Dokumentace zápočtového programu pro Programování v C++

Michal Tichý

Školní rok 2020/2021

Contents

1 Anotace

Zápočtový program měl za cíl vytvoření vlastní verze hry Pazaak, která se poprvé objevila ve hře Star Wars: Knights of the Old Republic. Zápočtový program tohoto dosáhl použitím jazyka C++ a multiplatformních knihoven SFML pro grafiku a Cereal pro serializaci dat.

2 Sestavení

2.1 Windows

Pokud máte nainstalované Visual Studio s nástroji pro vývoj C++ projektů, tak otevřete solution soubor v nejvyšším adresáři a sestavte projekt standardně. Důležité je, abyste sestavili projekt do 64 bitové aplikace.

Pokud Visual Studio nemáte, tak si stáhněte "Build tools for Visual studio" zde pod sekcí "Tools for Visual Studio" a instalujte si nástroje pro sestavení C++ aplikace, "C++ Build Tools". Nainstaluje te tak MSVC i MSBuild. Následně stačí na příkazové řádce spustit příkaz
msbuild Pazaak.sln /t:Build /p:Configuration="Debug/Release" /property:Platform="x64".

Pokud se *msbuild* nepřidal do PATH, tak asi nejjednoduší je spustit *msbuild.exe* přímo. Měl by se nacházet v
C:\"Program Files (x86)"\"Microsoft Visual Studio"\2019\BuildTools\MSBuild\Current\Bin\msbuild.exe

Aplikace se sestrojí do adresáře x64\(Debug/Release).

2.2 Linux

Nainstalujte CMake a C++ překladač, doporučuji g++. Následně v nejvyšším adresáři spustě *setup.sh*. Toto sestaví aplikaci. Program následně můžete spustit spuštěním *run.sh*.

3 Uživatelská část

3.1 Úvod

Pazaak je varianta karetní hry blackjack pro dva hráče. Cílem každého z hráčů je dosáhnout pomocí karet, které si tahají z balíčku, lepší hodnoty než soupeř, která je ale také stále pod limitou hodnoty. K dosáhnutí lepší hodnoty jim také pomáhají speciální karty, které drží v ruce, a které mohou při svém tahu použít.

3.2 Popis hry Pazaak

3.2.1 Průběh hry

Hru Pazaak vyhrává ten hráč, který jako první vyhrál tři kola. Hráči se střídají, v tom kdo je první na tahu.

Na začátku každého kola první hráč vytáhne ze společného balíčku zelenou kartu, která má hodnotu od jedné do deseti, a položí ji na stůl před sebe. Hodnota této karty se přičítává k jeho celkové hodnotě. První hráč má poté možnost použít jednu kartu ze své ruky. Jeho celková hodnota je eventuálně upravena dle efektu použité karty.

Následně se hráč rozhodne, zda pouze ukončí svůj tah a bude pokračovat hrát, nebo zda bude „stát“, to je přestat s taháním dalších karet z balíčku a tak zastavit součet svých karet na aktuální hodnotě. Následně je na tahu protihráč.

Druhý hráč provede to samé. Vytáhne si zelenou kartu z balíčku, položí ji na stůl, potenciálně zahraje jednu kartu z ruky a nakonec se rozhodne zda stojí, nebo bude pokračovat v kole. Pokud první hráč již nestojí, tak je opět na tahu. Takto hra pokračuje dokud oba hráči nestojí, nebo některý z hráčů na konci svého tahu překročil součtem svých karet hodnotu dvacet, nebo hráč má před sebou již devět karet a nemůže táhnout dál.

Hráč ihned prohrává kolo, pokud překročil limitní hodnotu dvacet, tedy po skončení jeho tahu je hodnota součtu jeho vytažených karet vyšší než dvacet, a nebo pokud na konci jeho tahu má před sebou na stole devět karet a tedy nemůže brát další z balíčku. Jinak na konci kola zvítězí hráč, jehož hodnota součtu karet před ním je svým součtem blíže hodnotě dvacet. V případě rovnosti hodnot obou hráčů končí kolo remízou.

Počet vyhraných kol je reprezentován žlutými žárovkami na krajích obrazovky. Červený indikátor ukazuje, který z hráčů je zrovna na tahu, přičemž u ubou těchto indikátorů jsou hráčovi indikátoři na levé straně.

3.2.2 Karty

- **Zelené karty** mají hodnoty 1 – 10. Společný balík je tvořen zelenými kartami.
- **Modré karty** mají hodnotu 1 - 6. Hráč je může mít v ruce. Po zahrání je jejich hodnota přidána k celkovému součtu.
- **Červené karty** mají hodnotu -1 - -6. Tyto karty fungují jako opak modrých karet. Po zahrání je jejich hodnota přidána k celkovému součtu.
- **Červenomodré karty** mají původně hodnoty 1 - 6, avšak jejich hodnotu může hráč před odehráním otočit na -1 - -6. Po zahrání je jejich hodnota přidána, nebo odeprána, k celkovému součtu.

- **Žlutá karta D** zdvojnásobí hodnotu předchozí karty.
- **Žluté karty 2&4** a **3&6** znegují hodnotu všech karet s absolutní hodnotou rovnou jedné z hodnot na kartě. Například po odeehrání karty **2&4** se karty s hodnotou 2 a 4 změní na -2 a -4 a karty s hodnotou -2 a -4 se změní na 2 a 4.
- **Karta Tiebreaker** má hodnotu 1, či -1, tedy funguje stejně jako červenomodrá karta, avšak pokud by kolo končilo remízou, tak hráč, který zahrál **Tiebreaker** kartu, vyhrává. Pokud oba hráči odehráli **Tiebreaker** kartu, tak kolo stále končí remízou.

3.2.3 Kampaň

V režimu kampaňe je přidáno pár složek ke hře. První z nich je obchod, v kterém si hráč může za své kredity koupit přístup ke kartám. Dostupné karty jsou na levé straně obrazovky. Hráč kartz přidává do svého balíčku, který je zobrazen napravo. Cena karet je zobrazena pokud se myš pohybuje nad myší. Hráč může přístup ke kartě také prodat zpět a vybrat si jiné karty.

Následně hráč má možnost si vsadit na následující hru. Pokud hráč zvítězí, tak získá dvojnásobek vsazených kreditů, pokud prohraje, tak vsazené kredity ztrácí.

Poté si hráč vybírá, které karty bude mít v ruce. Z karet, ke kterým má hráč přístup, zobrazeny nalevo, si hráč vybere deset karet. Z těchto deseti karet se následně vybere náhodně čtyři karty, které bude mít na začátku hry v ruce. Tento výběr karet je k dispozici i pokud hráč hraje rychlou hru, v tom případě však má hráč na výběr všechny karty ve hře.

Progres v kampani se ukládá po každé hře. Pokud je kampaň zdolána, tak je poslední save se nachází před poslední hrou. Program má pouze jeden save, který je tedy při každém startu nové kampaně přepsán.

4 Programátorská část

4.1 Bežný průběh

Program začíná s třídou *Game*, která načítá všechny důležité zdroje, a následně přechází do hlavní herní smyčky. Tato smyčka se skládá ze získání současného stavu, v kterém se program nachází, a následného updatování tohoto stavu a vykreslení současného stavu na obrazovku.

Každý stav při updatu reaguje na uživatelův vstup, zmáčknutí klávesnic a myši. Na základě tohoto vstupu pak pozmění svůj stav a následně stav vyobrazí své grafické prvky na obrazovku.

Pokud je za potřebí přestoupit do jiného stavu, tak se zavolají příslušné metody *Game* a při dalším cyklu zpracovává uživatelský vstup a zobrazuje se další stav.

Každý stav je rozdělen na grafickou a logickou část, kde uživatelský vstup mění logickou část a následně se grafická část pozmění dle současného stavu logické části.

4.2 Struktura

Popíšu několik z důležitějších tříd, z kterých se program skládá.

4.2.1 Game

Hlavní třídou celého programu je třída *Game*. Tato třída obsahuje ostatní třídy, které jsou použité během celého průběhu programu. Mezi tyto třídy patří například *TextureManager*, *FontManager*, *StatesHandler*, či *Campaign*. Také obsahuje RenderWindow z SFML, které se stará o vykreslení hry na obrazovku.

Game se také stará o ukládání a načítání informací ohledně kampaně. Serializace dat je provedena pomocí knihovny *Cereal*. Data se serializují do xml souboru.

4.2.2 Campaign

Campaign nese sebou informace ohledně hráčova postupu kampaní. To zahrňuje jeho kredity, karty a také kolik soupeřů porazil.

4.2.3 TextureManager a FontManager

Skladují textury a fonty po celý průběh programu. Tyto externí data jsou skladovány pouze jednou v celém programu, jelikož zabírají relativně mnoho prostoru a pokud by byly zkopírovány všude, kde jsou potřeba, tak by se velikost paměti, kterou program využívá několikanásobně zvětšila.

Množství a velikost textur a fontů je dostatečně malé, že všechny soubory zabírají jen několik MB, což je přijatelné množství.

4.2.4 GameStateHandler

GameStateHandler funguje jako zásobník, kde současný stav se nachází na jeho vrchu. *Game* vždy vezme tento vrchní stav a volá jeho metody pro *update* a *draw*.

Jelikož fungeje jako zásobník, tak můžeme přepínat do nového stavu a následně se vrátit k předchozímu. Takto bychom mohly například přidat menu, které se kdykoliv zobrazí a uživatel ho opět může odvolat. Tuto funkcionalitu v programu nevyužívám, ale je přítomna.

4.2.5 GameState

Základní třída pro všechny ostatní herní stavy. Funkce *update* a *draw* se volají při každém cyklu a tedy tvoří dohromady jeden snímek hry. Detalnější popis stavů přijde později.

4.2.6 GuiStyle

GuiStyle je souhrná třída, která obsahuje různé informace ohledně konkrétního stylu. Ostatní GUI objekty (*Button*, *Indicator*, ...) přímají tyto objekty a získávají od nich potřebné informace ohledně jejich vzhledu, například barvu textu, či barvu pozadí.

Běžný GUI objekt nepotřebuje všechny informace, které jeden objekt *GuiStyle* obsahuje, avšak jelikož máme všechny tyto informace pohromadě, tak se různými GUI objekty lépe pracuje.

4.2.7 GameConst

Struktura, která obsahuje mnoho konstant, které se objevují na různých místech v programu, nebo informace ohledně externích dat.

4.3 Game States

Každý herní stav obsahuje funkci *handleInput*, která zpracovává uživatelův vstup a následně mění svůj stav. Dále je také každý stav rozdělen na logickou část, která se stará o korektnosti stavů, a grafická část, která se stará o rozložení a zobrazení grafických prvků.

Jelikož SFML nemá žádný designer, tak všechna data ohledně rozložení a vzhledu jsou naprogramována jako pojmenované konstantní hodnoty. Pokud se grafická část nějak mění při interakci uživatele s programem, tak obsahuje metodu *update*, která si vezme současný stav z logické části herního stavu aa provede odpovídající změny vzhledu.

4.3.1 GameStateMenu

Menu je stav, v kterém začínáme. Grafická část je velmi jednoduchá skládá se z čtyř tlačítek. Pokud je nějaké z tlačítek zmáčknuto, tak se provede příslušná akce, což je přechod do jiného stavu, nebo ukončení aplikace.

Pokud uživatel chce pokračovat v kampani, tak se musí načíst stav kampaně ze souboru. Pokud toto načítání selže, tak se objeví okno, které informuje uživatele, že načtení selhalo.

4.3.2 GameStateShop

Logická část, *Shop* tohoto stavu se skládá z dvou kolekcí karet, přičemž jedna jsou všechny karty dostupné k zakoupení a druhá obsahuje karty, které uživatel

zakoupil. Tato část má dvě metody, které jsou volány v reakci na uživatelský vstup, *buy* a *sell*. Obě metody zkontrolují, zda uživatel danou kartu může prodat, nebo koupit, a následně provedou danou operaci.

Grafická část, *GuiShop*, při každém updatu zkontroluje jaké karty se nacházejí v kolekcích a případně poupraví existující *GuiCard*, odebere existující, nebo přidá nové.

Při přechdou z tohoto stavu je aktualizována *Campaign* s novým množstvím kreditů a novou kolekcí karet, ke kterým má uživatel přístup.

4.3.3 GameStateBet

Tento stav obsahuje pouze talčítka, která slouží k nastavení částky, kterou uživatel chce vsadit na další hru.

Logická část se skládá pouze z objectu třídy *Counter*, která si pamatuje současný vsazený počet kreditů.

Grafická část, *GuiBet*, se skládá pouze z několika tlačítek, které příčítají nebo
odečítají různý počet kreditů k vsazenému množství.

Při přestupu na další stav je objekt kampaně informován o množství vsazených kreditů.

4.3.4 GameStateDeckBuild

V tomto stavu si uživatel vybírá, které karty si vezme do hry. Logická část, *DeckBuild* opět není moc složitá. Skládá se z dvou kolekcí karet, první jsou ty karty, které má uživatel na výběr a druhé jsou karty, které si uživatel vybral.

Grafická část, *GuiDeckBuild*, si opět získá informace z logické části a následně poupraví, které karty jsou zobrazeny v kolekcích.

Při přechodu do dalšího stavu, čímž je již hra samotná, tak tento stav posílá dál uživatelem vybrané karty a také další informace ohledně oponenta, které získá z kampaně.

4.3.5 GameStateMatch

Stav reprezentující hru samotnou je nejsložitější. Grafická část, *GuiMatch*, se opět dotazuje an informace od logické části, *Match*. Grafická část tohoto stavu sice musí spravovat více položek, avšak kromě tohoto faktu není složitější než předchozí grafické části.

Logická část, *Match*, se skládá z několika prvků. Jmenovitě dvou hráčů, *Player*, kteří u sebe mají svoje karty v ruce, *Hand*, a také karty, které mají vyložené na stole, *Table*, a další informace ohledně hráče. Logická část také obsahuje instanci objektu *Deck*, z které hráči tahají karty na stůl.

Při každém tahu je z *Deck* odebrána jedna karta a přidána jednomu z hráčů, *Player*, na stůl. Následně hráč provede tah. Pokud je to lidský hráč, tak se čeká na vstup od uživatele, pokud je to hráč ovládaný počítačem, tak třída *AI* provede tah.

Třída *AI* obsahuje jednoduchý rozhodovací funkci, která se snaží projít všemi možnostmi, který hráč kontrolovaný počítačem má a snaží se vybrat tu nejlepší volbu.

Na konci každého kola jsou oba hráči a také *GameStateMatch* obeznámeni s výsledkem kola a vhodě upraví svůj stav.

Pokud se nacházíme v kampani, tak na konci hry je kampaň informována o výsledku a také je uložen uživatelův pokrok v kampani.