci hráči lépe s v kartách zorientovat.