ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA EKONOMICKÁ

Bakalářská práce

Simulační hra pro podporu výuky základů finanční gramotnosti pro střední školy

The Simulation Game for the Support of the Elementary Financial Literacy for High Schools

Jiří Pešík

Plzeň 2010

Prohlašuji, že jsem bakalářskou práci na téma

„Simulační hra pro podporu výuky základů finanční gramotnosti pro střední školy“

vypracoval samostatně pod odborným dohledem vedoucího bakalářské práce za použití pramenů uvedených v přiložené bibliografii.

V Plzni, dne .....

podpis autora

Poděkování

Chtěl bych touto cestou poděkovat vedoucímu bakalářské práce panu RNDr. Mikuláši Gangurovi, Ph.D za cenné rady a připomínky, které mi pomohly při zpracování. Dále bych rád poděkoval panu JUDr. Ing. Davidu Martinčíkovi za významnou pomoc při návrhu ekonomického modelu.

Obsah

0 Úvod

Základy finanční gramotnosti se stávají stále důležitější součástí našich životů. Lidé často stojí před problémem jak nejlépe investovat své peníze, zda si mohou dovolit úvěr či kdy je pro ně nejvýhodnější prodávat či nakupovat určité zboží. Mnohdy se bohužel rozhodnou nesprávně a jejich omyl často mívá vážné následky. Čím více informací však o této problematice každý má, tím vyšší má šanci, že se rozhodne správně.

Důkazem nutnosti orientovat se v základech finanční gramotnosti dokládá i to, že se nyní stává součástí výuky už i na základních školách.

Na pomyslné druhé straně pak stojí ekonomové. Jednání jednotlivých lidí utvářely dějiny našeho světa od pradávna. Ekonomové se snaží co nejpřesněji popsat, podle čeho a jak se lidé vlastně rozhodují. Na svých teoriích pak staví ekonomické a ekonometrické modely. Mnohdy však tyto modely selhávají a ekonomové, tváří v tvář realitě, musejí hledat nová řešení.

Tato bakalářská práce se věnuje oběma těmto problémům. Hráči si při jejím hraní osvojí základní znalosti z oblasti finanční gramotnosti, zatímco administrátoři budou mít možnost analyzovat chování hráčů za pomocí ekonometrických modelů.

Práce je rozdělena na teoretickou a praktickou část.

Teoretická část této bakalářské práce předkládá jednoduchý model ekonomiky s dvěma druhy zboží a několika trhy, který však obsahuje základní principy, kterými se dominantní neoklasická ekonomická teorie snaží vysvětlit fungování celé ekonomiky.

Výstupem praktické části této práce je simulační hra pojmenovaná *Dynamic Stochastic General Equilibrium Game (DSGE Game)*, která je postavená na popsaném modelu. Je určena pro širokou skupinu hráčů, kteří budou v pravidelných intervalech provádět určitá rozhodnutí a tím ovlivňovat jak vlastní bodové hodnocení, tak i makroekonomické veličiny ekonomiky.

Úplný popis systému, který bude mít k dispozici administrátor hry, umožní testovat různé ekonometrické modely a určit, které dokážou nejlépe vysvětlit a předvídat chování hráčů. Tato data jsou jednou z hlavních výhod hry – v reálném světě samozřejmě takto komplexní informace zdaleka nemáme. Administrátor navíc může ovlivňovat chování hráčů změnou mnoha různých parametrů hry.

Asi nejvýznamnějším přínosem hry je zapojení „živých“ hráčů. Tím se odlišuje od množství simulací, které využívají namísto hráčů virtuální (počítačově řízené) subjekty[1] v ekonomice.

Co se týče aspektů finanční gramotnosti, hráči si při hře uvědomí vzácnost zdrojů, nutnost orientovat se ve vývoji cen a naučí se opatrně hospodařit se svou hotovostí. Dále si uvědomí význam peněz v čase, tedy fakt, že peněžní příjem v současnosti má větší hodnotu než nominálně stejný příjem v budoucnosti.

Tato bakalářská práce si klade následující cíle:

- navrhnout ekonomický model hry
- podrobně popsat ekonomické aspekty v hry a použité algoritmy
- navrhnout objektový model pro implementaci hry
- zpracovat a předložit funkční implementaci simulační hry
- sepsat přehlednou dokumentaci pro uživatele a podrobnou programátorskou dokumentaci pro případné další úpravy hry

V první kapitole je popsán základní model hry, principy jeho fungování, možnosti hráčů a administrátora a sledované údaje.

Následující kapitola je zaměřena na popis ekonomických aspektů, které hra obsahuje – například amortizace kapitálového zboží, různé způsoby výpočtu produkce nebo hodnocení hráčů.

Třetí kapitola navazuje na druhou a předkládá popis klíčových algoritmů, jako například určování ceny na trhu, které jsou v modelu obsaženy. Je zmíněna i jejich praktická implementace ve hře.

Programátorská a uživatelská dokumentace jsou součástí příloh práce.

[1] V terminologii těchto aplikací se pro subjekty zpravidla využívá výraz „agenti“, v této práci autor používá výraz „hráči“ ke zdůraznění rozdílu mezi počítačem řízenými a lidskými subjekty.

1 Popis modelu hry

Simulační hra *Dynamic Stochastic General Equilibrium Game (DSGE Game)* je simulací uzavřené ekonomiky resp. autarkie. Autarkii obecně definujeme jako *„stav či životní podmínky jedné osoby, národa nebo geografické oblasti, která je hospodářsky nebo intelektuálně soběstačná, a tedy nezávislá na jiné z hlediska obchodu, znalostí či přežití.“* (Mises, 2006, s. 855) Ekonomiku ve hře si lze představit jako geografickou oblast mající atributy této definice.

Při tvorbě hry bylo třeba dbát na vyvážení dvou protichůdných vlastností. Na jedné straně aproximace skutečné ekonomiky vede ke složitosti uživatelského rozhraní a k velkému množství hráčem zadávaných příkazů, na straně druhé přitažlivost hry pro širokou skupinu hráčů vyžaduje jednoduchost uživatelského rozhraní (a např. zadávání pouze celočíselných hodnot, které usnadňují hráčům výpočty a orientaci ve hře).

Autor se při návrhu a implementaci hry snažil poskytnout administrátorovi co nejvyšší možnost ovlivnit parametry hry, a to i za cenu vyšší složitosti administrátorského rozhraní (to je však kompenzováno dokumentací pro administrátory, která je součástí této práce). V této kapitole jsou zdůrazněny možnosti, které hra administrátorovi poskytuje.

Při návrhu modelu hry bylo postupováno tak, aby byly splněny níže specifikované cíle.

1.1 Cíle hry

Prvním cílem je simulace skutečné ekonomiky. Tato ekonomika se dělí na reálný a peněžní sektor. Reálný sektor můžeme popsat jako sektor produkující zboží, peněžní sektor se v našem modelu omezuje na trh finančních aktiv, kde se střetávají nabídky a poptávky finančních aktiv (a tím dochází ke vzniku úvěrových vztahů).

Dalším cílem hry je studium možnosti použití různých predikčních ekonometrických postupů. Jak již bylo zmíněno v úvodu, hra poskytne kompletní data o celé ekonomice, což umožní v průběhu hry využít ekonometrické modely k predikci následného vývoje ekonomiky. Předpovídat můžeme např. vývoj cen na jednotlivých trzích nebo vývoj hrubého domácího produktu ekonomiky. *„Hrubý domácí produkt (HDP) je tržní hodnota všech finálních statků vyrobených v ekonomice za dané časové období.“* (Mankiw, 2000) Tyto prognózy bude moci následně porovnat se skutečným stavem.

Administrátor hry bude současně moci kdykoli přistupovat k datům z již odehrané hry a provádět testy modely na nich.

Různé možnosti hře poskytuje i existence centrální banky. *„Centrální banka je emisní bankou, má zákonný monopol na emisi peněz.“* (Holman, 2004, s. 57) S existencí centrální banky souvisí pojem monetární (neboli měnové) politiky. *„Pojem monetární politika se vztahuje na stanovení nabídky peněz v hospodářství.“* (Frank, Bernake, 2003, s. 441)

Jelikož v ekonomické teorii i praxi existuje více přístupů k monetární politice, klade si hra za cíl umožnit studium a porovnání vlivu rozdílných přítupů k monetární politice jak na chování jednotlivých hráčů, tak na úrokové míry nebo např. vývoj cen a HDP. Monetární politika bude významně ovlivněna uzavřeností ekonomiky – rozhodnutí administrátora tedy nebude ovlivněno vlivem monetární politiky na měnový kurz.

1.2 Základní principy hry

Hra se hraje v jednotlivých kolech, která představují určité období ve skutečné ekonomice. Délka kola ve smyslu délky období v reálném světě, které uplyne od počátku do konce kola ve hře, je stanovena administrátorem vzhledem k předpokládaným časovým možnostem hráčů (čím kratší kola bude hra mít, tím častěji budou hráči nuceni se hře věnovat) a v průběhu tohoto kola mají hráči možnost zadat příkazy. Na konci každého kola provede hra administrátorem specifikovanými algoritmy analýzu zadaných příkazů, vypočte potřebné údaje (např. tržní ceny na jednotlivých trzích) a provede realizaci jednotlivých příkazů.

V modelu ekonomiky existují dva druhy vyráběných statků – kapitálové a spotřební zboží. Kapitálové zboží slouží jako výrobní faktor, spotřební zboží je spotřebováváno přímo hráči a tato spotřeba je jedním z faktorů pro hodnocení hráčů (viz. dále).

Ke směně mezi hráči dochází na trzích. Trh ve hře funguje ve smyslu obecné ekonomické teorie, tedy *„trh je mechanismus, jehož prostřednictvím se kupující a prodávající střetávají, aby určili cenu zboží a množství, jež se nakoupí a prodá.“* (Samuelson, Nordhaus, 2007, s. 27)

Jako prostředek směny na trzích slouží v ekonomice peníze. Peníze můžeme obecně definovat jako *„jakékoli aktivum, které je všeobecně přijímáno při placení za zboží a služby nebo při úhradě dluhu.“* (Revenda a kolektiv, 1998, s. 22) Peníze v modelu

DSGE můžeme můžeme definovat přesněji jako bankovky centrální banky – mohou být emitovány pouze centrální bankou.

Hráči vstupují na všechny trhy jako výrobci i jako spotřebitelé. Hráči zadávají pro jednotlivé trhy příkazy, které určují, v jakých cenových intervalech bude nabízeno resp. poptáváno určité množství statku obchodovaného na konkrétním trhu. Cenu chápeme jako hodnotu zboží vyjádřenou v penězích (Samuelson, Nordhaus, 2007).

Systém hry je navržen tak, aby ceny působily na hráče ve smyslu ekonomické teorie. *„Ceny koordinují rozhodování výrobců a spotřebitelů na trhu. Vyšší ceny vedou k omezení nákupů spotřebitelů, ale na druhou stranu motivují k větší výrobě. Při nižších cenách naopak dochází k růstu spotřeby, ale k poklesu produkce. Prostřednictvím cen se udržuje rovnováha na trhu.“* (Samuelson, Nordhaus, 2007, s. 27)

Na základě analýzy příkazů hráčů k nabídce a prodeji se spustí algoritmus k hledání tržní rovnováhy.

„K tržní rovnováze dochází při ceně, kdy se poptávané a nabízené množství rovnají. V této rovnováze nemá cena tendenci růst ani klesat. Rovnovážná cena se nazývá cena vyčišťující trh. Tím chceme říci, že nepřebývají už žádné objednávky ani nabídky a poptávající i nabízející jsou uspokojeni.“ (Samuelson, Nordhaus, 2007, s. 54-55)

Ošetřena je i možnost, že nabízené množství se při dané ceně nebude rovnat poptávanému množství. V takovém případě bude část poptávek nebo nabídek neuspokojena. Obecně je tato situace způsobena (kromě vlivu konkrétního použitého algoritmu) využíváním celočíselných hodnot ve hře, která způsobují, že na trzích nedochází k přesnému stavu tržní rovnováhy (např. cena vyčišťující trh by dosáhla hodnoty reálného, a nikoli přirozeného čísla).

1.2.1 Trh spotřebního zboží

Na tomto trhu hráči nabízejí jimi vyrobené zboží a poptávají spotřební zboží ke vlastní spotřebě. Hráči nemusejí nabízet veškeré jimi vyrobené zboží, ale část produkce mohou uskladnit pro případný prodej v dalších kolech. Hráčům není umožněno přímo spotřebovávat jimi vyrobené zboží, to mohou pouze skladovat a prodávat. Všechno zboží tedy musí procházet trhem.

Nákup spotřebního zboží představuje přímo spotřebu zboží, hráči nemohou skladovat nakoupené zboží např. za účelem spekulace na růst ceny (nicméně mohou spekulovat s prodejem jimi vyrobeného zboží).

1.2.2 Trh kapitálového zboží

Kapitálové zboží je výrobním faktorem, který vstupuje do individuální produkční funkce hráče (produkční funkci je věnována podkapitola 1.2.5). Kapitálové zboží je, stejně jako spotřební zboží, výsledkem předchozí výroby, je tedy vstupem i výstupem výrobního procesu.

Obdobně jako u spotřebního zboží, i u kapitálového zboží lze k výrobě využít pouze zboží nakoupené na trhu, nikoli zboží přímo vyrobené hráčem. Obdobně lze kapitálové zboží také skladovat pro prodej v dalších kolech.

U kapitálového zboží, které je zapojeno ve výrobě, dochází k postupnému opotřebování (amortizaci) podle zvoleného způsobu.

U každého hráče tedy musíme, v souvislosti s kapitálovým zbožím, sledovat dva údaje: kapitálové zboží zapojené do výroby jako výrobní faktor a kapitálové zboží, které hráč vyrobil a skladuje.

Nakoupené zboží vstupuje přímo do výroby. Veškeré kapitálové zboží je považováno za homogenní faktor, libovolné dvě jednotky kapitálového zboží tedy poskytnou v jedné produkční funkci stejný přírůstek produkce.

1.2.3 Trh práce

Vedle kapitálového zboží existuje ve hře jako výrobní faktor práce. Práci bude moci každý hráč využít pro svoji vlastní výrobu nebo nabízet na trhu práce. Dále budou moci hráči na trhu práci poptávat a zakoupené množství práce využít k vlastní výrobě. Dojde tím k určitému rozdělení „rolí“, kdy část hráčů vystupuje jako „podnikatelé“, kteří práci poptávají, a „zaměstnanci“, kteří práci nabízejí. Role však nebudou nijak pevně určeny a jsou tvořeny pouze rozhodnutími hráčů. Role se mohou v průběhu hry měnit a hráči mohou vystupovat i v obou rolích současně (např. do určité ceny mohou poptávat práci ostatních hráčů a od určité ceny vlastní práci nabízet).

Práce je taktéž homogenním výrobním faktorem, jedna hodina práce nakupená na trhu přinese v produkční funkci stejný přírůstek výroby jako jedna hodina vlastní práce konkrétního hráče.

Počet hodin, které může hráč v daném kole odpracovat, je omezeno (např. na 24 hodin) – toto omezení se týká součtu práce využitého pro vlastní výrobu a nejvyššího nabízeného množství práce.

Nakoupená práce spolu s vlastní prací hráče okamžitě vstupují do hráčovy produkční funkce v daném kole.

1.2.3.1 Substituční a důchodový efekt

Hráči jsou však hodnoceni i za množství volného času (tj. rozdílu mezi disponibilním časem a počtem odpracovaných hodin), tím jsou motivováni při relativním poklesu cen nabízet méně práce (volný čas jim přinese vyšší bodový zisk než objem spotřebních statků, které je možné za obdrženou mzdu nakoupit).

Oporu tomuto tvrzení v teorii nabízí substituční efekt. Předpokládejme, že v ekonomice rostou reálné mzdy. Pracovník, který chce získat o jednu hodinu volného času více, se musí vzdát vyšší peněžní částky, kterou by získal za onu jednu hodinu práce. Při vyšších mzdách se tedy každá volná hodina stává dražší. Hráč má proto motivaci více pracovat. (Samuelson, Nordhaus, 2007)

Proti substitučnímu efektu může působit důchodový efekt. Ten způsobuje, že při vyšších mzdách si pracovník může dovolit vyšší spotřebu a současně kratší pracovní dobu. Tu tedy zkracuje a roste jeho množství volného času. (Samuelson, Nordhaus, 2007)

Který efekt převáží, závisí na metodice hodnocení volného času.

1.2.4 Trhy finančních aktiv

Dále v ekonomice existují trhy finančních aktiv, kde mohou hráči nabízet část své hotovosti k zapůjčení a další budou hotovost poptávat. Roli tržní ceny na tomto trhu přebírá úroková míra (nebo také úroková sazba), neboť *„úroková sazba představuje cenu, kterou dlužník platí věřiteli za použití peněz v určitém časovém období.“* (Samuelson, Nordhaus, 2007, s.269)

Nabídku na tomto trhu tvoří hráči, kteří se rozhodli část své hotovosti poskytnout ostatním hráčům za určitý úrok (vystupují tedy jako věřitelé).

Poptávku tvoří hráči, kteří využívají úspory jiných hráčů pro nákupy na ostatních trzích a za to platí úrok (vystupují jako dlužníci, kteří chtějí za daných podmínek získat úvěr).

Trhů s finančními aktivy může být v ekonomice více a vzájemně se liší délkou, na kterou jsou finanční prostředky poskytnuty.

Na tomto trhu může kromě hráčů vystupovat i centrální banka, pro niž jsou nabídka a poptávka finančních aktiv nástrojem její monetární politiky (rozšiřování a kontrakce úvěrů), přičemž objem peněžních prostředků, které centrální banka může nabídnout, není nijak omezen – to představuje možnost banky provádět emisi peněz. Centrální banka tímto způsobem ovlivňuje výši úrokové míry. Roli centrální banky bude provádět administrátor hry nebo jím pověřená osoba.

Splácení přijatých půjček bude provádět automaticky systém, splátka jistiny bude provedena na konci doby splatnosti, splátka úroku probíhá pravidelně v každém kole. Způsob výpočtu úroku určí administrátor. Úvěr i úspory jsou po celou dobu úročeny úrokovou mírou platnou v době uzavření úvěru.

1.2.5 Individuální produkční funkce

„Produkční funkce vyjadřuje maximální množství výstupu, které může být vyrobeno při daném množství vstupů. Je definována pro daný stav technologické znalosti.“ (Samuelson, Nordhaus, 2007, s. 108) V modelu hry definuje produkční funkce přesně množství produkce, které je hráčem vyprodukováno (oproti teoretické definici tedy neuvažujeme slovo „maximální“). Hráči totiž přímo nerozhodují o výrobním procesu (např. volbou používaných technologií výroby nebo kvalifikačními požadavky pracovníků), ale pouze o přidělených zdrojích a vyráběném zboží. Nemají tedy možnost ovlivnit efektivitu využití výrobním faktorů. Rozdíl mezi hodnotou produkční funkce a skutečným výsledkem výroby však v reálném světě bývá způsobena právě neefektivním využíváním výrobních faktorů.

Nezávislými proměnnými produkční funkce jsou množství práce a kapitálového zboží, které hráč v daném kole zapojuje do výroby, aktuální kolo (souvisí s technologickou úrovní, viz. dále) a případně náhodný parametr (viz. dále).

Tvar produkční funkce taktéž závisí na rozhodnutí administrátora hry a dále pak určí i vzájemný poměr, v němž je možné oba vyráběné statky ve výrobě substituovat. Hráč si v každém kole zvolí, který druh zboží chce vyrábět, a na tento druh zboží budou využity všechny výrobní faktory daného hráče.

Ekonomiku může ovlivňovat exogenně daný technologický pokrok. Ten je implementován jako parametr produkční funkce. Čím vyšší je číslo kola, které je nezávislou proměnnou produkční funkce, tím vyšší bude za jinak stejných podmínek množství výstupu.

Dále může v produkční funkci vystupovat i náhodný parametr, který způsobí, že skutečný objem výroby za určitých výrobních faktorů a v určitém kole není deterministický.

1.3 Zadávání příkazů

Pro definici nabízeného a poptávaného množství budou hráči zadávat, jaké množství zboží budou hráči nabízet resp. poptávat v určitých cenových intervalech. Počet těchto intervalů bude dán uživatelským rozhraním a to ovlivní administrátor (více intervalů umožní přesnější specifikaci příkazů, na druhou stranu však zvyšují časovou náročnost hry pro hráče – narážíme zde na dilema zmíněné výše).

Bude povoleno zadávání pouze celočíselných hodnot.

1.4 Začátek a průběh hry

Na začátku dostanou všichni hráči stejné množství spotřebního a kapitálového zboží na skladech, stejné množství kapitálového zboží ve výrobě a stejnou hotovost.

V průběhu kola bude moci hráč uskutečnit několik rozhodnutí

- jak využít aktuální výši hotovosti – může za ni nakoupit spotřební či kapitálové zboží, práci nebo ji nabídnout na trhu finančních aktiv a získat tak úrok
- kolik hodin bude pracovat a zda bude pracovat pro sebe, nabídne svou práci na trhu práce či zvolí kombinaci předchozích možností, či zda nebude pracovat vůbec a bude preferovat volný čas
- druh zboží (kapitálové či spotřební), do jehož výroby vloží své výrobní faktory

1.5 Hotovost

Hráč bude mít k dispozici určité množství hotovosti (hotovost není úročena).

Problémem je možnost záporného stavu hotovosti, protože hráč např. může nakoupit příliš mnoho zboží a zároveň získat malé příjmy z práce a/nebo výroby a tím jeho výdaje převýší příjmy a zůstatek hotovosti z předcházejících kol.

Hra nabízí následující způsoby řešení:

- bodová penalizace za záporný stav hotovosti – ta způsobí, že hráč se bude zápornému stavu vyhýbat
- úročení záporné hotovosti – záporný stav hotovosti bude variantou kontokorentního úvěru (tedy úvěru, kdy je povoleno čerpat zůstatek bankovního účtu do mínusu a tento záporný zůstatek je poté úročen), který poskytují banky (úroková míra musí být zvolena tak, aby hráči nepreferovali tento způsob financování výdajů před trhem finančních aktiv)

S dlouhodobou zápornou hotovostí souvisí i problematika možného krachu hráče.

1.6 Údaje sledované u hráče

Následující hodnoty jsou sledovány u každého hráče v každém kole a pro každé kolo jsou postupně ukládány v databázi (i z důvodu dostupnosti dat pro aplikaci ekonometrických modelů):

- množství hotovosti
- množství spotřebního zboží, které uživatel nakoupil (spotřeboval)
- množství spotřebního zboží, které má na skladě (jím vyrobené zboží v předcházejících kolech, které dosud neprodal)
- množství kapitálové zboží zapojené do výroby
- množství kapitálového zboží, které má hráč na skladě (jím vyrobené zboží)
- počet hodin práce, které hráč využil pro vlastní výrobu
- počet hodin práce, které hráč prodal na trhu práce (z této a předcházející položky lze poté snadno dopočíst volný čas, který měl hráč k dispozici)
- počet hodin práce, které hráč nakoupil na trhu práce
- úvěry získané hráčem na trhu finančních aktiv
- úspory, které hráč poskytl na trhu finančních aktiv ostatním hráčům

1.7 Hodnocení hráčů

Hráči budou hodnoceni prostřednictvím bodů. Body budou přidělovány na základě následujících kritérií:

- aktivita v jednotlivých kolech (měřítkem aktivity může být např. přihlášení hráče do systému nebo zadání příkazu ke spotřebě)
- spotřeba spotřebního zboží v jednotlivých kolech (bude diskontovaná v čase podle zadaného diskontního faktoru)
- množství volného času v jednotlivých kolech (bude také diskontováno v čase)
- záporné peněžní zůstatky (minusové body)

Konkrétní vzorec výpočtu bodů bude zadávat administrátor na začátku hry (případně jej bude moci v průběhu hry změnit) a všichni hráči budou o tomto vzorci informováni. Vzorce hodnocení hráčů mohou být odvozeny např. z ekonomických funkcí užitku a varianty, které hra nabízí, jsou popsány v následující kapitole.

Hráči mohou mít průběžný přehled o svém absolutním hodnocení i o relativní hodnocení ve srovnání s ostatními hráči.

2 Ekonomické aspekty hry

Tato kapitola představuje popis aspektů ekonomického modelu hry, které vychází z ekonomické teorie, a na nich založeného ekonometrického modelu hry. Je zde popsána Cobb-Douglasova produkční funkce hry, která určuje, kolik zboží každý hráč se svými zdroji vyrobí, funkce pro hodnocení hráčů a popis amortizace kapitálu.

Tento ekonomický model je však z hlediska hry naprosto nezávazný, protože administrátor si do rozhraní hry může velmi snadno dosadit vlastní produkční a hodnotící funkce, které mohou mít odlišný teoretický základ.

2.1 Cobb-Douglasova produkční funkce

V modelu hry je Cobb-Douglasova funkce použita jako individuální produkční funkce, tedy funkce určující velikost výroby pro jednotlivé subjekty v ekonomice. Na vzniku této funkce se podíleli americký ekonom Paul Douglas a matematik Charles Cobb ve 20. letech 20. století. Tato funkce vychází z Douglasových analýz empirických dat americké ekonomiky. (Holman, 2004)

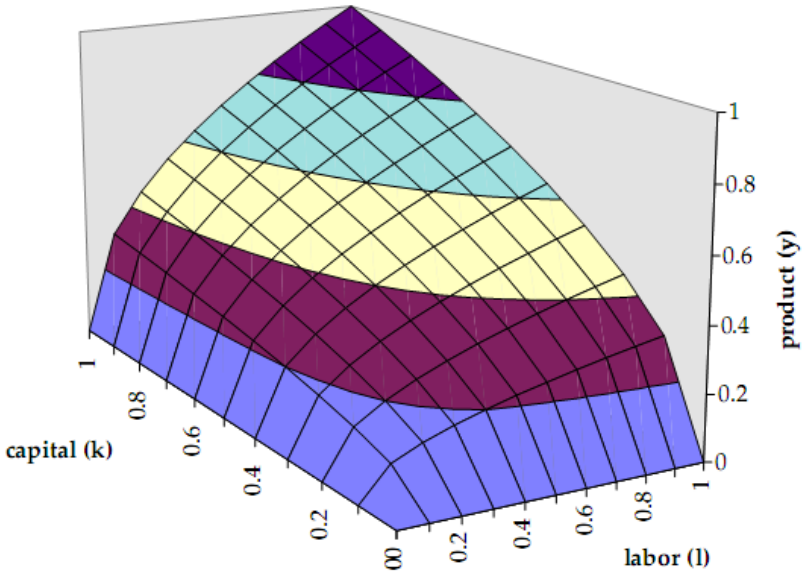
Podstatnou vlastností této funkce jsou konstantní výnosy z rozsahu. *„Konstantní výnosy z rozsahu označují situaci, kdy změna všech vstupů povede k proporcionální změně výstupu. Pokud se například práce, půda, kapitál a jiné vstupy zdvojnásobí, potom se při konstantních výnosech z rozsahu zdvojnásobí i výstup.“* (Samuelson, Nordhaus, 2007, s.111)

Cobb-Douglasova produkční funkce má obecný tvar:

$$Y = A * K^{\alpha} * L^{(1-\alpha)} \tag{3.1}$$

kde: Y ... domácí produkt
A ... konstanta určující produktivitu dostupné technologie
K ... množství kapitálu ve výrobě
L ... množství práce ve výrobě
α ... konstanta z intevalu <0;1>, definuje podíl kapitálových důchodů na produktu
$(1 - \alpha)$ ... podíl pracovních důchodů na domácím produktu

Průběh takto definované funkce je zobrazen na následujícím obrázku.



Obrázek 1 Průběh Cobb-Douglasovy produkční funkce – zdroj: (Doepke a kolektiv, 1999, s. 10)

V modelu uvažujeme exogenní technologický pokrok, tedy technologický pokrok, který je pevně daný rozhodnutím administrátora a není ovlivněn ostatními parametry modelu. Technologický pokrok záleží na aktuálním kole, ve kterém se hra nachází.

Do funkce tedy vstupuje číslo akutálního kola jako další parametr. Technologický pokrok zvyšuje efektivitu práce i kapitálu rovnoměrně. Produkční funkce ve hře má tento tvar:

$$Y_t = A^t * K_t^{\alpha} * L_t^{(1-\alpha)} \tag{3.2}$$

kde: t ... číslo aktuálního kola
ostatní značení zůstává stejné jako u rovnice (3.1)

2.2 Funkce pro hodnocení hráčů (užitková funkce)

Jako základ funkce pro hodnocení hráčů je použita užitková funkce s konstantní elasticitou substituce (CES funkce). Pro účely hry je tato funkce upravena o penalizaci za záporný peněžní zůstatek a neaktivitu hráčů. Tyto položky však nemají vliv na teoretickou funkci v modelu.

Obecný tvar funkce s konstantní elasticitou substituce lze definovat (Henderson, Quandt, 1980):

$$q = A[\alpha * x_1^{-\rho} + (1-\alpha) * x_2^{-\rho}]^{\frac{-1}{\rho}} \tag{3.3}$$

kde: A ... parametr funkce (např. diskontní faktor nebo technologický pokrok), $A > 0$
α ... parametr funkce, platí $0 < \alpha < 1$

Tato funkce má konstantní výnosy z rozsahu, což dokazuje následující rovnost:

$$A[\alpha * (tx_1)^{-\rho} + (1-\alpha) * (tx_2)^{-\rho}]^{\frac{-1}{\rho}} = tA[\alpha * (x_1)^{-\rho} + (1-\alpha) * (x_2)^{-\rho}]^{\frac{-1}{\rho}} \tag{3.4}$$

Cobb-Douglasova produkční funkce ve tvaru (3.1) této charakteristice odpovídá. V případě nahrazení parametru α a $(1 - \alpha)$ za parametry α a β, pro které platí $\alpha + \beta \neq 1$ však nejsou zaručeny konstatní výnosy z rozsahu (Henderson, Quandt, 1980).

Užitková funkce v modelu, která má konstantní elasticitu substituce, má tvar:

$$u_t(c_t, l_t) = \beta^t (\alpha * c_t^{\ \rho} + (1-\alpha) * (D - l_t)^{\rho})^{1/\rho}$$

kde: u_t ... užitek
c_t ... spotřeba
D ... konstanta představující disponibilní množství času (např. 24 hodin)
l_t ... množství práce
β^t ... diskontní faktor
α ... váha spotřeby
ρ ... parametr

Elasticitu substituce určíme z následujícího vztahu:

$$s = \frac{1}{1-\rho}$$

kde: s ... elasticita substituce

Další možností pro užitkovou funkci je následující funkce:

$$u_t(c_t, l_t) = A\left(\frac{c_t^{\ 1-\theta}}{1-\theta} - \chi \frac{(l_t / D)^{1+\eta}}{1+\eta}\right)$$

kde χ ... měřítková konstanta umožňující srovnání užitku ze spotřeby a z volného času
$1/\theta$ elasticita intertemporální substituce
$1/\eta$ elasticita nabídky práce

Tato funkce je však mnohem náročnější z hlediska rozhodování hráčů.

2.3 Ekonomický model hry

Vlastnictví firem je jasně definované vlastnictvím kapitálového zboží a skladovaných zásob konkrétním hráčem. Každý z hráčů má přiřazenou produkční funkci. Struktura ekonomiky není rozdělena na výrobce a spotřebitele, jako je tomu běžně u neoklasických ekonomických modelů, ale každý hráč může být současně spotřebitelem i výrobcem.

Maximalizace zisku výrobců zde není definována jako předpoklad modelu, ale je dána snahou hráčů maximalizovat svůj užitek. Maximalizace zisku totiž umožní maximalizaci spotřeby, za kterou jsou hráči hodnoceni.

Uvažujeme hráče, který maximalizuje svůj užitek v nekonečném časovém horizontu. Tento předpoklad je v růstových ekonomických modelech oblíbený. *„Jde ale o sporný předpoklad, protože člověk je smrtelná bytost Můžeme však argumentovat tím, že lidé se nestarají pouze o svůj vlastní užitek, ale i o užitek svých dětí, a ty se zase snaží o blaho svých potomků atd. Navíc je-li život člověka dostatečně dlouhý (a jeho diskontní faktor poměrně velký), může nekonečný horizont sloužit jako dobrá první aproximace.“* (Čihák a kolektiv, 2000, s. 39)

Rozhodování hráče ve smyslu těchto předpokladů lze formálně vyjádřit takto:

$$\sum_{t=1}^{\infty} \beta^t * U_t \longrightarrow \max \tag{3.5}$$

kde: t ... číslo kola

U_t ... užitek hráče v kole t

Užitkovou funkci pro každé kolo definuje vztah:

$$U_t = C_t^{\kappa} * H_t^{\lambda} \tag{3.6}$$

kde: U_t ... užitek v kole t

β ... diskontní faktor, který definuje vliv budoucí spotřeby na současné rozhodování, platí $0 < \beta < 1$

C_t ... spotřeba zboží hráčem v kole t

H_t ... množství volného času v kole t

κ ... koeficient spotřeby zboží v užitku hráče
λ ... koeficient volného času v užitku hráče

Množství volného času, které má hráč v kole t k dispozici, lze vyjádřit vztahem:

$$H_t = D - LS_t - LMS_t \tag{3.7}$$

kde: D ... množství času, které má hráč k dispozici
LS_t ... množství jednotek práce, které hráč v kole t prodal na trhu práce
LMS_t ... množství jednotek práce, které hráč v kole t odpracoval ve vlastní výrobě

Pro analýzu příjmů je potřeba nejprve definovat, které trhy finančních aktiv v ekonomice fungují. Nechť existuje jeden trh finančních aktiv, kde jsou úvěry postkytovány na dobu dvou období. Analyzujeme-li příjmy v kole t, pak hráč obdrží úrok z úspor zapůjčených v kole $t-1$a dále pak úrok z úspor zapůjčených v kole $t-2$ spolu s celou jistinou, kterou hráč v kole $t-2$ zapůjčil.

Příjmy hráče jsou tvořeny prostřednictvím prodeje zboží, práce a získání úvěru. Pro příjmy hráče tedy platí následující rovnost:

$$LS_t * w_t + CS_t * CP_t + KS_t * KP_t + SS_{t-1} * i_{t-1} + SS_{t-2} * (1 + i_{t-2}) + DS_t + M_{t-1} \tag{3.8}$$

kde: w_t ... nominální mzdová sazba v kole t
CS_t ... realizované nabízené množství (neboli nabídka ex post) v kole t
CP_t ... nominální jednotková tržní cena na trhu spotřebního zboží v kole t
KS_t ... realizovaná nabídka kapitálového zboží v kole t
KP_t ... nominální jednotková tržní cena na trhu kapitálového zboží v kole t
SS_{t-1} ... realizované nabízené množství na trhu finančních aktiv v kole $t-1$
i_{t-1} ... úroková míra na trhu finančních aktiv v kole $t-1$
DS_t ... realizovaná poptávka po úsporách v kole t (neboli získaný úvěr)
M_{t-1} ... nominální hodnota peněžního zůstatku hráče z předchozího kola

Prodané kapitálové a spotřební zboží je odečteno ze skladových zásob hráče. U každého hráče jsou sledovány jeho skladové zásoby. Kladný přírůstek zásob zajišťuje výroby zboží. Vyrobené zboží v kole t ale může hráč prodat až v kole $t+1$. Pro dané kolo tedy dochází ke změně skladových zásob pouze v důsledku prodeje zboží. Můžeme tedy provést následující formální úpravu rovnice:

$$LS_t * w_t - \Delta CW_t * CP_t - \Delta KW_t * KP_t + SS_{t-1} * i_{t-1} + SS_{t-2} * (1 + i_{t-2}) + DS_t + M_{t-1}$$

(3.9)

kde: ΔCW_t ... změna skladových zásob spotřebního zboží v průběhu kola t
ΔKW_t ... změna skladových zásob kapitálového zboží v průběhu kola t

Nyní formálně vyjádříme výdaje hráče. Výdaje hráče tvoří realizovaná poptávka (nákupy) spotřebního zboží, kapitálového zboží a práce. Dále je výdajem hráče zapůjčení úspor na trhu finančních aktiv. Poslední položkou je platba úroků z dříve uzavřených úvěrů a vrácení dříve získaných úvěrů.

$$LD_t * w_t + CD_t * CP_t + KD_t * KP_t + SD_{t-1} * i_{t-1} + SD_{t-2} * (1 + i_{t-2}) + SS_t + M_t$$

(3.10)

kde: LD_t ... realizovaná poptávka po práci v kole t
CD_t ... realizovaná poptávka po spotřebním zboží v kole t
KD_t ... realizovaná poptávka po kapitálovém zbožím v kole t
SD_{t-1} ... realizovaná poptávka po úsporách v kole $t-1$
SS_t ... realizovaná nabídka úspor v kole t

Množství spotřebního zboží, které hráč nakoupí, v tomtéž kole vstupuje do hodnotící funkce jako množství spotřebovaného zboží. Dále nakoupené kapitálové zboží ihned vstupuje do hráčovy produkční funkce (zvyšuje množství kapitálového zboží ve výrobě) a zvyšuje tak (za jinak stejných podmínek) množství zboží vyrobené hráčem. Upravená rovnice vypadá takto:

$$LD_t * w_t + C_t * CP_t + \Delta K_t * KP_t + SD_{t-1} * i_{t-1} + SD_{t-2} * (1 + i_{t-2}) + SS_t + M_t$$

(3.11)

kde: ΔK_t ... je změna množství kapitálového zboží ve výrobě v průběhu kola t (hrubé investice)

Definujeme-li ΔK_t jako změnu kapitálového zboží v průběhu kola t, tak tím uvažujeme pouze změnu množství kapitálového zboží v důsledku nákupu kapitálového zboží. Není tím uvažována amortizace kapitálu. K té totiž formálně přistupujeme tak, že do výroby v následujícím kole vstupuje pouze část kapitálového zboží z předchozího kola. Velikost této části definuje amortizační faktor. Formalizace tohoto vztahu vypadá následujícím způsobem:

$$K_t = K_{t-1} * (1 - \delta) + \Delta K_t$$

(3.12)

kde: δ ... amortizační faktor, platí $0 < \delta < 1$

Jelikož může hráč vyrábět dva různé druhy zboží, teoreticky by mohl mít i dvě různé produkční funkce. Jedna by určovala množství vyrobeného zboží při produkci spotřebního zboží, druhá při produkci kapitálového zboží. V modelu hry je uvažována jedna produkční funkce, což znamená dokonalé nahrazování spotřebního a kapitálového zboží při výrobě (dokonalou výrobní substituci). Produkční funkce vychází z Cobb-Douglasovy produkční funkce, která je vysvětlena v kapitole 3.1.

Zboží, které je vyprodukováno, navýší skladované množství zboží hráče v příštím kole.

$$CW_{t+1} + KW_{t+1} = A^t * K_t^{\alpha} * L_t^{(1-\alpha)} + CW_t + KW_t$$

(3.13)

Vzhledem k diskontnímu faktoru uvažujeme, že držení zásob není optimání. Potom získáváme následující rovnost:

$$CW_{t+1} + KW_{t+1} = A^t * K_t^{\alpha} * L_t^{(1-\alpha)}$$

(3.14)

kde: A ... technologický pokrok
K_t ... množství kapitálového zboží ve výrobě v kole
Y ... domácí produkt

2.3.1 Rovnováha na trzích

Nyní zavedeme podmínky rovnováhy trhů.

Tržní nabízené množství je součtem individuálních nabízených množství všech hráčů při dané ceně, tržní poptávané množství je součtem individuálních poptávaných množství všech hráčů. Trhy se obecně nacházejí v rovnováze, pokud se tržní nabízené množtsví rovná tržnímu poptávanémnu množství.,

Trh práce se nachází v rovnováze, pokud se nabízená práce rovná poptávané práci, tedy:

$$\sum LS_t = \sum LD_t$$

(3.15)

Trh spotřebního zboží se nachází v rovnováze, pokud se tržní nabízené množství zboží rovná tržnímu poptávanému množství. Tržní poptávané množtsví při tržní ceně se rovná součtu individuální spotřeby všech subjektů v ekonomice.

$$\sum CS_t = \sum CD_t = \sum C_t = -\sum \Delta CW_t$$

(3.16)

Tržní nabízené množství při tržní ceně na trhu kapitálového zboží se rovná součtu úbytku skladových zásob kapitálového zboží všech hráčů. Dále pak součet všech realizovaných poptávek se rovná součtu kladných přírůstků kapitálového zboží v produkčních funkcích všech hráčů.

$$\sum KS_t = \sum KD_t = \sum \Delta K_t = -\sum \Delta KW_t$$

(3.17)

Dále budeme uvažovat ekonomiku skládající se z racionálně jednajících subjektů, což je běžný předpoklad neoklasických růstových modelů (Čihák a kolektiv, 2000). Pro reprezentativního racionálního agenta platí následující rovnosti:

$$LS_t = LD_t$$

$$CS_t = CD_t = C_t = -\Delta CW_t$$

$$\Delta K_t = -\Delta KW_t$$

2.3.2 Výsledná omezení

Vynásobíme-li předchozí rovnosti tržní cenou, pak vidíme, že objem finančních prostředků vynaložený na nákupy se rovná objemu finančních prostředků vynaložených na prodeje. Množství finančních prostředků, které se nachází v ekonomice, tj. součet peněžních zůstatků všech hráčů, se pak mezi jednotlivými koly nemění. Formálně tento závěr vyjádříme vztahem:

$$\sum M_{t-1} = \sum M_t$$

(3.18)

Vrátíme-li se k předpokladu racionálně jednajícího agenta, tak pro reprezentativního agenta platí následující rovnost:

$$M_{t-1} = M_t$$

Abychom zjistili celkový objem vyrobeného zboží v ekonomice, je třeba zjistit celkový odpracovaný čas za dané kolo. Vyjdeme ze vztahu pro individuální množství volného času jednoho hráče (3.7). Abychom zjistili, kolik jednotek času bylo v ekonomice celkem odpracováno, nejprve vyjádříme, kolik volného času v ekonomice měli všichni hráči v daném kole k dispozici:

$$\sum H_t = \sum (D - LS_t - LMS_t)$$

(3.19)

V ekonomice je tedy celkem odpracováno hodin:

$$\sum (H_t - D) = \sum (-LS_t - LMS_t)$$

Neboli:

$$\sum (D - H_t) = \sum (LS_t + LMS_t)$$

(3.20)

Vrátíme-li se k předpokladu racionálně jednajícího jedince a vyjdeme-li z formálních podmínek tržní rovnováhy, pak množství času odpracované jedním hráčem je rovno:

$$D - H_t = LS_t + LMS_t$$

(3.21)

Dále množství kapitálového zboží, které má subjekt k dispozici v kole t, vyjádříme jako:

$$K_t = K_{t-1} * (1 - \delta) + \Delta K_t$$

(3.22)

Můžeme tedy provést následující úpravu rovnice vyjadřující produkt:

$$Y_t = CW_{t+1} + CK_{t+1} = A^t * (K_{t-1} * (1 - \delta) + \Delta K_t)^{\alpha} * (D - H_t)^{(1-\alpha)}$$

(3.23)

Toto ještě není konečný tvar rovnice, protože můžeme zavést podmínku neoptimálního držení zásob racionálními subjekty:

$$-C_{t+1} - \Delta K_{t+1} = A^t * (K_{t-1} * (1 - \delta) + \Delta K_t)^{\alpha} * (D - H_t)^{(1-\alpha)}$$

(3.24)

Úpravou vztahu můžeme zjistit omezení spotřeby v kole t:

$$C_t = A^{t-1} * (K_{t-2} * (1-\delta) + \Delta K_{t-1})^{\alpha} * (D - H_{t-1})^{(1-\alpha)} - \Delta K_t$$

(3.25)

Množství kapitálového zboží ve výrobě racionálního subjektu definujeme jako část kapitálového zboží z minulého kola a změna stavu kapitálového zboží v důsledku nákupu na trhu v aktuálním kole. To formálně vyjadřeuje rovnice (3.22). Provedeme její úpravu a vyjádříme si velikost změny kapitálu mezi kolem $t-1$ a t:

$$\Delta K_t = K_t - K_{t-1} * (1-\delta) = K_t - K_{t-1} + \delta K_{t-1}$$

(3.26)

A provedeme úpravu produkční funkce:

$$C_t = A^{t-1} * (K_{t-2} * (1-\delta) + K_{t-1} - K_{t-2} * (1-\delta))^{\alpha} * (D - H_{t-1})^{(1-\alpha)} - K_t + K_{t-1} * (1-\delta)$$

Neboli:

$$C_t = A^{t-1} * K_{t-1}{}^{\alpha} * (D - H_{t-1})^{(1-\alpha)} - K_t - K_{t-1} * (1-\delta)$$

(3.27)

2.3.3 Obecné řešení

Maximalizujeme užitek v nekonečném časovém horizontu:

$$\max \sum_{t=0}^{\infty} \beta^t u(C_t, L_t)$$

Z toho vyjádříme:

$$C_t = f(K_{t-1}, L_{t-1}) - K_t - K_{t-1}(1-\delta)$$

Neboli:

$$f(K_{t-1}, L_{t-1}) = C_t + K_t - K_{t-1} + \delta K_{t-1}$$

Rovnice podmínky má tedy tvar:

$$L(C_t, L_t, K_t, \lambda_t) = \sum_{t=0}^{\infty} \beta^t u(C_t, L_t) - \sum_{t=0}^{\infty} \beta^t \lambda_t (f(K_{t-1}, L_{t-1}) - K_t + K_{t-1}(1-\delta) - C_t)$$

Určíme si dva členy sum – pro t a t+*1*:

$$\beta^t u(C_t, L_t) - \beta^t \lambda_t (f(K_{t-1}, L_{t-1}) - K_t + K_{t-1}(1-\delta) - C_t) +$$
$$\beta^{t+1} u(C_{t+1}, L_{t+1}) - \beta^{t+1} \lambda_{t+1} (f(K_t, L_t) - K_{t+1} + K_t(1-\delta) - C_{t+1})$$

Provedeme parciální derivace podle jednotlivých proměnných. Parciální derivace podle C_t:

$$\frac{\partial L}{\partial C_t} = \beta^t \frac{\partial u(C_t, L_t)}{\partial C_t} + \beta^t \lambda_t = 0$$

Z toho plyne:

$$\frac{\partial u}{\partial C_t} = -\lambda_t$$

Parciální derivace podle L_t:

$$\frac{\partial L}{\partial L_t} = \beta^t \frac{\partial u(C_t, L_t)}{\partial L_t} - \beta^{t+1} \lambda_{t+1} \frac{\partial f(K_t, L_t)}{\partial L_t} = 0$$

Z toho plyne:

$$\frac{\partial u}{\partial L_t} = \beta \lambda_{t+1} \frac{\partial f}{\partial L_t}$$

Parciální derivace podle K_t:

$$\frac{\partial L_t}{\partial K_t} = \beta^t \lambda_t - \beta^{t+1} \lambda_{t+1} \left(\frac{\partial f(K_t, L_t)}{\partial K_t} + 1 - \delta\right) = 0$$

Z toho plyne:

$$\lambda_t = \beta^t \lambda_{t+1} \left(\frac{\partial f}{\partial K_t} + 1 - \delta\right)$$

Parciální derivace podle λ_t (výsledné rozpočtové omezení):

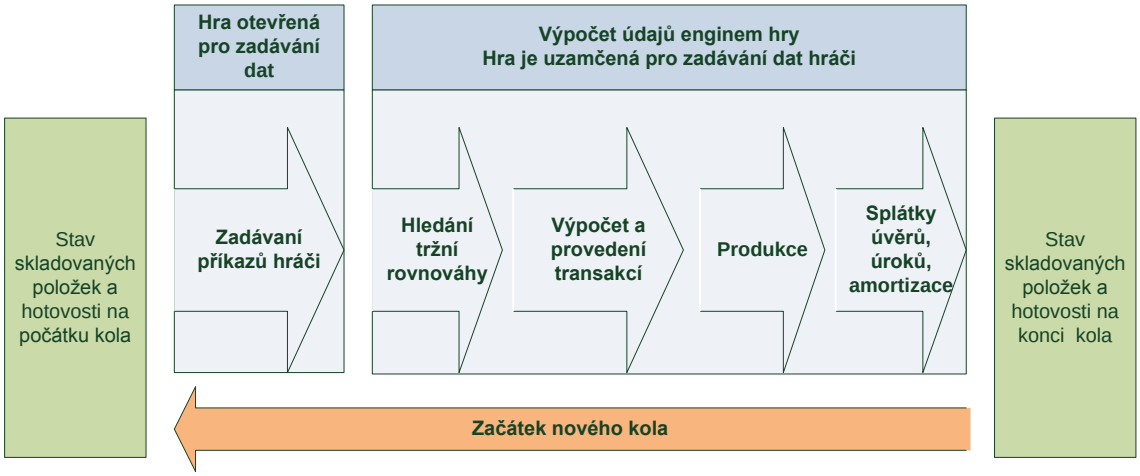
$$\frac{\partial L}{\partial \lambda_t} = -\beta^t (f(K_{t-1}, L_{t-1}) - K_t + K_{t+1}(1-\delta) - C_t) = 0$$

3 Klíčové algoritmy hry

3.1 Plynutí času ve hře

Před výkladem o jednotlivých algoritmech je třeba se věnovat popisu plynutí času ve hře. Jak již bylo řečeno v předcházejících kapitolách, obchodování na trzích probíhá v kolech.

V níže uvedeném schématu je šipkami zobrazen zjednodušený průběh hlavních událostí jednoho kola. Horní dva obdélníky označují, kdy dochází k zadávání dat uživateli a kdy k výpočtům ze zadaných dat.



Obrázek 2 Průběh času ve hře - zdroj: vlastní

Poté, co jsou aktualizovány stavy skladů, hotovosti a bodové hodnocení hráčů, dojde k zahájení nového kola. Konečný stav tohoto kola je rovný počátečnímu stavu následujícího kola.

3.2 Čas ve hře a ve skutečném světě

Pro pochopení principu hry je důležité srovnání plynutí času ve skutečném světě a v herní ekonomice. Vycházíme-li z výše vloženého schématu, pak šipka *„Zadávání příkazů hráči“* reprezentuje ve skutečném světě období v řádu několika dnů (dle rozhodnutí administrátora). Hráči v tomto čase pouze rozhodují, co se bude vyrábět a za kolik se co prodá. Neprobíhají žádné obchodní transakce ani výrobní operace.

Po určitém časovém intervalu je možnost zadávání příkazů ukončena. Následně dojde k operacím, které jsou zachyceny ve výše uvedeném schématu v obdélníku *Výpočet údajů enginem hry*.

Uvažujeme-li čas ve skutečném světě, pak tyto operace trvají několik minut a jsou determinovány pouze časem, za který je herní server schopen provést všechny potřebné výpočty a záznamy dat v databázi.

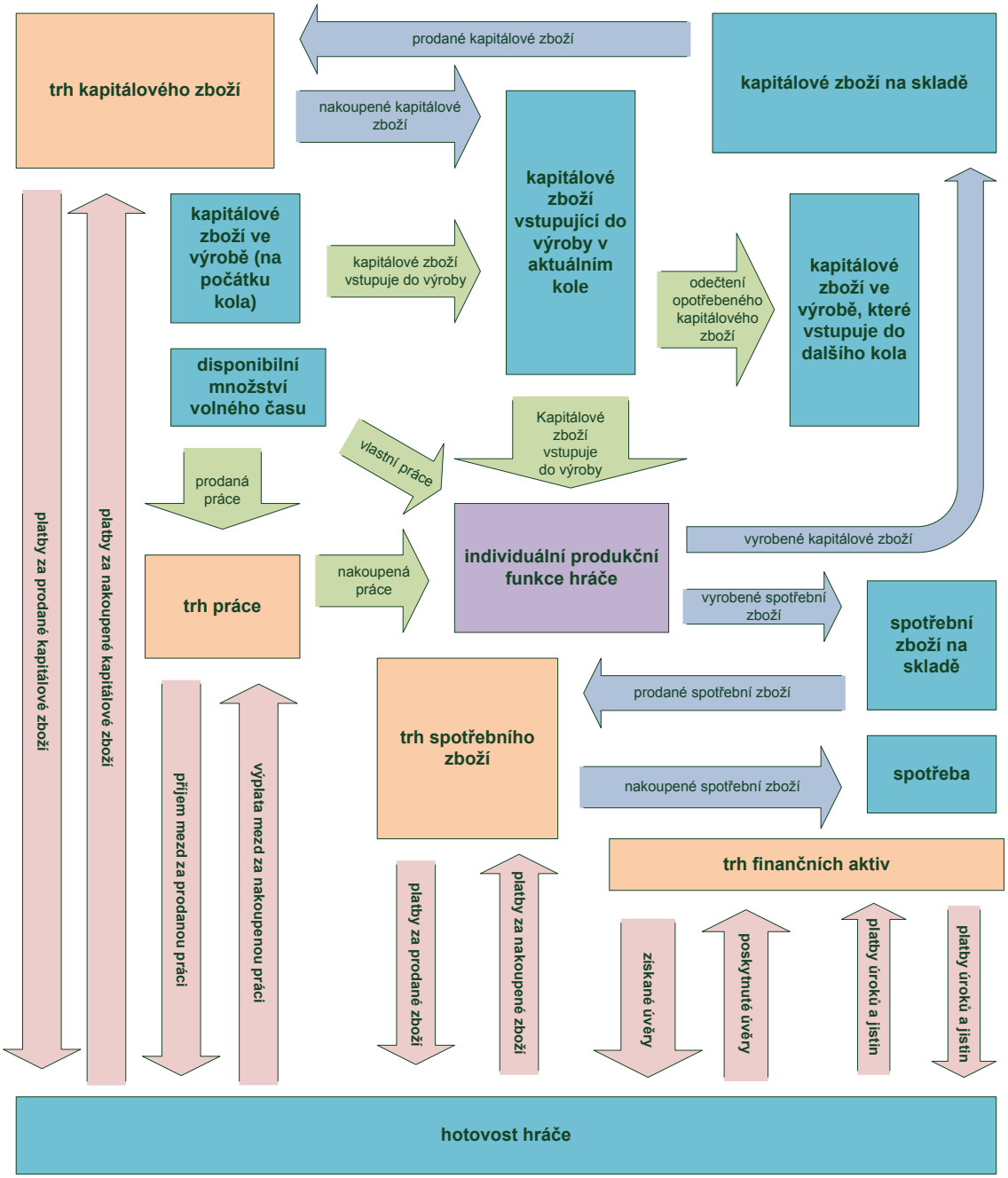
Z pohledu plynutí času ve skutečném světě tedy existuje následující paradox: subjekty se rozhodují, co budou vyrábět a za kolik prodávat po mnohonásobně delší dobu, než probíhá samotné obchodování a výroba.

V modelu, na kterém je hra postavena, je čas považován za diskrétní – dělí se na jednotlivá období (neboli kola), která lze označovat pouze diskrétními hodnotami. Události, ke kterým v ekonomice dojde, lze časově vztahovat k určitému kolu, nijak blíže je však určit nemůžeme. Můžeme tedy např. říci, že v desátém kole prodal určitý hráč deset kusů spotřebního zboží za tržní cenu 200, ale blíže nemůžeme specifikovat, „kdy přesně“ (např. v kolik hodin) k prodeji došlo.

3.3 Koloběh zboží, peněz a výrobních faktorů v modelu hry

Výpočtový proces na konci kola realizuje přesuny položek mezi hráči. K obdobným tokům dochází i mezi subjekty v reálné ekonomice. Přesuny v *DSGE* modelu jsou z velké části realizovány prostřednictvím trhů. Při nákupu na trhu dochází k přesunu hotovosti od hráče ve prospěch ostatních hráčů a současně k toku zboží směrem k hráči nebo navýšení jako produkčních možností. Naopak při prodeji můžeme pozorovat přesun zboží nebo produkčních možností od hráče vůči ostatním subjektům a současně k toku příslušné hotovosti ve prospěch hráče.

Dále je třeba uvažovat platby úroků a jistin z úvěrů, které byly poskytnuty v předcházejících kolech. Jako transakci, která nevstupuje na žádný trh, můžeme chápat práci hráče pro sebe samotného, protože hráč obětuje užitek z volného času a práce vstupuje do hráčovy produkční funkce. Stav položek na skladě je navýšen o produkci, která je výsledkem vyrobeného zboží hráčem. A konečně mezi množstvím kapitálového zboží vloženého do výroby a výsledným stavem kapitálového zboží na konci kola dochází k odečtení části kapitálového zboží vlivem amortizace. Všechny tyto operace jsou graficky zobrazeny na následujícím schématu.



Obrázek 3 Toky peněz, zboží a výrobních faktorů během kola - zdroj: vlastní

Ve schématu jsou tyrkysovou barvou zobrazeny položky evidované u hráče, oranžovou jednotlivé trhy, světle červenou šipkou toky peněz, modrou šipkou toky zboží a zelenou šipkou pohyb práce.

3.4 Systém událostí

Objekty v aplikaci používají pro komunikaci systém událostí a ovladačů, které na tuto událost reagují. Možnost využití událostí pro komunikaci mezi třídami aplikace nabízí

některé objektově orientované jazyky, např. Java nebo C#. PHP tuto možnost standardně nenabízí.

Hra DSGE Game má vlastní jednoduchý systém komunikace pomocí událostí, který se použitím blíží jazykům, které tuto vlastnost podporují. Implementace v DSGE Game je postavena na modelu Davida Fellse (Fells, 2006).

Událost je ve své podstatě způsob, kterým jeden objekt sděluje ostatním, *„že došlo k něčemu zajímavému“* (Puš, 2005). Podstatné u tohoto způsobu komunikace je, že třída tuto informaci neposílá nějakému konkrétnímu příjemci, ale systému řízení událostí. Ten zajišťuje, že zprávu obdrží každá třída, která o ni má zájem. To je důležité pro další rozšiřování aplikace, protože je možné přidávat další třídy a funkce bez zásahů do již naprogramovaných tříd.

Každý objekt, který může generovat události pro ostatní třídy, má vlastní datovou strukturu nazvanou fronta událostí. Jedná se o datovou strukturu typu FIFO. To znamená, že jednotlivé položky jsou v ní tedy řazeny v pořadí, ve kterém byly vloženy.

V této frontě jsou všechny události, které může třída vyvolat. Události ve frontě čekají na vyvolání, ke kterému poté může dojít v proběhu činnosti třídy.

Reakce tříd na události je zajišťována prostřednictvím ovladačů. Na každou událost může být napojené libovolné množství ovladačů. Jsou ukládány do datové struktury nazývané fronta ovladačů. Jedná opět o datovou strukturu typu FIFO. V případě, že vydáme frontě ovladačů příkaz k aktivaci všech zařazených ovladačů, jsou tyto ovladače aktivovány přesně v pořadí, ve kterém byly vloženy.

Ovladač jednak reprezentuje nějakou akci, ke které dojde v důsledku vyvolání události. Je navázán na právě jednu třídu, která je schopna reagovat na událost. V případě, že je událost vyvolána, zajistí ovladač provedení konkrétní činnosti. Provedení činnosti je zajištěno voláním některé z metod třídy, na kterou je ovladač navázán.

Následující schéma zobrazuje princip fungování řízení události. Pro zjednodušení jsou v něm pouze některé objekty z objektového modelu hry. Ve schématu je jeden objekt, který generuje událost – objekt *Časovač*. Ten v pravidelných intervalech vyvolává událost *Konec kola*. Dále jsou ve schématu dva objekty, které budou na událost *Konec kola* reagovat – *Správce trhu* a *Správce produkce*.

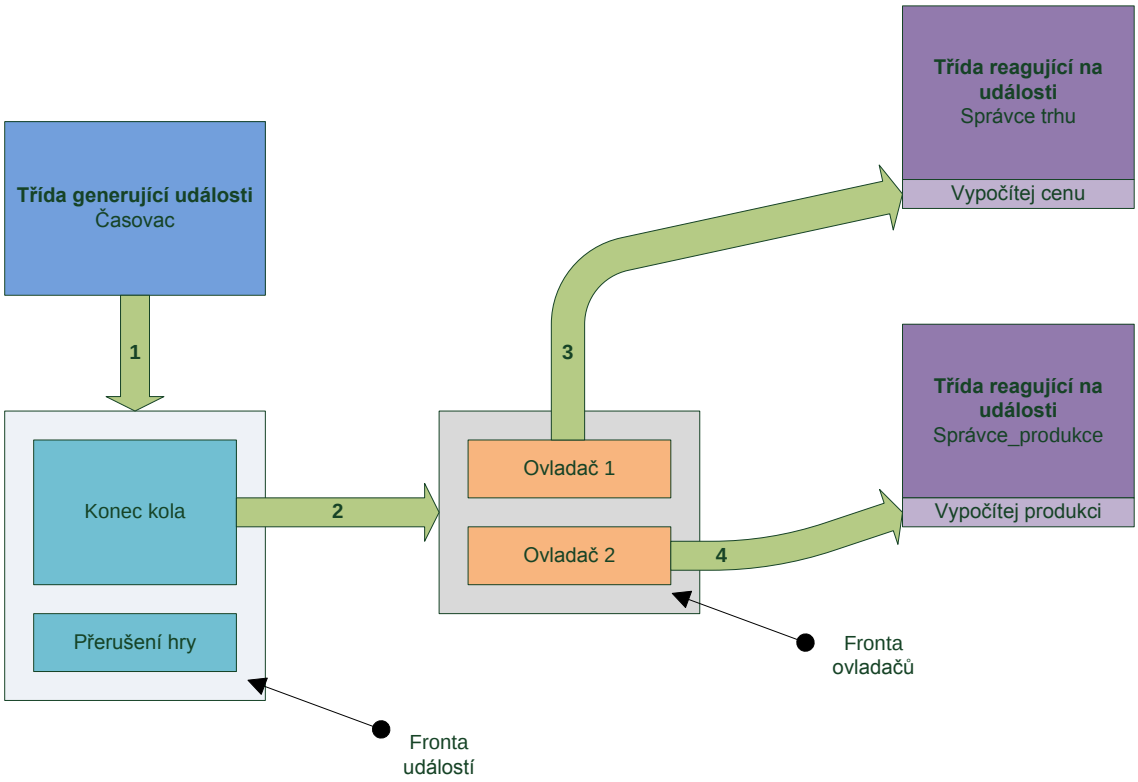
Zelenými šipkami je zobrazena posloupnost akcí, které vzniknou při vyvolání události.

Šipka 1 označuje vyvolání události. V tomto případě objekt *Časovač* obdržel časový signál a na základě jeho analýzy zjistil, že nastal čas k ukončení současného kola a začátku nového. Vyvolává tedy událost *Konec kola*, která mezitím čeká ve frontě událostí, aby informovala ostatní třídy.

Fronta událostí nalezne událost *Konec kola*. Na schématu je fronta ovladačů této události, která obsahuje dva ovladače. Tato událost vydává své frontě ovladačů příkaz, aby aktivovala všechny ovladače, které jsou v ní zařazeny – tento příkaz reprezentuje *šipka 2*.

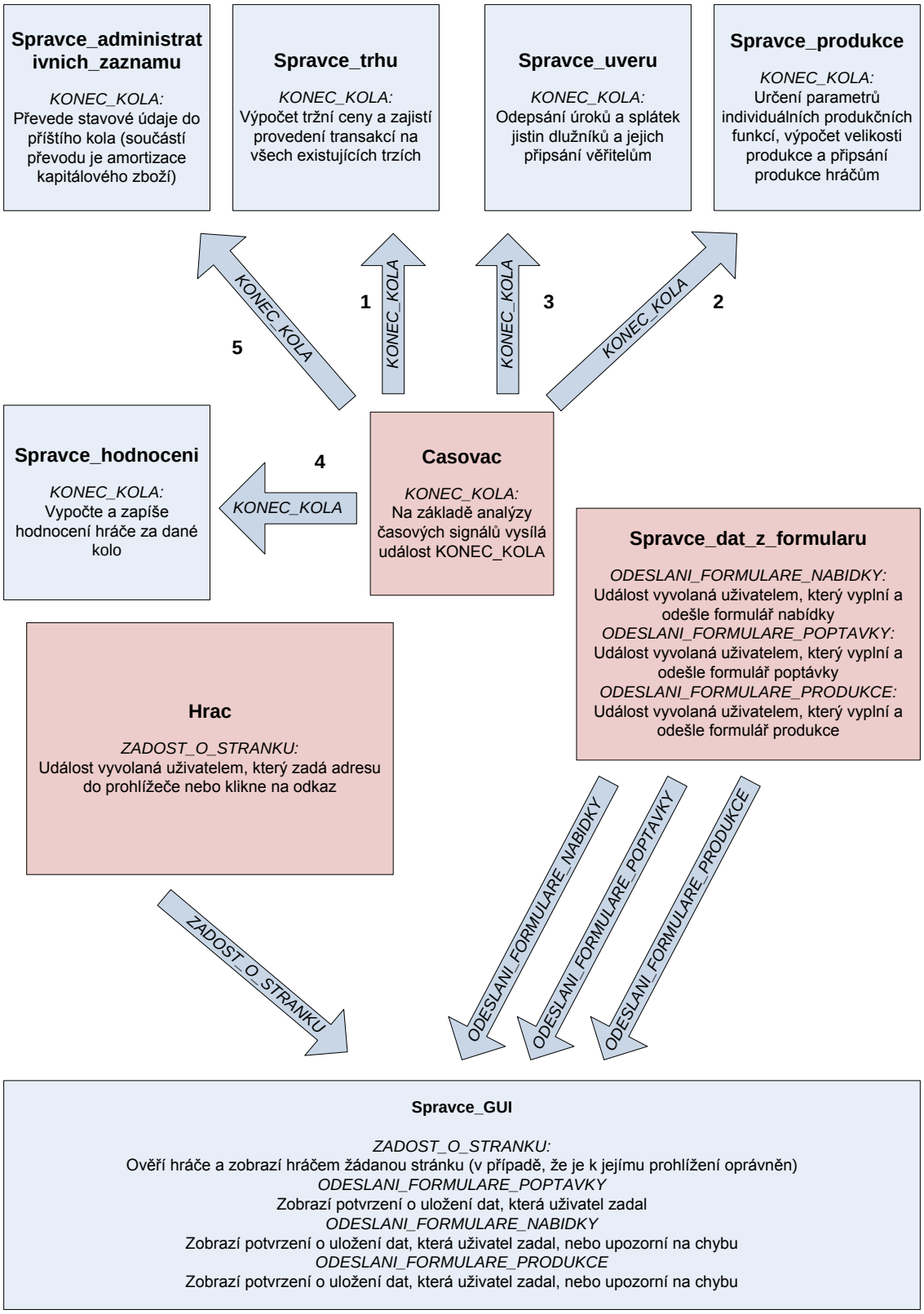
Dříve vloženým ovladačem byl *Ovladač 1*, ten tedy jako první provádí definovanou činnost – volá objekt *Správce trhu* a předává mu zprávu, aby spustil metodu *Vypočítej cenu – šipka 3*.

Po jejím provedení se aktivuje *Ovladač 2*, která u objektu *Správce produkce* spouští metodu *Vypočítej produkci – šipka 4.*



Obrázek 4 Průběh vyvolání události - zdroj: vlastní

3.5 Objektový model hry



Obrázek 5 Zjednodušený objektový model hry - zdroj: vlastní

Tento model zobrazuje objekty, které mají schopnost generovat nebo reagovat na události. Objekty generující události jsou zobrazeny červenou barvou, objekty na ně reagující modrou barvou. Šipky symbolizují události a čísla u šipek pořadí obsluhy ovladačů v případě, že je na jednu událost u jednoho objektu navěšeno více ovladačů.

3.6 Určení tržní ceny a tržních transakcí

Algoritmus výpočtu tržní ceny ve hře je založen na principu určení tržní ceny v mikroekonomii a dle metody maximálního obratu, která je používána na kapitálových trzích (Šedivá, 2009). Je navržen (a implementován) pro práci s diskrétně zadanými nabídkami a poptávkami a výsledkem algoritmu jsou vždy celočíselné realizované nabídky a poptávky a celočíselná tržní cena.

Algoritmus nejprve určí tržní nabídku a tržní poptávku.

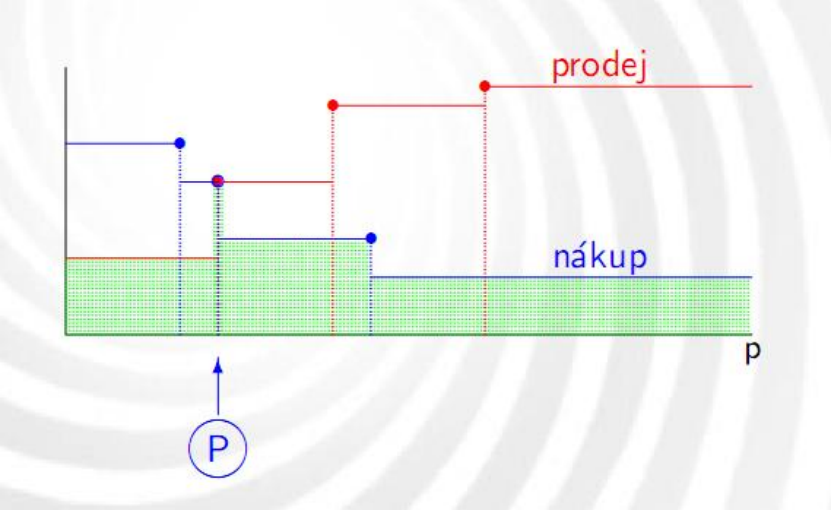
Při výpočtu tržní poptávky jsou postupně procházeny individuální poptávky. Pro každou z cen, kdy poptávané množství alespoň jednoho z hráčů je větší než nula, je součtem individuálních poptávaných množství všech hráčů určeno tržní poptávané množství.

Tržní nabídka je určena obdobně. Nabídky subjektů jsou však omezeny zdola (tedy cenou, při které není žádný ze subjektů na trhu ochoten nabízet). Algoritmus zjistí nejnižší cenu, při které je na trhu ochoten nabízet alespoň jeden subjekt.

Poté je určena nejvyšší cena, při které je nabízené množství některého z hráčů odlišné od nabízeného množství při ceně o jedna nižší. Jelikož nabídka není shora omezena, tržní nabízené množství při této ceně se rovná tržnímu nabízenému množství pro kteroukoli z vyšších cen. V rozmezí těchto dvou cen je pak pro každou cenu určen součet všech individuálních nabízených množství, což se rovná tržnímu nabízenému množství.

V následujícím kroku algoritmus vypočítá rozdíl mezi tržním nabízeným a poptávaným množstvím. Horní hranicí, pro kterou algoritmus rozdíl množství zjišťuje, je nejvyšší cena, pro kterou je tržní poptávané zboží nenulové. Jinými slovy se jedná o nevyšší cenu, při které na trhu poptává alespoň jeden hráč. Dolní hranicí pro výpočet rozdílu je cena 1.

Algoritmus zjistí cenu nebo interval cen, kde je absolutní hodnota rozdílu mezi tržním nabízeným a poptávaným množstvím nejmenší. Tento nejmenší rozdíl může být roven nule nebo může být větší než nula.

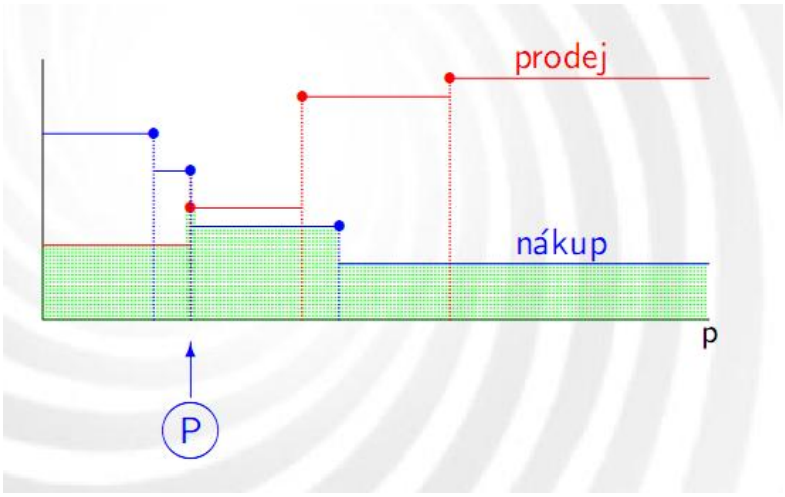


Obrázek 6 Určení tržní ceny, tržní nabízené a poptávané množství se rovnají – zdroj: (Šedivá, 2009)

V případě, že je rozdíl vyšší než 0, nedochází k úplnému vyčištění trhu. To je způsobeno diskrétní povahou cen, přičemž trh by byl vyčištěn při ceně, která by byla reálným číslem. Řešení takové situace je popsáno níže.

Minimální převis nabídky nad poptávkou nebo poptávky nad nabídkou může být platný pro větší počet tržních cen. Hráči totiž nedefinují nabízená a poptávaná množství pro jednotlivé ceny, ale pro intervaly cen (z důvodu časové náročnosti hry pro hráče). Pokud je počet cen, pro které platí minimální převis, lichý, je pak cena určena jako cena nacházející se ve středu tohoto intervalu. V případě, že je počet těchto cen sudý, je pak cena určena jako střed intervalu zaokrouhlený nahoru.

U nenulového rozdílu mezi nabízeným a poptávaným množství je nejprve rozlišena situace, kdy tržní poptávané množství převyšuje tržní nabízené množství, a kdy je tržní nabízené množství vyšší než tržní poptávané množství.



Obrázek 7 Tržní cena při převisu poptávky – zdroj: (Šedivá, 2009)

V prvním z uvedených případů nebudou plně uspokojeni všichni poptávající. Algoritmus provádí poměrné krácení poptávek. Individuální poptávané množství při vypočtené tržní ceně každého hráče je násobeno koeficientem krácení, který je roven podílu tržního nabízeného a poptávaného množství rovněž při vypočtené tržní ceně. Formálně můžeme koeficient vyjádřit takto:

$$K = \frac{\sum S}{\sum D} < 1$$

kde: K ... koeficient krácení

$\sum S$ … součet všech individuálních nabízených množství

$\sum D$ … součet všech individuálních poptávaných množství

Vzhledem k tomu, že nabízené množství je menší je poptávané, je tento koeficient menší než 1. Vynásobením individuálního poptávaného množství a koeficientu krácení získáme číslo menší, než je původní poptávané množství. Agregací takto získaných čísel od všech hráčů bychom získali číslo rovné tržnímu nabízenému množství, což dokazuje následující vztah:

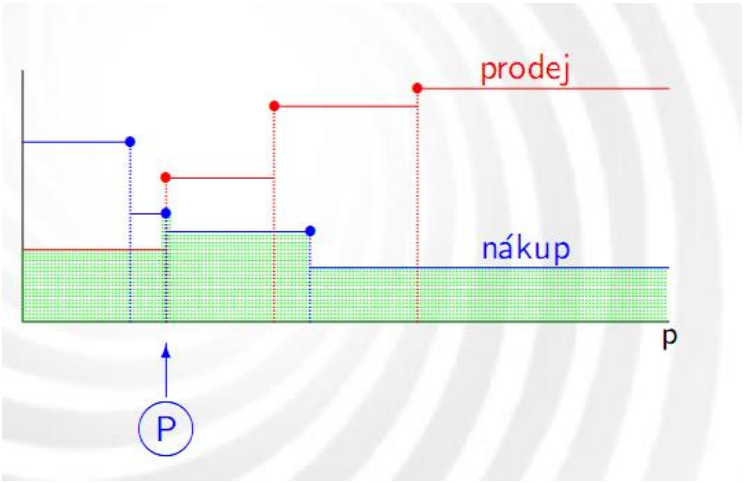
$$\sum (K * D) = K \sum D = \frac{\sum S}{\sum D} * \sum D = \sum S$$

Není však zaručeno, že všechna individuální nabízená množství budou celočíselná. Proto je každé poptávané množství po vynásobení koeficientem zaokrouhleno dolů. Výše popsaná rovnost pak obecně neplatí a součet upravených poptávaných množství všech hráčů na trhu může být menší než nabízené množství. Aby bylo dosaženo stavu

popsaného výše, tedy rovnosti poptávaného a nabízeného množství, je rozdíl mezi tržním poptávaným a nabízeným množstvím rozdělen náhodně mezi hráče.

Rozdělování probíhá v cyklu. Nejprve je zkontrolováno, zda již bylo dosaženo popisované rovnosti, a pokud ne, je některému z poptávajících náhodně připočteno k realizovanému poptávanému množství jednotka. Pokud je podmínka rovnosti splněna, dojde k zapsání určených údajů do databáze a algoritmus se ukončí.

V případě, že nabízené množství převyšuje poptávané množství, nejsou v plné výši realizovány nabídky všech nabízejících. Výpočet skutečně realizovaných nabízených množství pak probíhá přesně analogicky dle výše popsaného mechanismu. I zde tedy platí poměrné krácení a náhodné určení zbytku transakcí.



Obrázek 8 Tržní cena při převisu nabídky – zdroj: (Šedivá, 2009)

3.7 Vyhodnocování realizovaných transakcí

Po určení tržní ceny na konkrétním trhu a určení velikosti všech transakcí pro každého uživatele je třeba o každé transakci provést záznam. Záznamy se liší pro každý druh trhu, protože informace, které je třeba o transakci ukládat, jsou u jednotlivých trhů odlišné. Některé ze záznamů jsou vedeny z důvodu usnadnění přidávání nových funkcí a modulů ke hře. Programátor tak nemusí editovat kód obstarávající tržní transakce, ale pouze zajistí načtení připravených dat z databáze.

3.7.1 Prodej

V případě prodeje je třeba ve všech případech (s výjimkou trhu s finančními aktivy) přičíst hráči hotovost ve výši násobku počtu prodaných jednotek násobeném tržní

jednotkovou cenou. Toto číslo je nominálním vyjádřením hodnoty prodaného zboží v penězích.

3.7.1.1 Spotřební zboží

Pro trh spotřebního zboží je potřeba hráčovi odečíst objem prodaného zboží ze skladových zásob vztahující se k danému kolu. Hodnota skladových zásob po odečtení prodaného zboží může být minimálně nulová – hráči není dovoleno nabízet při jakékoli ceně více, než má v daném kole na skladě (definice nabídky, která ve které by nabízené množství převyšovalo skladové zásoby, by rozhraní hry vyhodnotilo jako neplatné a upozornilo by hráče na chybu). Takto uložené informace by však byly nedostatečné v případě, že by docházelo ke snižování skladových zásob v důsledku opotřebení (tj. v případě, že by se zboží dlouhodobým skladováním znehodnocovalo). Při zpětné rekonstrukci dat by totiž nebylo možné určit, kolik zboží ubylo v důsledku znehodnocení a kolik bylo prodáno. Proto jsou v databázi vedeny ještě záznamy o prodejích spotřebního zboží, které jednotliví hráči v každém kole prodali.

3.7.1.2 Kapitálové zboží

Situace u trhu kapitálového zboží je obdobná situaci u trhu spotřebního zboží. Je sníženo množství kapitálového zboží na skladě (přičemž hráč se opět nemůže dostat do záporných hodnot) a je proveden záznam o tom, kolik kapitálového zboží hráč v daném kole prodal.

3.7.1.3 Práce

V případě trhu práce samozřejmě nemá smysl mluvit o skladovaných položkách. Je však omezeno množství práce, kterou může hráč nabídnout. Omezením je disponibilní čas, který má hráč k dispozici. V případě trhu práce musí být zaznamenáno, kolik hodin hráč v daném kole odpracoval – to je nutné ke zjištění množství volného času, který měl hráč k dispozici.

3.7.1.4 Trh finančních aktiv

V případě trhu finančních aktiv nepřičítáme kapitál. Prodej na trhu finančních aktiv reprezentuje poskytnutí úvěru jinému hráči (neboli zapůjčení úspor) a tržní cena je úrokovou mírou, za kterou byl úvěr poskytnut. Hráči je tedy odečtena hotovost o výši přesně rovné velikosti objemu transakce. Následkem této operace se může hotovost hráče dostat do záporných hodnot, v době zadávání nabídek úvěrů totiž není možné

určit, kolik prostředků hráč získá prodejem či naopak či naopak kolik získá nákupem na ostatních trzích. V případě, že se hotovost hráče dostane do mínusu, je dle rozhodnutí administrátora hráč za tuto skutečnost bodově penalizován (případně je na to reagováno jiným způsobem).

O zapůjčení úspor je proveden záznam do tabulky záznamu o zapůjčených úsporách. Z této tabulky jsou dále určovány úroky, které jsou hráči za jím zapůjčené úspory vypláceny, a dále vrácení úspor hráči po uplynutí doby, na kterou byly úspory zapůjčeny.

3.7.2 Nákup

Obdobně jako u prodeje, i u nákupu je v případě trhů zboží a práce odečtena hráči hotovost rovná násobku množství nakoupených jednotek a tržní ceně. Tato částka je vyjádřením hodnoty nakoupené komodity v penězích.

3.7.2.1 Trh spotřebního zboží

U každého hráče je veden záznam o spotřebě zboží v průběhu hry. Ze spotřeby hráče se počítá bodové hodnocení pro každé kolo a záznamy o spotřebě jsou nutné pro rekonstrukci hry z nasbíraných dat, protože množství spotřebovaných statků je důležitým údajem vypovídajícím o rozhodování a chování hráče.

3.7.2.2 Trh kapitálového zboží

Stejně tak je veden záznam o nákupu kapitálového zboží. Tyto záznamy jsou základem pro implementaci různých způsobů odepisování majetku. Informace o nákupu kapitálového zboží v jednotlivých kolech pak je základem pro implementaci např. rovnoměrného odepisování.

Dále je nákup kapitálového zboží zaznamenán do tabulky evidující množství tohoto výrobního faktoru ve výrobě. Zboží tak vstupuje do výroby ihned po nákupu a objem vyrobené produkce v daném kole je tak vyšší, než jakého by bylo dosaženo bez tohoto nákupu (za jinak stejných podmínek). Ve hře tedy neexistuje rozdíl mezi krátkým a dlouhým obdobím, jak ho definuje mikroekonomie, tedy že krátké období představuje časový úsek, po který je alespoň jeden výrobní faktor fixní (Samuelson, Nordhaus, 2007).

3.7.2.3 Trh práce

Nakoupená práce vstupuje do výroby právě v kole, kdy byla zakoupena. Je třeba připomenout, že do produkční funkce je nutné připočítat i počet hodin, které odpracoval sám hráč pro vlastní výrobu.

Nákupy práce hráčem jsou zaznamenávány do tabulky záznamů nakoupené práce. Ze záznamů nakoupené práce je pak možné zjistit, ve kterých kolech vystupoval hráč jako zaměstnavatel (a spolu se záznamy o prodané práci lze určit, kdy přijal roli zaměstnance).

3.7.2.4 Trh finančních aktiv

Nákup na trhu finančních aktiv znamená získání úvěru. K hotovosti hráče je připočtena částka rovnající se velikosti transakce v jednotkách. Tržní cena opět představuje úrokovou míru, na kterou byl úvěr přijat. O získání úvěru je dále veden záznam v tabulce získaných úvěrů, podle kterého jsou hráči strhávány úroky a po uplynutí období, na které byl úvěr uzavřen, je hráči zapůjčená částka odečtena.

4 Závěr

Bakalářská práce je rozdělena na teoretickou a praktickou část. Cílem teoretické části bylo navržení ekonomického modelu hry a objektového modelu, který je podkladem pro praktickou část. Praktická část spočívala v implementaci software dle objektového modelu a zpracování programátorské dokumentace.

Při navrhování ekonomického modelu bylo stěžejní určení obchodovaných komodit, výrobních faktorů a aktivních trhů. Součástí teoretické části je pak i matematická formalizace hry včetně teoretického popisu vlastností produkční a užitkové (hodnotící) funkce.

V další části teoretické části je popsán objektový model hry. Klíčovou pro návrh objektového modelu byla analýza toků zboží, služeb, výrobních faktorů a peněz v ekonomickém modelu. Objektový model byl pak navržen tak, aby na základě rozhodnutí hráčů provedl přesný výpočet všech toků, ke kterým v důsledku rozhodnutí dojde. Pro komunikace mezi objekty je využito systému událostí, jehož návrh je další součástí teoretické části práce.

Objektový návrh a systém událostí umožňují snadné rozšíření hry o další funkce. K rozšiřování hry pak není nutné zasahovat do již vytvořených objektových tříd.

Dále byl navržen klíčový algoritmus pro výpočet tržní ceny a určení velikosti transakcí při zachování předpokladu diskrétních cen a velikosti obchodních transakcí. Algoritmus uvažuje i řešení možnosti, kdy při tržní si nejsou rovny tržní nabízené a poptávané množství.

Výstupem praktické části je implementace hry na základě objektového modelu, který je popsán v teoretické části. Díky využití objektově orientovaného programování a systému generování a registrování událostí je hra velmi snadno rozšiřitelná o další funkce, respektive o přidání objektů, které tyto funkce budou reprezentovat.

Testování hry bylo zahájeno 26. 4. 2010 a aktivně se jej účastnilo 10 osob (studentů a akademických pracovníků FEK ZČU). Během testování byly shromažďovány podněty a připomínky od hráčů.

5 Seznam obrázků

6 Seznam použité literatury

Achour, M., a další. 2010. PHP Manual. *PHP: Hypertext Preprocessor*. [Online] The PHP Group, 9. 4 2010. http://www.php.net/manual/en/index.php.

Čihák, M. a Holub, T. 2000. *Teorie růstové politiky.* 2000. ISBN 80-2450126-0.

Doepke, M., Lenhert, A. a Sellgren, A. W. 1999. *Macroeconomics.* 1999.

Fells, D. 2006. Simulating Events with PHP 5. *Open Source Web Development Tutorials - Dev Shed.* [Online] 20. 2 2006. http://www.devshed.com/c/a/PHP/Simulating-Events-with-PHP-5/.

Frank, R. H. a Bes, B. S. 2003. *Ekonomie.* Praha : Grada Publishing a. s., 2003. ISBN 80-247-0471-4.

Henderson, J. M. a Quandt, R. E. 1980. *Microeconomic Thery - A Mathematical Approach.* New York : McGraw-Hill, Inc., 1980. ISBN 0-70-028101-7.

Holman, R. 2004. *Makroekonomie: středně pokročilý kurz.* Praha : C. H. Beck, 2004. ISBN 80-7179-764-2.

Mankiw, N. G. 2000. *Zásady ekonomie.* Praha : Grada Publishing, spol. s r. o., 2000. ISBN 80-7169-891-1.

Mises, L. von. 2006. *Lidské jednání: Pojednání o ekonomii.* Praha : Liberální institut, 2006. ISBN 80-86389-45-6.

Puš, P. 2005. Poznáváme C# a Microsoft.NET 16. díl – události. *O počítačích, IT a internetu - Živě.cz.* [Online] 25. 3 2005. http://www.zive.cz/clanky/poznavame-c-a-microsoftnet-16-dil--udalosti/sc-3-a-123623/default.aspx.

Revenda, Z. a kolektiv. 1998. *Peněžní ekonomie a bankovnictví.* Praha : MANAGEMENT PRESS, Ringier ČR, a.s., 1998. ISNB 80-85943-49-2.

Samuelson, P. A. a Nordhaus, W. D. 2007. *Ekonomie.* Praha : NS Svoboda, 2007. ISBN 978-80-205-0590-3.

Šedivá, B. 2009. Organizátoři trhu cenných papírů. *Blanka Šedivá - KPT.* [Online] 2009. [Citace: 8. 5 2010.] http://home.zcu.cz/~sediva/kpt/pred_04.pdf.

7 Sezam příloh

Hra je v současnosti (květen 2010) spuštěná na adrese www.dsgegame.zcu.cz, na tomtéž místě je pak uložen i zdrojový kód.

8 Abstrakt

PEŠÍK J., Simulační hra pro podporu výuky základů finanční gramotnosti pro střední školy, Plzeň: Fakulta ekonomická ZČU v Plzni, 98 s., 2010

Klíčová slova: ekonomická hra, finanční gramotnost, ekonomické modely

V bakalářské práci autor řešit problematiku vytvoření simulační hry pro výuku finanční gramotnosti. Byl vytvořen ekonomický model hry na základě ekonomické teorie. Modelovaná ekonomika je uzavřená a funguje bez zásahů vlády. V ekonomice existuje jeden druh spotřebních statků. Výrobními faktor jsou práce a kapitálové zboží, prostředkem směny peníze. Hráči mohou na trhu finančních aktiv spořit nebo získat úvěr. Dále byl navržen objektový model pro implementaci hry. Výstupem praktické části je simulační hra pro neomezené množství hráčů. Výsledky práce mohou být uplatněny při výuce finanční gramotnosti (vzácnost zdrojů, hodnota peněz v čase, vliv inflace). Data ze hry mohou být využita pro ekonometrické zkoumání chování subjektů v ekonomice. Model hry i hra mohou být dále rozšířeny o existenci vlády, exogenní šoky a otevřenou ekonomiku (rozdělení hráčů do více propojených ekonomik).

9 Abstract

PEŠÍK J., Simulační hra pro podporu výuky základů finanční gramotnosti pro střední školy, Plzeň: Fakulta ekonomická ZČU v Plzni, 98 s., 2010

Key words: economic game, financial literacy, economic models

The purpose of this paper was to solve a creation of a simulation game for education of financial literacy. Author created an economic model, which is based on economic theory. The economy in the model is self-sufficiency with no influence of government. There is one brand of consumption good in the economy. There two manufacturing factors: work and capital good. Medium of exchange is money. Players are allowed to save or get a loan on a market of financial actives. Author made an object model for an implementation of the game. The outcome of practical part of paper is the simulation game for unlimited number of players. This game can be used for education of financial literacy (scarcity of resources, value of money in time, inflation) and for econometric studies of decision-making. The economic model and the game can be expanded by influence of government, exogenous shocks and open economy (dividing players in several connected economies).

10 Příloha A: Programátorská dokumentace a objektový model hry

Tato kapitola se věnuje samotné implementaci hry. Je v ní vysvětlen objektový model, na kterém je hra postavena, popsány všechny důležité objektové třídy a události, které v aplikaci postupně nastávají.

10.1 Využité knihovny a třídy třetích stran

Při implementaci bylo využito několika softwarových produktů třetích stran. Všechny využité produkty jsou šířeny pod svobodnou licencí a u všech je výslovně povolena implementace do projektů jiných autorů.

10.1.1 EvalMath

Zdroj: http://www.phpclasses.org/package/2695-PHP-Safely-evaluate-mathematical-expressions.html

Třída pro vyhodnocování matematických výrazů z řetězců. Umožňuje administrátorovi zadávat produkční a hodnotící funkce jako řetězec, který je pak strojově touto třídou zpracován.

Licence: BSD[2].

10.1.2 JpGraph

Zdroj: http://www.aditus.nu/jpgraph/

Objektově orientovaná knihovna pro generování grafů ze zadaných matic. Skripty využívající knihovnu JpGraph se od kódu stránky vkládají pomocí tagu img a knihovna nepoužívá souborový systém pro ukládání generovaných obrázků. Tato knihovna podporuje mnoho druhů grafů.

Licence: QPL[3]

10.1.3 TinyMCE

Zdroj: http://tinymce.moxiecode.com/

[2] http://www.opensource.org/licenses/bsd-license.html

[3] http://qt.nokia.com/doc/4.0/qpl.html

Editor HTML napsaný v jazyce JavaScript. Tento editor umožňuje snadné vkládání formátovaného XHTML obsahu do aplikace. To je využito u editace úvodní stránky a dále je možné do aplikace snadno doimplementovat správce článků, který bude rovněž využívat tento editor. Administrátor tak může vkládat formátovaný text bez znalosti XHTML syntaxe.

Licence: LGPL[4]

10.2 Systém řízení událostí

Teoretický princip systému je popsán ve čtvrté kapitole. Zde je popsána jeho praktická implementace. Implementace v DSGE Game je rovněž postavena na modelu Davida Fellse (Fells, 2006).

Systém je navržený na principu postupného „probublávání“ události třídy, která informaci o události vyslala, přes celý systém řízení událostí až ke třídám, které projevily zájem na tuto událost reagovat. Nadřízená třídy vždy pouze zavolá patřičnou metodu třídy, která je v systému předávání událostí další v pořadí, a ta si již sama zajišťuje veškeré náležitosti. K programování tříd, které události generují nebo které naopak mají přidělenou reakci na událost, není nutné studovat implementaci řešení řízení událostmi. Stačí pouze nastudovat dokumentaci k rodičovským třídám, jejichž potomky musejí být všechny třídy, které událost generují nebo na ni reagují.

10.2.1 Ovladac

Třída Ovladac reprezentuje a provádí akci, která je provedena, pokud dojde k vyvolání události.

10.2.1.1 Vlastnosti třídy

private $nazev_udalosti

Identifikátor události, na kterou je ovladač napojen.

private $nazev_spoustene_metody

Název metody, kterou ovladač u napojeného objektu spouští v případě vyvolání události. Tato metoda musí být public a musí mít parametry odesilatel a parametry (význam parametrů je vysvětlen u popisu třídy Trida_reagujici_na_udalost).

[4] http://tinymce.moxiecode.com/license.php

private $napojeny_objekt

Objekt, na který je událost napojena. Ovladač u tohoto objektu volá metodu, která je určena vlastnosti nazev_spoustene_metody.

10.2.1.2 Metody třídy

function __construct($nazev_udalosti, $nazev_spoustene_metody, $napojeny_objekt)

Konstruktor třídy, který definuje všechny vlastnosti třídy.

public function proved_akci_ovladace($odesilatel, $parametry)

Volání definované metody za pomocí příkazu eval. Parametr odesilatel je odkaz na objekt, který událost vyvolal, a parametry je polem parametrů, které s událostí předává třída, která událost vyvolala. Oba tyto parametry jsou dále předávány napojenému objektu.

10.2.2 Fronta_ovladacu

Každá událost má vlastní strukturu, do které jsou ukládány její ovladače. Tuto strukturu reprezentuje právě třídy Fronta_ovladacu. Je to datová struktura typu FIFO – pokud tedy budeme volat všechny ovladače ve frontě, budou akce ovladačů prováděny právě v tom pořadí, ve kterém byly do fronty vloženy.

10.2.2.1 Vlastnosti třídy

private $identifikator_udalosti

Jednoznačně označuje událost, které fronta ovladačů náleží. Tento identifikátor musí být shodný se všemi ovladači, které jsou ve frontě.

private $fronta_ovladacu

K ukládání ovladačů je využita třída ArrayObject. Tato třída je dynamickou datovou strukturou, takže do ní mohou být postupně vkládány nové položky bez nutnosti toho, aby při vytvoření byl zadán jejich počet.

10.2.2.2 Metody třídy

function __construct($identifikator_udalosti)

Konstruktor třídy definuje vlastnost identifikator_udalosti a inicializuje vlastnost fronta_ovladacu.

public function pridej_ovladac($ovladac)

Přidá ovladac na poslední místo ve frontě.

public function reaguj_na_vyvolani_udalosti($odesilatel, $parametry)

Reakcí na vyvolání události je provedení akcí všech ovladačů ve frontě. Všem ovladačů jsou předány parametry odesilatel a parametry, tato metoda jinak parametry nezpracovává.

10.2.3 Udalost

Je reprezentací události tak, jak byla popsána ve čtvrté kapitole. Každá třída, která má schopnost nějakou událost vyvolávat, si nejdříve vytvoří instanci třídy Udalost, která je uložena ve frontě a čeká, až bude vyvolána. Informace o vyvolání poté postupně dostanou všechny třídy, které na instanci třídy Udalost navěsily svůj ovladač.

10.2.3.1 Vlastnosti třídy

private $identifikator

Tato vlastnost jednoznačně identifikuje událost. Podle tohoto identifikátoru jsou události vyvolávány a podle něj jsou na něj i navěšovány ovladače. Vlastnost identifikator je řetězec a skládá se pouze z velkých písmen a podtržítek.

private $fronta_ovladacu

Je instancí třídy Fronta_ovladacu a obsahuje ovladače, jejichž definované akce budou prováděny při volání události.

10.2.3.2 Metody třídy

public function __construct($identifikator)

Konstruktor třídy nastaví identifikator, který po dobu existence instance již nebude měněn. Dále je inicializována Fronta_ovladacu.

public function registruj_ovladac(Ovladac $ovladac)

Registrace (neboli navěšení) ovladače na událost.

public function vyvolani_udalosti($odesilatel, $parametry)

Vyvolání události, na které reagují všechny navěšené ovladače.

10.2.4 Fronta_udalosti

Každá třída, která může generovat a vyvolávat události, má vlastní datovou strukturu, do které ukládá události, které může vyvolat, a čekají tam na své vyvolání. Tato struktura je implementovaná jako třída Fronta_udalosti.

10.2.4.1 Vlastnosti třídy

private $fronta_udalosti

Je vnitřní reprezentací fronty, využívá třídu ArrayObject. Na uložení událostí ve frontě ale, na rozdíl od ovladačů, nezáleží, protože z fronty je vždy volána právě jedna konkrétní událost (v případě ovladačů jsou volány vždy všechny ovladače ve frontě).

10.2.4.2 Metody třídy

function __construct()

Konstruktor třídy. Zajistí vytvoření instance třídy ArrayObject a jeho přiřazení vlastnosti fronta_udalosti.

public function pridej_udalost_do_fronty(Udalost $pridavana_udalost)

Vloží parametr pridavana_udalost do fronty. Nejdříve je ovšem zkontrolováno, zda fronta již událost s tímto identifikátorem neobsahuje. To zjišťuje metoda obsahuje_udalost.

public function obsahuje_udalost(Udalost $testovana_udalost)

Projde všechny události ve frontě a v případě, že u některé z událostí narazí na stejný identifikátor jako testovana_udalost, vrátí true, v opačném případě vrací false.

public function prirad_udalosti_ve_fronte_ovladac(Ovladac $ovladac)

Přiřadí parametr ovladac konkrétní události. Každá instance třídy Ovladac má vlastnost nazev_udalosti, který jednoznačně identifikuje událost, které ovladac náleží. Parametr ovladac je přiřazen správné události právě na základě této vlastnosti.

public function vyvolani_udalosti_ve_fronte($identifikator_udalosti, $odesilatel, $parametry)

Vyvoláním události ve frontě rozumíme samotné předávání informace ostatním třídám. Vyvolání události vždy provádí třída, které fronta události náleží. Třída vždy pouze předá frontě příkaz k vyvolání událostí.

10.2.5 Trida_generujici_udalosti

Všechny třídy, které mají schopnost generovat (vyvolávat) události, musejí být potomky této třídy. Tato třída poskytuje veškeré metody, které jsou nutné z hlediska systému řízení událostmi pro třídu, která události generuje.

Tato třída disponuje možností vyvolávat události, na které mohou další třídy reagovat. Pro instanci této třídy není důležité, které objekty budou na vyvolání události čekat. Tyto objekty se registrují v hlavním programu.

10.2.5.1 Vlastnosti třídy

Klíčové slovo protected zajišťuje, že vlastnosti budou přímo přístupné pro tuto třídu a všechny třídy, které jsou jejím potomkem.

protected $fronta_udalosti

Je instancí třídy Fronta_udalosti. Obsahuje veškeré vlastnosti, které třída může generovat.

protected $pole_identifikatoru_udalosti

Obsahuje identifikátory všech událostí, které může třída generovat. Toto pole je potřeba vždy naplnit v konstruktoru potomka a poté zavolat konstruktor rodičovské třídy, jinak nebudou události se správnými identifikátory zapojeny do fronty a třída je nebude moci vyvolat. Viz. dále konstruktor třídy.

10.2.5.2 Metody třídy

public function __construct()

Konstruktor třídy. Jsou vytvořeny události pro všechny identifikátory, které jsou uloženy ve vlastnosti pole_identifikatoru_udalosti, a zařazeny do fronty náležející třídy. Tento konstruktor musí být vždy volán rodičovskou třídy uložení všech identifikátorů do pole_identifikatoru_udalosti!

Pokud tedy bude chtít programátor zjistit, které události generuje ta která třída, stačí se podívat do jejího konstruktoru – není potřeba studovat dokumentaci, požadovaná informace je zde velmi snadno a rychle zjistitelná ze zdrojového kódu.

Pro objasnění následuje příklad konstruktoru třídy, která generuje událost:

```
public function __construct() {
    $this->pole_identifikatoru_udalosti[] = 'ODESLANI_FORMULARE_NABIDKY';
    $this->pole_identifikatoru_udalosti[] = 'ODESLANI_FORMULARE_POPTAVKY';
    $this->pole_identifikatoru_udalosti[] = 'ODESLANI_FORMULARE_PRODUKCE';
                        ... //případné další příkazy třídy
    parent::__construct();
}
```

public function registruj_ovladac($ovladac)

Tato třída je volána z hlavního programu. Slouží k navěšování ovladačů na tuto třídu (přesněji na jednu z událostí generovaných třídou). Metoda pouze předá parametr ovladac frontě událostí (princip „probublávání“).

Příklad navěšení ovladače v hlavním programu je ukázán po popisu třídy Trida_reagujici_na_udalosti.

10.2.1 Trida_reagujici_na_udalosti

Musí být rodičovskou třídou pro všechny třídy, které mají mít funkci reakce na nějakou událost.

Reakce na událost je prakticky realizována voláním některé z metod třídy. Jelikož z pohledu této třídy jde o volání z vnějšku, tato metoda musí být public a dále musí mít právě dva parametry: odesilatel a parametry. Parametr odesilatel je odkaz na třídu, která událost generovala. Zde tak zjistit, kdo vlastně událost vyvolal a tomu upřesnit reakci (událost se stejným identifikátorem totiž může být vyvolána různými třídami). Parametry je asociativní pole, u něhož identifikátor vždy identifikuje parametr a hodnota je hodnotou parametru.

10.2.1.1 Vlastnosti třídy

protected $pole_identifikatoru_udalosti

Stejně jakou u trida_generujici_udalost, má i trida_reagujici_na_udalost pole, ve kterém jsou textové identifikátory událostí, se kterými třída pracuje. V případě této třídy ale je pole asociativní. Identifikátor události je v tomto poli klíčem a hodnotou je název metody, která bude spuštěna při vyvolání události.

10.2.1.2 Metody třídy

public function __construct()

Konstruktor třídy. V současné době je prázdný, pro případ změn v systému řízení událostí je však doporučeno volat konstruktor rodičovské třídy v konstruktoru potomka.

V konstruktoru potomka je však nutné definovat kompletně naplnit pole_identifikatoru_udalosti! To je jednak nutné pro funkčnost programu a dále pro přehlednost. Programátor totiž z konstruktoru třídy okamžitě vidí, na které události třída reaguje a jak na ně reaguje – tedy která metoda je zavolána.

public function generuj_ovladac_na_udalost($identifikator_udalosti)

Tato metoda vrátí ovladač pro událost s identifikátorem podle parametru identifikator_udalosti. Ovladač je napojený na třídu, která tuto metodu vlastní (princip zapouzdření) a může být přímo navěšen na událost, která danou událost vyvolává.

10.2.2 Navěšování ovladačů a registrace k událostem v hlavním programu

Navěšování ovladačů se provádí v kódu souborů, které jsou přímo spouštěny webovým serverem (např. soubor index.php), tedy mimo kód jakékoli třídy. Nejprve je potřeba vytvořit instance všech tříd, které mohou události generovat a instance všech tříd, které budou na vyvolané události reagovat. V příkladě bude spravce_dat_z_formularu bude generovat události ODESLANI_FORMULARE_NABIDKY, ODESLANI_FORMULARE_POPTAVKY a ODESLANI_FORMULARE_PRODUKCE. Spravce_GUI bude na všechny tyto události reagovat. Dále spravce_zaznamu_o_aktivite bude reagovat na událost ODESLANI_FORMULARE_POPTAVKY.

```
$spravce_dat_z_formularu = new Spravce_dat_z_formularu($_POST);
$spravce_GUI = new spravce_GUI();
$spravce_zaznamu_o_aktivite = new Spravce_zaznamu_o_aktivite();

$spravce_dat_z_formularu->registruj_ovladac($spravce_GUI-
>generuj_ovladac_na_udalost('ODESLANI_FORMULARE_NABIDKY'));
$spravce_dat_z_formularu->registruj_ovladac($spravce_GUI-
>generuj_ovladac_na_udalost('ODESLANI_FORMULARE_POPTAVKY'));
$spravce_dat_z_formularu->registruj_ovladac($spravce_GUI-
>generuj_ovladac_na_udalost('ODESLANI_FORMULARE_PRODUKCE'));

$spravce_dat_z_formularu->registruj_ovladac($spravce_zaznamu_o_aktivite-
>generuj_ovladac_na_udalost('ODESLANI_FORMULARE_POPTAVKY'));
```

Velmi důležité je nezapomenout na fakt, že v tomto případě při vyvolání události ODESLANI_FORMULARE_POPTAVKY bude nejprve obsloužen ovladač objektu spravce_GUI a následně spravce_zaznamu_o_aktivite. Pořadí obsloužení ovladačů je plně deterministické a definuje ho pořadí řádku s příkazy navěšení ovladače.

10.3 Události generované a zpracovávané hrou

Tato podkapitola přináší přehled událostí spolu s popisem toho, jaký děj v modelu hry konkrétní akce reprezentuje. Události v aplikace jsou vyvolány dvěma způsoby – buď uživatelem, který se hrou komunikuje prostřednictvím webového prohlížeče, nebo časovým signálem, který je aplikaci odeslán službou Cron.

10.3.1 KONEC_KOLA

Událost KONEC_KOLA reprezentuje konec aktuálního kola ve hře. Je vyvolána třídou Casovac, který na základě údajů o aktuálním čase rozhodne o ukončení kola a zahájení nového. Při ukončení kola je potřeba určit tržní ceny na všech trzích, na základě těchto cen provést na všech trzích tržní transakce, určit velikost produkce podle výrobních faktorů zapojených do výroby u každého hráče, provést splátky úvěrů a úroků z úvěrů a amortizaci kapitálového zboží.

Dále může být vyvolán třídou Spravce_dat_z_formularu v případě, že se administrátor, který je vybaven patřičným oprávněním, rozhodne manuálně ukončit kolo a zahájit nové

(tato možnost je užitečná v případě, že není k dispozici možnost volání skriptu démonem Cron).

Tato událost je na jedné straně klíčovou událostí pro hru, na druhou stranu počet jejích vyvolání je roven počtu kol hry (následující události jsou vyvolávány v mnohonásobně častějších intervalech).

10.3.2 ZADOST_O_STRANKU

Je vyvolána hráčem, přesně řečeno zadáním adresy do webového prohlížeče, kliknutím na odkaz, který směřuje na soubor index.php v kořenovém adresáři hry nebo odesláním formuláře. Událost je zpracována na základě GET parametrů, které zadaná adresa obsahuje. Tyto parametry specifikují stránku, o kterou uživatel žádá. Tuto žádost zpracovává třída Spravce_GUI.

10.3.3 ODESLANI_FORMULARE_NABIDKY

Je vyvolána hráčem, který zadal data do formuláře nabídky a tato data odeslal. Hlavní aplikace zachytává veškerá data zasílaná metodou POST a filtruje je. Třídou Spravce_udalosti jsou z těchto dat generovány události Odeslani_formulare_nabidky. Těmto událostem jsou přiřazovány parametry. Parametrem chyba_v_datech je logická hodnota, která má hodnotu true, pokud data zadaná uživatelem obsahují chybu.

Chybou je v případě nabídky myšlena situace, kdy

- hráč nabízí více zboží, než má na skladě
- součet počtu hodin, které hráč plánuje využít pro vlastní výrobu a některé z nabízených množství na trhu práce, překračují 24 hodin
- hráč nabízí na trhu finančních aktiv větší hotovost, než má k dispozici

Za chybu neoznačujeme zadání nečíselného znaku. V případě, že je takový znak hráčem zadán, je ze vstupních dat odstraněn a z dat jsou uvažovány pouze zadané číselné znaky (to slouží i jako obrana proti zadání škodlivého kódu do formulářů).

Parametrem pole_meznich_cen je jednoznačnou definicí nabídky, kterou uživatel zadal. Je shodné se shodně pojmenovaným atributem třídy Individualni_nabidka. Toto pole je blíže specifikována v subkapitole věnující se třídě Individualni_nabidka.

Parametrem pole_chyb je pole, které obsahuje informace specifikace chybných údajů, které hráč zadal. Výše specifikované chyby jsou označeny E.

10.3.4 ODESLANI_FORMULARE_POPTAVKY

Dále může hráč vyvolat událost odeslání formuláře specifikujícího poptávku. V případě poptávky nelze hovořit o chybě ve smyslu výše uvedených příkladů. Hráč není nijak omezen v tom, kolik nakoupí. Jediným parametrem události je tedy parametr pole_meznich_cen, který je definicí zadané poptávky totožnou s vlastností třídy Individualni_poptavka.

10.3.5 ODESLANI_FORMULARE_PRODUKCE

Další událost vyvolaná odesláním formuláře, konkrétně formulářem definujícím příkaz produkce. Ve specifikaci toho, kolik hodin hráč zapojí do vlastní výroby, může být hodnota chybná v následujících případech:

- hráč zadal vyšší hodnotu než 24 hodin, tedy více, než má pro každé kolo k dispozici
- hráč nejprve definoval svoji nabídku práce a pak příkaz produkce, přičemž součet některého z nabízených množství práce a počet hodin práce pro vlastní výrobu je vyšší než 24 hodin (ale samotný příkaz produkce je nižší nebo roven 24 hodinám)

Aby došlo k odlišení těchto dvou případů a byla zachována možnost doporučit hráči odlišné návrhy řešení chyby (na první případ totiž musí hráč reagovat změnou příkazu produkce, ve druhém případě ale může kromě změny příkazu produkce provést změnu zadané individuální nabídky práce a následně znovu zadat stejný příkaz produkce), jsou oba případy předávány jako parametry události s odlišným označením.

Parametr pole_chyb události tedy obsahuje znak E pro klíč 1 pro případ první chyby a S pro případ druhé chyby.

Parametrem chyba_v_datech je logická hodnota, která je rovna true v případě, že uživatel zadal chybnou hodnotu (ve smyslu výše jmenovaných případů).

Posledním parametrem je pole default_data, které obsahuje dvě hodnoty. Pro klíč hodin_prace je to počet hodin práce, které hráč zadal. Pro klíč druh_zbozi má hodnotu 1 v případě, že hráč zvolil výrobu spotřebního zboží, a hodnotu 2, pokud zvolil výrobu kapitálového zboží.

10.3.6 ZADOST_NOVEHO_HRACE_O_ZAPOJENI_DO_HRY

Tuto událost vyvolá vždy hráč, který poprvé navštíví web hry a rozhodne se do hry zapojit. V reakci na tuto událost třída Spravce_administrativnich_zaznamu vytvoří záznam o tomto hráči v databázi a přiřadí mu počáteční množství skladovaných položek a kapitálového zboží ve výrobě, které nastavuje administrátor.

10.4 Třídy generující události

10.4.1 Hrac

Tato třída má zvláštní postavení, protože v některých okamžicích vystupuje jako třída generující události a v jiných jako datová třída. Tento zdánlivý rozpor má však logický důvod: V okamžiku, kdy hráč ovládá hru prostřednictvím webového rozhraní, generuje události a očekává, že na ně hra bude reagovat.

Při analýze tržní situace naopak objekt, který hráče reprezentuje, předává trhu informace o příkazech zadaných hráčem k následné analýze. Tento objekt tedy načte z databáze data, o která byl přímo požádán jiným objektem, a tato data mu předá. To splňuje účel datové třídy.

10.4.1.1 Události generované třídou

Tato třída může generovat událost ZADOST_O_STRANKU.

10.4.1.2 Vlastnosti třídy

private $mnozstvi_kapitalu

Množství kapitálu neboli hotovost, kterou má hráč k dispozici. Hotovost nemusí být celé číslo z důvodu splátek úroků. Násobíme totiž celé číslo (velikost úvěru) úrokovou mírou, což je reálné číslo. Výsledkem operace tedy může být reálné číslo. Třída však při dotazu na množství kapitálu vrací celé číslo (zaokrouhlování dolů). Důvodem zaokrouhlení je zjednodušení pro hráče, kteří kalkulují s diskrétními cenami.

private $mnozstvi_kapitaloveho_zbozi

Množství kapitálového zboží, které má hráč na skladě v aktuálním kole. Celé číslo.

private $mnozstvi_kapitaloveho_zbozi_ve_vyrobe

Množství kapitálového zboží, které je zapojeno do výroby. Celé číslo.

private $mnozstvi_spotrebniho_zbozi

Množství spotřebního zboží, které hráč skladuje. Toto zboží nelze přímo spotřebovat, ale musí nejdříve projít trhem.

private $login

Login jednoznačně identifikuje hráče. Je primárním klíčem nebo částí primárního klíče ve většině tabulek databáze. V případě, že je částí klíče, pak je další položkou klíče číslo aktuálního kola. V takové tabulce jsou pak archivovány údaje o hráčích pro každé kolo.

private $cislo_aktualniho_kola

Informace hráče o tom, ve kterém kole se hra nachází. Pokud je hráč požádán o nějakou informaci, která se v průběhu hry mění, pak vrátí vždy informaci, která se vztahuje k aktuálnímu kolu.

10.4.1.3 Metody třídy

V případě metod, které upravují některou z vlastností instance třídy, je součástí metody i aktualizace příslušného údaje v databázi. Data mezi instancí třídy a databází musejí být neustále konzistentní.

public function __construct($login)

Konstruktor třídy. Jediným parametrem konstruktoru je login, který je uložen do příslušné vlastnosti třídy. Dále jsou nastaveny všechny další vlastnosti, a to načtením databáze a vlastnost cislo_aktualniho_kola je zjištěna od objektu spravce_konfigurace, který musí být součástí kódu hlavního programu (tímto způsobem kolo zjišťují i jiné třídy). Tímto postupem je zabráněno vytváření více instancí této třídy, která by pak vysílala zbytečné dotazy na databázi a navíc by mohlo dojít i k nekonzistenci dat mezi různými instancemi těchto tříd.

public function zvys_mnozstvi_kapitalu($zvyseni)

Zvýší velikost hotovosti hráče o daný parametr. Ten může být celočíselný nebo i reálné číslo (v případě připisování úroků).

public function sniz_mnozstvi_kapitalu($snizeni)

Sníží velikost hotovosti o zadaný parametr. I v tomto případě může být parametr celočíselný i reálný.

public function zvys_mnozstvi_kapitaloveho_zbozi($zvyseni)

Zvýší množství kapitálového zboží, které má hráč na skladě pro následný prodej, o parametr. Tato funkce je volána při zpracovávání výsledků výroby, protože nakoupené kapitálové zboží jde rovnou do výroby.

public function sniz_mnozstvi_kapitaloveho_zbozi($snizeni)

Sníží množství kapitálového zboží, které hráč skladuje. Tato metoda je využívána v případě prodeje zboží, protože prodané zboží se odečítá ze skladovaných položek (a nikoli z kapitálového zboží, které je již zapojeno do výroby – viz kapitola 4).

public function zvys_mnozstvi_spotrebniho_zbozi($zvyseni)

Zvýší množství spotřebního zboží, které hráč skladuje. Tato metoda je volána v případě výpočtu velikosti vyráběného zboží, protože vyrobené zboží navyšuje množství zboží skladovaného hráčem.

public function sniz_mnozstvi_spotrebniho_zbozi($snizeni)

Sníží množství spotřebního zboží skladované hráčem. Tato metoda je využívána v případě prodeje spotřebního zboží.

public function get_mnozstvi_volneho_casu()

Vypočte množství volného času, které měl hráč k dispozici. Tato metoda dává odlišné výsledky v průběhu přípravy na kolo a při uzavírání kola, protože v případě přípravy na kolo je známo pouze to, kolik hodin hráč odpracuje sám pro sebe, ale chybí informace o práci prodané na trhu práce ostatním hráčům. Tato metoda pak poskytuje informaci o tom, kolik času může hráč na trhu práce vlastně nabídnout – nemůže totiž na trhu práce a sám pro sebe odpracovat dohromady více časových jednotek, než má k dispozici.

V případě, že byly vypočítány příslušné transakce na trhu práce, pak metoda vrací množství jednotek volného času, které měl hráč opravdu k dispozici. Toto číslo pak již může být využito pro jako parametr hodnocení hráče.

public function get_mnozstvi_spotrebovanych_statku()

Vrátí množství zboží, které hráč v daném kole spotřeboval. V případě, že dosud nebylo kolo uzavřeno, vrátí 0, protože v takovém případě ještě nebyly provedeny transakce na trhu spotřebního zboží a hráč tedy nemohl žádné zboží spotřebovat.

public function get_pravo_zmena_konfigurace_hry()

Zjistí, zda má hráč právo měnit konfiguraci hry. V případě, že jej má, vrací 1, jinak 0.

public function get_pravo_prohlizeni_konfigurace_hry()

Zjistí, zda má hráč právo prohlížet konfiguraci hry. V případě, že jej má, vrací 1, jinak 0.

public function get_aktivita_v_aktualnim_kole()

Zjistí, zda byl hráč v aktuálním kole aktivní. Tato funkce funguje v průběhu přípravy na kolo (např. pro prognózování výsledků jsou hráči odečtena penalizace za neaktivitu) i při výpočtu hodnocení za kolo (hráč je za případnou neaktivitu s konečnou platností penalizován).

public function over_opravneni_k_akci($id_opravneni)

Zjistí, zda má uživatel právo k akci, která je předávaná jako parametr. Tento parametr musí být shodný s názvem sloupce v tabulce administrátorských práv v databázi, která informaci o tomto oprávnění poskytuje. V případě, že jej má, vrací 1, pokud jej nemá nebo oprávnění neexistuje, vrací 0.

public function get_pocet_bodu()

Vrátí počet bodů, které hráč dosud získal. Je použito funkce SUM v SQL dotazu, protože v tabulce hodnocení se ukládají vždy body získané v aktuálním kole (nikoli kumulativně).

public function over_predchozi_ucast_ve_hre()

Ověří, zda se hráč již zúčastnil hry. Tato funkce má uplatnění v případě, že hra získává informaci o loginu uživatele pomocí vnějšího autorizačního systému. V takovém případě je vždy nejdříve ověřeno, zda je vlastnost login této instance již v tabulce hráčů a pokud tomu tak není, pak je hráči nabídnuta možnost zapojit se aktivně do hry.

public function pozadavek_hrace($_GET)

Požadavek hráče je ve skutečnosti otevření webového rozhraní hry nebo kliknutí na odkaz. Požadavek reprezentuje žádost hráče o nějakou informaci. Tato metoda vyvolává událost ZADOST_O_STRANKU. Parametry události jsou položky $_GET pole, které definují, o kterou informaci hráč požádal. Tato metoda je volána hlavním programem.

10.4.2 Spravce_dat_z_formularu

Tato třída odchytává veškerá data, která hráč odeslal prostřednictvím formuláře, a na jejich základě generuje příslušné události. Součásti tohoto generování je i sestavení příslušných parametrů pro události.

10.4.2.1 Události generované třídou

Tato třída může generovat následující události:

- ODESLANI_FORMULARE_NABIDKY
- ODESLANI_FORMULARE_POPTAVKY
- ODESLANI_FORMULARE_PRODUKCE
- RESET_HRY
- KONEC_KOLA
- ZADOST_NOVEHO_HRACE_O_ZAPOJENI_DO_HRY

10.4.2.2 Vlastnosti třídy

private $data_ke_zpracovani

Zachycená data, která má objekt zpracovat. Nejčastěji se jedná o předané pole $_POST.

private $hrac

Hráč, který data odeslal. Tato vlastnost je nutná pro ověření práv v případě, že byl odeslán některý z formulářů, který vyžaduje administrátorská práva.

private $cislo_aktualniho_kola

Číslo kola, ve které se hra aktuálně nachází.

private $pole_chyb_formulare_nabidky

Toto pole obsahuje informace o chybách, které hráč provedl při odesílání formuláře nabídky. Formát tohoto pole byl vysvětlen při popisu události ODESLANI_FORMULARE_NABIDKY. Toto pole je napojeno na událost jako jeden z parametrů.

private $pole_chyb_formulare_produkce

Obdobně se jedná o pole, které bylo popsáno při definici události ODESLANI_FORMULARE_PRODUKCE.

private $spravce_konfigurace

Správce konfigurace poskytuje informace o aktuálním kole a ukládá informace o změně konfigurace hry.

10.4.2.3 Metody třídy

public function __construct($data_ke_zpracovani)

Konstruktor třídy. Jelikož je tato třída potomkem třídy Trida_generujici_udalosti, nejprve je potřeba naplnit pole_identifikatoru_udalosti. Dále jsou z předávaných dat u příslušných položek odstraněny nečíselné znaky zavoláním funkce odstran_neciselne_znaky. Vlastnosti cislo_aktualniho_kola a spravce_konfigurace jsou získány z hlavního programu.

public function spust_zpracovani_dat()

Tato metoda určí, o jaký ty formuláře se jedná, a podle toho zavolá příslušnou metodu této třídy.

private function odstran_neciselne_znaky()

Odstraní nečíselné znaky ze vstupních dat, kde jsou očekávány pouze číselní vstupy. Nahrazení probíhá pomocí regulárních výrazů.

private function zpracovani_formular_trh_poptavka()

Zpracovává formulář poptávky odeslaný hráčem. Nejprve je vytvořena instance třídy individualni_poptavka a jsou jí zadána data dle vstupních dat. Následně je tabulka uložena do databáze, přičemž je zjištěno, zda uživatel na stejném trhu a ve stejném kole poptávku již definoval. Pokud ano, je stávající příkaz aktualizován, v opačném případě je do tabulky vložen nový záznam. Dále je vyvolána událost ODESLANI_FORMULARE_POPTAVKY. Jejími parametry jsou mezní ceny tak, jak je zadal uživatel. Pro případně hromadné zpracovávání tržních příkazů je události připojen formální parametr false pro hodnotu chyba_v_datech.

private function zpracovani_formular_trh_nabidka()

Vytvoří instanci třídy individualni_nabidka a vloží do ní data zadaná uživatelem. Následně je ověřeno, zda má vlastnost třídy pole_chyb_formulare_nabidky nulovou velikost. Pokud tomu tak je, jsou data vložena do databáze (je vložena nová položka tabulky nebo aktualizována již existující). Nakonec je vyvolána událost

ODESLANI_FORMULARE_NABIDKY. Jejími parametry jsou mezní ceny a množství zadaná uživatelem. Dále pokud nebyla v datech zjištěna žádná chyba, je připojen parametr s identifikátorem chyba_v_datech o hodnotě true, v opačném případě má hodnotu false. Nakonec je připojena vlastnost třídy pole_chyb_formulare_nabidky s identifikátorem pole_chyb.

private function zpracovani_formular_produkce()

Zpracuje zadaný příkaz produkce, a pokud v něm nenajde žádnou chybu, uloží jej do databáze. Dále vyvolá událost ODESLANI_FORMULARE_PRODUKCE s parametry specifikujícími zadaný příkaz a pole_chyb.

private function zkontroluj_nabidku()

Zkontroluje nabídku podle trhu, ke kterému náleží. Ověří, zda hráč nenabízí více zboží než má na skladě, více práce než je povoleno, nebo více hotovosti než má k dispozici. V případě trhu práce je brán v úvahu i počet časových jednotek, které hráč použije v daném kole pro vlastní výrobu.

private function zkontroluj_zadane_hodiny_prace()

Slouží ke kontrole příkazu produkce. Kontroluje, zda hráč při žádné tržní ceně v důsledku zadaného příkazu produkce neodpracuje více časových jednotek, než je povoleno. Výsledek kontroly je uložen ve vlastnosti třídy pole_chyb_formulare_produkce, která je poté předána jako parametr události ODESLANI_FORMULARE_PRODUKCE (z toho vyplývá stejný formát pole, jako který je definován v popisu události).

private function zpracovani_formular_administrace()

Tato metoda je volána až po ověření hráčových práv ke změně konfigurace hry. Předá administrátorem definované nastavení hry správci nastavení, který je uloží.

private function zpracovani_formulare_editoru()

I tato metoda je volána až po ověření příslušných práv. Předá zadaná data k uložení správci.

10.5 Třídy reagující na události

10.5.1 Spravce_GUI

Tato třída řídí veškeré výstupy aplikace. Sleduje veškeré žádosti hráče, případně ověřuje jeho práva, a na základě těchto požadavků a zadaných dat generuje uživatelské rozhraní. Pro zobrazení formulářů tato třída využívá příslušných tříd ze skupiny generující formuláře.

10.5.1.1 Reakce na události

Třída reaguje na následující události:

- na událost ZADOST_O_STRANKU metodou zpracuj_zadost_o_uzivatelske_rozhrani
- na událost ODESLANI_FORMULARE_POPTAVKY metodou uloz_data_z_odeslaneho_formulare_poptavky
- na událost ODESLANI_FORMULARE_NABIDKY metodou uloz_data_z_odeslaneho_formulare_nabidky
- na událost ODESLANI_FORMULARE_PRODUKCE metodou uloz_data_z_odeslaneho_formulare_produkce

10.5.1.2 Vlastnosti třídy

private $cislo_aktualniho_kola

Číslo aktuálního kola.

private $hrac

Hráč, který aktuální instanci správce používá.

private $zpracovana_data_z_odeslanych_formularu

Tato vlastnost obsahuje data získaná z formulářů, které uživatel odeslal. Je napojena na systém řízení událostí a shromažďuje parametry vyvolané události. Tato data jsou pak zpracována a brána v úvahu při generování uživatelského rozhraní. Především jsou důležitá pro upozornění hráče na chybně zadaná data a zobrazení těchto dat.

private $prekladac

Instance třídy Prekladac, který zařizuje interpretaci všech textů, které se uživateli zobrazují.

private $spravce_konfigurace

Instance třídy Spravce_konfigurace získaný od hlavního programu.

10.5.1.3 Metody třídy

function __construct()

V konstruktoru třídy je nejprve naplněno pole_identifikatoru_udalosti identifikátory událostí, na které bude třída reagovat. Tyto události jsou spolu s příslušnými metodami popsány na začátku popisu této třídy.

public function uloz_data_z_odeslaneho_formulare_nabidky($odesilatel, $parametry)

Uloží data z formuláře nabídky, získané jako parametr události, do vlastnosti zpracovana_data_z_odeslanych_formularu.

public function uloz_data_z_odeslaneho_formulare_poptavky($odesilatel, $parametry)

Uloží data z formuláře poptávky, získané jako parametr události, do vlastnosti zpracovana_data_z_odeslanych_formularu.

public function uloz_data_z_odeslaneho_formulare_produkce ($odesilatel, $parametry)

Uloží data z formuláře produkce, získané jako parametr události, do vlastnosti zpracovana_data_z_odeslanych_formularu.

public function zpracuj_zadost_o_uzivatelske_rozhrani($odesilatel, $parametry)

Tato metoda řídí generování uživatelského rozhraní tak, aby bylo zajištěné správné vložení všech XHTML značek ve správném pořadí a aby bylo uživateli zobrazené rozhraní, které žádal a k jehož zobrazení je oprávněn. K vkládání samotné XHTML syntaxe je využito statických metod třídy html_struktura. Nejprve je vložena XHTML hlavička a logo hry, poté (pokud je uživatel přihlášen) horizontální proužek s údaji o hráči, žádané uživatelské rozhraní (za což odpovídá metoda vygeneruj_a_vloz_zadane_uzivatelske_rozhrani), menu (sestavené na základě uživatelských práv) a nakonec patička.

private function vygeneruj_a_vloz_zadane_uzivatelske_rozhrani($parametry)

Tato metoda nejprve ověří, zda je již hráč zapojen aktivně do hry (v případě, že je jeho login získán z vnějšího autorizačního systému) a pokud tomu tak není, je vygenerován formulář dle třídy Formular_registrace pro případ vnějšího autorizačního systému. Pokud se hráč do hry již dříve aktivně zapojil, je jeho požadavek analyzován. Pokud je to nutné, jsou ověřena jeho uživatelská práva a dále je zavolána metoda této třídy určená pro vložení žádaného rozhraní.

private function zjisti_a_vloz_trzni_cenu_z_minuleho_kola($id_trhu)

Tato metoda z databáze načte a vloží do rozhraní tržní cenu pro dané období z minulého kola, která usnadní hráčům odhad tržní ceny v tomto kole. Tato metoda neřeší vložení tržní ceny pro nulté kolo, která je definována administrátorem – o tom se stará Spravce_administrativnich_zaznamu, která tato data vloží do databáze pro nulté období.

private function vloz_obsah_stranky_trh($id_trhu)

Obsluhuje požadavek na stránku, která souží pro definování poptávky a nabídky na určitém trhu. Tato metoda volá metody zjisti_a_vloz_trzni_cenu_z_minuleho_kola pro vložení tržní ceny, formular_trhu_zajisteni_dat_a_vygenerovani pro vložení formulářů nabídky a poptávky a vloz_graf_vyvoje_trzni_ceny pro vložení grafu vývoje tržní ceny v čase.

private function vloz_obsah_stranky_produkce()

Obsluhuje požadavek na stránku pro zadání příkazu vlastní výroby. Formulář pro definici příkazu produkce je vložen pomocí metody zjisti_vychozi_data_pro_formular_produkce a dále je volána metoda vytvor_a_vloz_rozhodovaci_graf, která zajistí vložení formuláře a grafu vykreslujícího produkční funkci.

private function formular_trhu_zajisteni_dat_a_vygenerovani($nazev_tabulky, $oznaceni_P)

Vloží formulář poptávky nebo nabídky v závislosti na parametru nazev_tabulky. Výchozí data jsou zjištěna metodou zjisti_vychozi_data_pro_formular_poptavky nebo zjisti_vychozi_data_pro_formular_nabidky.

private function zjisti_vychozi_data_pro_formular_poptavky ($nazev_tabulky)

Tato metoda zjistí vložení výchozích dat pro formulář poptávky.

Nejprve je zjištěno, zda uživatel již v některém z předchozích kol poptávku na tomto trhu nezadal a pokud ano, jsou do pole výchozích dat zapsány ceny dle poslední poptávky na daném trhu. Nemusí jít nutně o předcházející kolo, ale i o předminulém kolo, pokud uživatel v minulém kole na tomto trhu poptávku nedefinoval. To zajistí část SQL příkazu „ORDER BY kolo DESC“. Dále je zpracována první položka z výsledku dotazu (pokud je vrácena alespoň jedna položka).

Dále je zjištěno, zda spolu s žádostí o stránku nebyl odeslán i formulář. Pokud ano, jsou znovu vložena data z tohoto formuláře – tím je zajištěno informování uživatele, že data byla v pořádku zpracována, nebo upozornění na chybu.

Další možností je načtení dříve uložené poptávky z databáze, pokud se týká daného trhu a aktuálního kola.

Všechna data jsou upravena tak, aby je třída Formular_poptavky mohl přímo a bez úprav zobrazit.

private function zjisti_vychozi_data_pro_formular_nabidky ($nazev_tabulky)

Funguje analogicky s metodou zjisti_vychozi_data_pro_formular_poptavky.

private function zjisti_vychozi_data_pro_formular_produkce()

Výchozí data pro formulář produkce jsou zjištěna z formuláře odeslaného v předchozím požadavku nebo z příkazu produkce pro aktuální kolo, pokud byl hráčem uložen již dříve.

public function zjisti_existenci_nabidky_nebo_poptavky_v_databazi($nazev_tabulky)

Tato metoda zjistí, zda se v tabulce dle parametru nazev_tabulky nachází zadaná tržní nabídka nebo poptávka daného hráče pro aktuální kolo. Pokud ano, vrátí tento příkaz v asociativním poli ve formátu pro zobrazení třídami Formular_nabidky a Formular_poptavky.

function zjisti_existenci_prikazu_produkce_v_databazi()

Zjistí, zda hráč v daném kole již zadal příkaz aktuální produkce. Pokud ano, vrátí jej v poli pro zobrazení třídnou Formluar_produkce.

private function vytvor_a_vloz_rozhodovaci_graf()

Nejprve je pomocí třídy Formular_grafu_produkce vložen graf, ve kterém si uživatel navolí, hodnota kterého z výrobních faktorů je pro graf fixní (při obou variabilních faktorech by graf musel být trojrozměrný), množství tohoto fixního výrobního faktoru a maximum osy x grafu. Tento graf používá metodu GET, aby si hráč mohl do záložek uložit určité nastavení grafu a nemusel je nastavovat opakovaně.

Rozhodovací graf pomůže uživateli s rozhodováním o velikosti výroby. Graf je vložen pomocí souboru grafy/individualni_produkcni_funkce.php. Tento soubor je volán pomocí GET parametrů, které jsou získány z dat formuláře grafu produkce.

private function vloz_graf_vyvoje_trzni_ceny($id_trhu)

Pokud se hra nenachází v prvním kole, zajistí tato metoda vložení grafu, který zobrazuje vývoj ceny na daném trhu. Vložení zajišťuje soubor grafy/vyvoj_trzni_ceny.php, GET parametrem při volání je vlastnost objektu id_tridy.

private function vloz_obsah_stranky_prehled()

Reakce na požadavek s id prehled. Tato metoda vloží přehled všech transakcí, které byly provedeny v minulém kole. Pro získávání informací je využívána datová třída Spravce_informaci_pro_prehled. Tato metoda vkládá následující informace:

- o provedených tržních transakcích
- o vyrobeném zboží
- o změně finanční hotovosti, o vyplacených a přijatých úrocích a jistinách
- o získaných bodech

Ve všech případech se jedná o informace ve vztahu k minulému kolu.

private function vloz_obsah_stranky_prognoza($parametry)

Reakce na požadavek s id prognoza. Poskytuje podklad především pro kontrolu finančních toků, aby si hráč ověřil, zda se v následujícím jeho finanční hotovost dostane do mínusu. Tato stránka je založena na dosazení tržních cen z minulého kola do poptávek a nabídek současného kola.

Podle seznamu platných trhů je pro každý trh určeno, kolik bude při zadané tržní ceně na trhu utraceno a za kolik bude naopak prodáno. V případě trhu finančních aktiv je pak

samozřejmě při prodeji hotovost odečtena a naopak. V případě trhu práce je uchováno, kolik hodin práce by hráč odpracoval (z důvodu dosazení do hodnotící funkce).

Dále tato stránka poskytuje informaci o vyplacených i přijatých splátkách úvěrů a úroků. Tato data jsou získána pomocí třídy Spravce_informaci_pro_prehled. Dále jsou všechna prognózovaná data vložena do funkce pro hodnocení hráčů a je vypočítáno, kolik by hráč za dané kolo obdržel bodů.

private function vloz_obsah_stranky_seznam_hracu($parametry)

Tato funkce vypíše seznam všech hráčů z databáze. Pokud hráč disponuje oprávněním rozsirene_prohlizeni_seznamu_hracu, jsou loginy hráčů odkazy a je možno si po kliknutí zjistit podrobnější informace o daném hráči.

10.5.2 Spravce_administrativnich_zaznamu

Tato třída převádí stavové položky (skladované položky včetně hotovosti a kapitálové zboží ve výrobě) z aktuálního do následujícího kola v případě, že byla vyvolána událost KONEC_KOLA.

10.5.2.1 Reakce na události

- na událost KONEC_KOLA reaguje metoda proved_aministrativni_zaznam_o_konci_kola
- na událost RESET_HRY reaguje metoda smaz_data_o_prubehu_hry
- na událost ZADOST_NOVEHO_HRACE_O_ZAPOJENI_DO_HRY reaguje metoda zapoj_hrace_do_hry

10.5.2.2 Vlastnosti třídy

private $cislo_kola

Číslo aktuálního kola

private $spravce_konfigurace

Odkaz na správce z hlavního programu.

10.5.2.3 Metody třídy

public function __construct()

Konstruktor třídy. Naplní pole_identifikatoru_udalosti a nastaví vlastnosti spravce_konfigurace a cislo_kola z hlavního programu.

public function proved_aministrativni_zaznam_o_konci_kola($odesilatel, $parametry)

Tato metoda nejprve převede všechny stavové položky do dalšího kola.

U kapitálu ve výrobě je během převodu provedena amortizace tak, že je do dalšího kola převeden pouze násobek čísla, které je doplňkem do jedné k amortizačnímu faktoru a množství kapitálového zboží ve výrobě v současném kole. Výsledné číslo je před zápisem do databáze zaokrouhleno dle matematických pravidel.

public function smaz_data_o_prubehu_hry ($odesilatel, $parametry)

Tato metoda smaže veškeré tabulky pro ukládání tržních nabídek, poptávek, příkazů výroby, stavových položek, o provedených transakcích a statistice kol. Jako aktuální kolo je nastaveno první kolo a všechny stavové položky jsou nastaveny na své výchozí hodnoty. Nakonec jsou nastaveny výchozí tržní ceny dané administrátorem jako ceny pro kolo 0 v tabulce pro vývoj tržních cen.

public function zapoj_hrace_do_hry ($odesilatel, $parametry)

V případě, že se pomocí vnějšího autorizačního systému do hry přihlásí první hráč a projeví zájem o zapojení do hry, je vyvolána událost ZADOST_NOVEHO_HRACE_O_ZAPOJENI_DO_HRY a tato metoda na ni reaguje. Vloží login hráče do tabulky hráčů a nastaví mu výchozí hodnoty u skladovaných položek.

10.5.3 Spravce_trhu

Tato třída zajišťuje provedení transakcí na všech aktivních trzích po ukončení kola.

10.5.3.1 Reakce na události

- nastane-li událost KONEC_KOLA, je provedena metoda zobchoduj_polozky_na_vsech_trzich

10.5.3.2 Vlastnosti třídy

private $pole_trhu

Toto pole obsahuje instance třídy Trh reprezentující všechny trhy, které jsou v aktuální hře aktivní, či přesněji řečeno na kterých administrátor povolil obchodování.

private $kolo

Aktuální kolo hry.

10.5.3.3 Metody třídy

public function __construct()

Konstruktor třídy. Naplní pole_identifikatoru_udalosti a nastaví vlastnost kolo.

public function zobchoduj_polozky_na_vsech_trzich ($odesilatel, $parametry)

Načte pomocí objektu spravce_konfigurace seznam identifikátorů trhů, které jsou aktivní. Pro každý trh vytvoří instanci, která tento trh reprezentuje, a uloží ji do pole_trhu. U každé instance je pak volána metoda zobchoduj_polozky_na_trhu, která zajistí výpočet tržní ceny a provedení všech tržních transakcí. Nakonec je zavolána metoda zaznam_trznich_cen.

private function zaznam_trznich_cen()

Tato metoda zaznamená do tabulky vyvoj_trznich_cen tržní ceny na všech aktivních trzích pro právě uzavírané kolo.

10.5.4 Spravce_produkce

Tato třída zajistí výpočet velikosti produkce pro všechny hráče a vyrobené zboží přičte ke skladovaným položkám.

10.5.4.1 Reakce na události

- na událost KONEC_KOLA třída reaguje metodou pripis_vysledky_produkce_vyrobcum

10.5.4.2 Vlastnosti třídy

private $kolo

Aktuální kolo, ve které se hra nachází.

private $pole_individualnich_produkcnich_funkci

Toto pole obsahuje instance třídy Individualni_produkcni_funkce pro každý příkaz produkce, který byl pro aktuální kolo zadán.

10.5.4.3 Metody třídy

function __construct()

Konstruktor třídy. Je v něm naplněno pole_identifikatoru_udalosti a zadána vlastnost kolo.

private function urci_parametry_individualnich_produkcnich_funkci()

Tato metoda nejprve z tabulky prikazy_produkce získá všechny příkazy produkce. Dále pro každý příkaz zjistí množství kapitálového zboží ve výrobě a množství práce, které je součtem vlastní práce hráče a práce nakoupené hráčem na trhu práce. Následně může být vytvořena instance třídy individualni_produkcni_funkce a uložena do pole_individualnich_produkcnich_funkci.

public function pripis_vysledky_produkce_vyrobcum()

Nejprve volá metodu urci_parametry_individualnich_produkcnich_funkci, který naplní pole_individualnich_produkcnich_funkci daty pro aktuální kolo.

Projde celé pole_individualnich_produkcnich_funkci a u každé položky zavoláním metody vygeneruj_a_proved_dotaz_na_upravdu_databaze připíše vyrobené zboží ke skladovým zásobám hráče.

10.5.5 Spravce_uveru

Tato třída zajišťuje odečtení úroků a zapůjčených částek dlužníkům a připsání úroků a navrácených úspor věřitelům. Tyto informace jsou čerpány na základě záznamů o transakcích z trhů finančních aktiv.

10.5.5.1 Reakce na události

- při vyvolání události KONEC_KOLA je volána metoda zuctuj_existujici_uvery

10.5.5.2 Vlastnosti třídy

private $aktualni_kolo

Číslo kola, které je aktuální (případně které je právě uzavíráno).

10.5.5.3 Metody třídy

function __construct()

Konstruktor třídy, naplní pole_identifikatoru_udalosti a nastaví vlastnost aktualni_kolo.

public function zuctuj_existujici_uvery($odesilatel, $parametry)

Tato metoda získá od objektu spravce_konfigurace seznam trhů, na kterých administrátor hry povolil obchodování, a vybere z nich trhy kapitálu. Pro každý z trhů kapitálu pak je volána metoda zuctuj_uvery_z_daneho_trhu, jako parametr je při volání zvolen název trhu.

private function zuctuj_uvery_z_daneho_trhu($nazev_trhu)

Každý trh má vlastní dvojici tabulek, do které se ukládají záznamy o úvěrech, které hráč přijal, a o úsporách, které zapůjčil na daném trhu. Tato metoda projde nejprve tabulku úvěrů pro daný trh.

Podmínka ($kolo < $this->aktualni_kolo) && (($kolo + $delka_uveru) >= $this->aktualni_kolo)) zajišťuje vybrání všech úvěrů, u kterých se má v aktuálním kole zaplatit úrok. První část podmínky zjistí, zda nebyl úvěr získán v aktuálním kole, a druhá zda již nebyl splacen, tj. zda od získání úvěru neuplynula doba delší, než jaká je jeho doba splatnosti. Tato podmínka vybere i úvěr, který má být v tomto kole splacen. Úrok je pak odečten.

Úrok zde může vyjít jako reálné číslo. Toto reálné číslo je pak hráči z jeho hotovosti odepsáno. Pokud by toto číslo bylo zaokrouhleno, systém by již negarantoval při splácení úvěrů stále stejné množství peněz v ekonomice, protože objem vyplacených úroků by mohl být odlišný od objemu přijatých úroků (případně by tato možnost musela být ošetřena např. náhodných přerozdělením části úroků). Volaná metoda sniz_mnozstvi_kapitalu u třídy Hrac akceptuje reálné číslo jako svůj parametr.

Druhá podmínka ($kolo + $delka_uveru == $this->aktualni_kolo) pak určí úvěry, které mají být v aktuálním kole splaceny. V těchto případech je hráči odečtena hotovost o velikosti jistiny.

10.5.6 Spravce_hodnoceni

Na základně zhodnocení chování v daném kole vypočte tato třída každému hráči jeho bodový zisk (nebo ztrátu) dle vzorce, který zadá administrátor hry.

10.5.6.1 Reakce na události

- na událost KONEC_KOLA reaguje metodou ohodnot_hrace

10.5.6.2 Vlastnosti třídy

private $kolo

Číslo aktuálního kola.

private $spravce_konfigurace

Spravce_konfigurace získaný z hlavního programu.

10.5.6.3 Metody třídy

function __construct()

Konstruktor třídy, naplní pole_identifikatoru_udalosti a nastaví vlastnosti třídy.

public function ohodnot_hrace($odesilatel, $parametry)

Načte z databáze loginy všech hráčů a pro každého hráče vytvoří instanci třídy Hrac – tato třída zde tedy funguje jako datová třída. Dále jsou zjištěny údaje nutné pro ohodnocení hráče. Hodnocení je vypočítáno s využitím statické metody hodnotici_funkce. Výsledné hodnocení je pak zapsáno do databáze. Do databáze se vždy zapisují body získané za aktuální kolo, pro získání celkového hodnocení je tedy nutné využít SQL příkaz SUM (nebo provést výpočet bodů v PHP skriptu).

public static function hodnotici_funkce($kolo, $mnozstvi_volneho_casu, $mnozstvi_spotrebovanych_statku, $zaporny_penezni_zustatek, $neaktivni)

Jedná se o statickou funkci. Tato funkce tedy může být volána bez nutnosti vytvářet instanci třídy Spravce_hodnoceni, což je využíváno např. v přehlednu prognózy finančních toků. Tato metoda si nejprve z hlavního programu získá spravce_konfigurace (samozřejmě nemůže využívat vlastnosti třídy) a získá od něj vzorec pro výpočet hodnocení.

Ve vzorci jsou pak klíčová slova nahrazena parametry metody. Každý z parametrů má vlastní klíčové slovo, které administrátor vloží do vzorce namísto parametru.

10.5.7 Spravce_zaznamu_o_aktivite

Úkolem této třídy je vést záznamy o aktivitě uživatelů, která pak může být součástí jejich hodnocení.

10.5.7.1 Reakce na události

Tato třída nemá přímo určenou událost, na kterou má reagovat, ale z principu může reagovat na libovolnou událost, která je vyvolána třídou Hrac. Určení události, na které bude třída reagovat, vlastně může být definicí toho, po provedení které akce bude hráč v daném kole označen jako aktivní a po které nikoli. K reakci na tyto události slouží metoda zaznamenej_aktivitu.

10.5.7.2 Vlastnosti třídy

private $hrac

Hráč, který událost vyvolal.

private $cislo_aktualniho_kola

Číslo aktuálního kola a tedy kola, pro které bude hráč označen jako aktivní.

10.5.7.3 Metody třídy

public function __construct()

V konstruktoru třídy jsou určeny události, které třída bude zaznamenávat. Ale pro všechny tyto události je ještě nutné v hlavním programu vygenerovat ovladače a ty navěsit na příslušné události.

public function zaznamenej_aktivitu($odesilate, $parametry)

Tato metoda provede záznam obsahující login a kolo do tabulky aktivita_hracu. Díky tomu je hráč pro dané kolo trvale označen jako aktivní a nemůže tedy být bodově penalizován za neaktivitu. K aktivitě je zaznamenán i příslušný čas.

Pokud byl hráč označen jako aktivní pro aktuální kolo již dříve, tato metoda neprovádí do databáze žádný záznam.

10.6 Datové třídy

Jak již bylo řečeno v předchozích subkapitolách, některé třídy v aplikaci mají schopnost generovat události a jiné na tyto události reagují. Část tříd v aplikaci však nemá ani jednu z těchto schopností. Tyto třídy jsou označeny jako datové třídy, jsou prostředníkem mezi třídami pracujícími s událostmi a databází. V souladu s filosofií objektové programování tyto třídy reprezentují nějaký objekt v realitě (či přesněji v ekonomickém modelu hry) a poskytují informace o tomto objektu. Tyto třídy jsou

schopny samy si načíst z databáze potřebná data a ostatní metody mohou prostřednictvím veřejných metod dodávat třídám nová data, případně vytvářet objekty pro data, která v databázi dosud zanesená nejsou. Třídy poté mají schopnost aktualizovaná data uložit do databáze.

10.6.1 Individualni_nabidka

Tato třída reprezentuje individuální nabídku jednoho hráče. Obsahuje informace o tom, kolik které komodity je uživatel za různé tržní ceny nabídnout. Je definována rozhodnutím hráče, které zadává do uživatelského rozhraní. Analogickou k této třídě je třída Individualni_poptavka.

10.6.1.1 Vlastnosti třídy

private $pole_meznich_cen

Klíčem tohoto pole je mezní cena a hodnotou nabízené množství. Od mezní ceny je směrem nahoru nabízeno množství, které je její hodnotou.

Příklad tabulky:

40	50
20	30

Pro ceny 20 až 39 bude nabízeno 30 položek a pro ceny 40 a vyšší bude nabízeno 50 položek. Pro cenu 19 a nižší nebude nabízeno nic.

private $pole_individualni_nabidky

Je získána z pole_meznich_cen. Klíčem u této tabulky je opět cena a hodnotou množství. Položky tabulky jsou určeny pro všechny ceny v intervalu od nejnižší po nejvyšší uživatelem definované ceny. Tato operace je provedena z důvodu vytváření tržní poptávky, kdy dochází pouze k agregaci těchto tabulek. Jedná se tedy o velmi transparentní algoritmus.

private $login

Řetězec obsahující login hráče, každá individuální nabídky musí připadat právě jednomu hráči.

private $kolo

Celé číslo, které označuje kolo, pro které byla individuální nabídky definována.

10.6.1.2 Metody třídy

function __construct($login, $kolo)

Konstruktor třídy, parametry nastavují hodnoty login a kolo instance po celou dobu její existence.

public function pridej_parametr_nabidky($mezni_cena, $mnozstvi)

Přidá další položku do pole pole_meznich_cen. Pole může mít neomezený počet položek a po každém přidání probíhá asociativní řazení podle klíče (tedy seřazení hodnot podle klíče při zachování vztahů mezi klíčem a jemu příslušnou hodnotou) funkcí ksort.

public function vypocti_nabidku()

Vytvoří a dodá správné hodnoty do pole_individualni_nabidky. Postupně od nejvyšší hodnoty prochází pole_meznich_cen postupně směrem k nižším hodnotám funkcí prev a pro jednotlivé ceny dosazuje hodnoty dle aktuální položky v pole_meznich_cen. Dosazování do pole končí při ceně rovné nejnižší ceně z pole_meznich_cen.

public function urci_nabizene_mnozstvi_pri_dane_cene ($cena)

Vrátí množství, které by bylo hráčem nabízené při ceně v hodnotě parametru funkce. Čerpá z pole_individualni_nabidky. Pro všechny ceny vyšší než nejvyšší cena v tomto poli je nabízené množství rovné nabízenému množství nejvyšší ceny v proměnné pole_individualni_nabidky.

public function urci_pokracene_nabizene_mnozstvi_na_trhu ($cena, $prepocitavaci_koeficient)

Nejprve zavolá vlastní funkci urci_nabizene_mnozstvi_pri_dane_cene s parametrem cena, který jí byl předán. Následně vrácenou hodnotu vynásobí proměnnou prepocitavaci_koeficient a tu zaokrouhlí směrem dolů. Výsledek operace je předán volající třídě jako návratová hodnota funkce.

public function vytvor_SQL_definici($nazev_tabulky, $vkladani)

Vygeneruje a vrátí SQL dotaz pro zapsání individuální nabídky do databáze. Parametr nazev_tabulky označuje tabulku, do které bude záznam vložen. Parametr vkladani je logická hodnota. Pokud je rovna true, je vrácen dotaz pro vložení, tedy SQL příkaz INSERT INTO. V opačném případě uživatel aktualizuje v databázi již uloženou definici nabídky a je využit SQL příkaz UPDATE.

10.6.2 Trzni_nabidka

Reprezentuje tržní nabídku na jednom trhu a vztahuje se k jednomu kolu. Je agregací individuálních poptávek jednotlivých hráčům. Tato třída si umí načíst data o individuálních poptávkách z databáze a provést jejich agregaci.

10.6.2.1 Vlastnosti třídy

private $pole_individualnich_nabidek

Pole obsahující instance tříd individualni_nabidka. Jsou v něm všechny individuální poptávky na daném trhu a v daném kole.

private $pole_trzni_nabidky

Jde o pole koncipované stejně, jako pole_individualni_nabidky u individuální nabídky. Klíčem je tržní cena a hodnotou nabízené množství, které jsou všichni prodávající na daném trhu a v daném kole ochotni za danou cenu nabídnout. Pole je zdola omezenou cenou, pro kterou prodávající nenabízejí nic, a shora cenou, od které již nedochází k žádným změnám v tržním nabízeném množství. Pro jakoukoli vyšší cenu je nabízené množství rovno nabízenému množství u této nejvyšší ceny popsané v této vlastnosti.

private $kolo

Označuje kolo, ke kterému se daná tržní nabídka vztahuje.

private $nazev_tabulky

Název tabulky, pod kterým jsou uložena v databázi data o individuálních nabídkách. Lze z něj získat i název trhu.

10.6.2.2 Metody třídy

function __construct($kolo, $nazev_tabulky)

Konstruktor třídy, který nastaví hodnoty vlastností kolo a nazev_tabulky dle parametrů.

public function nacti_data_nabidek_z_databaze()

Získá potřebná data z databáze a na jejich základě vytvoří pole_individualnich_nabidek. Metoda přímo přistupuje k databázi na základě údajů v konfiguračním souboru.

public function vytvor_trzni_nabidku()

Vytvoří tržní nabídku z individuálních nabídek uložených v poli pole_individualnich_nabidek. Nejdříve je vypočten rozsah pole, pro který je potřeba jej

definovat. Poté metoda projde celé pole_individualnich_nabidek a pro každou z cen přičte k tržnímu nabízenému množství individuální nabízené množství oné nabídky.

10.6.3 Individualni_poptavka

Reprezentuje individuální nabídku hráče. Poskytuje tedy odpověď na otázku, kolik je hráč ochoten zakoupit při různých cenách.

10.6.3.1 Vlastnosti třídy

private $pole_meznich_cen

Klíčem v poli je mezní cena, od které se mění poptávané množství, a hodnotou poptávané množství. Mechanismus funguje opačně než u nabídky, hráč mění rozhodnutí od mezní ceny směrem dolů. Následuje příklad:

20	120
10	200

Pro cenu 20 a vyšší nebude poptáváno nic, pro cenu v intervalu 19 až 10 bude poptáváno 120 položek, pro cenu 9 až 1 bude poptáváno 200 položek.

private $pole_individualni_poptavky

Je pole získané z pole_meznich_cen. Klíčem je opět cena a hodnotou množství poptávané při ceně, která je klíčem. Tabulka je definovaná v rozsahu cen od 1 až po nejvyšší cenu z pole_meznich_cen.

private $login

Login uživatele, kterému poptávka náleží.

private $kolo

Kolo, ke kterému se poptávka vztahuje.

10.6.3.2 Metody třídy

function __construct($login, $kolo)

Konstruktor třídy, který nastaví login a kolo dle hodnot parametrů.

public function pridej_parametr_poptavky($mezni_cena, $mnozstvi)

Přidání položky do pole_meznich_cen. Počet položek, které mohou být přidány, není nijak omezen.

public function je_klesajici()

Ověří, zda je poptávka klesající – tedy zda při rostoucí ceně uživatel poptává stále menší množství statku. U racionálně jednajícího hráče je v našem modelu poptávka vždy klesající.

public function vypoctiPoptavku()

Z hodnot získaných z pole_meznich_cen vytvoří pole_individualni_poptavky naplněné příslušnými hodnotami. Toto pole opět vzniká z důvodu transparentního vytváření tržní poptávky agregací individuálních poptávek.

public function urci_poptavane_mnozstvi_pri_dane_cene($cena)

Určí, kolik položek bude poptáváno při ceně rovné parametru cena. Toto poptávané množství je návratovou hodnotou funkce.

public function urci_pokracene_poptavane_mnozstvi_na_trhu ($cena, $prepocitavaci_koeficient)

Tato metoda je využívána při převisu poptávky nad nabídkou na trhu. Nejprve zavolá funkci urci_poptavane_mnozstvi_pri_dane_cene a předá mu parametr cena. Navrácenou hodnotu vynásobí parametrem prepocitavaci_koeficient a provede zaokrouhlení směrem dolů. Výsledek tohoto postupu je návratovou hodnotou metody.

public function vytvor_SQL_definici($nazev_tabulky, $vkladani)

Vytvoří SQL dotaz, který vloží data o individuální nabídce do databáze, a tento dotaz předá jako návratovou hodnotu.

Parametr nazev_tabulky označuje tabulku, do které bude záznam vložen. Parametr vkladani je logická hodnota. Pokud je rovna true, je vrácen dotaz pro vložení, tedy SQL příkaz INSERT INTO. V opačném případě uživatel aktualizuje v databázi již uloženou definici nabídky a je využit SQL příkaz UPDATE.

10.6.4 Trh

Objektová třída Trh reprezentuje trh tak, jak jsme jej popsali v předchozích kapitolách. Instance třídy je vytvořena na konci každého kola a v případě, že jsou data poskytovaná touto třídou požadována administrátorem prostřednictvím administrátorského rozhraní. Vytvořená instance musí vždy reprezentovat určitý trh definovaný v modelu. Tato informace je určená již při vytváření instance.

Trh si umí vygenerovat svou tržní nabídku a poptávku, z nich vypočítat tržní cenu a provést všechny transakce, ke kterým na trhu dojde.

10.6.4.1 Vlastnosti třídy

public $pole_individualnich_poptavek

Jde o pole instancí třídy Individualni_poptavka, které obsahuje veškeré hráči definované individuální poptávky v daném kole. Jsou využity při vytváření příkazů k nákupu.

public $pole_individualnich_nabidek

Pole instancí třídy Individualni_nabidka, které obsahuje (obdobně jako pole_individualnich_poptavek) veškeré uživateli zadané nabídky v daném kole.

private $nazev_trhu

Jednoznačně definuje daný trh, který je definován v konstruktoru a později již nesmí být měněna. Ostatní třídy ji mohou číst. Od tohoto identifikátoru se odvíjí i názvy tabulek v databázi pro ukládání nabídek a poptávek uživatelů.

private $trzni_poptavka

Je instancí třídy Trzni_nabidka. Reprezentuje tržní nabídku, která náleží konkrétnímu trhu. Třída Trh si tuto instanci vytváří sama, pro výpočet tržní ceny totiž poskytuje klíčové informace.

private $trzni_nabidka

Je instancí třídy Trzni_poptavka. Reprezentuje tržní nabídku. Stejně jako instance třídy Trzni_poptavka je vytvořena instancí třídy Trh a využita k výpočtu tržní ceny.

private $trzni_cena

Je cenou, za kterou je v aktuálním kole na tomto trhu obchodována konkrétní komodita. Na trhu kapitálových aktiv je úrokovou mírou vyjádřenou v procentech. Instance třídy Trh si tuto cenu sama vypočítá. Ostatní třídy mohou tuto hodnotu číst.

private $kolo

Celé číslo, které označuje aktuální kolo. I když existence konkrétního trhu není obecně vázána na konkrétní kolo, jedna vytvořená instance načítá data vždy pouze pro jedno kolo (bylo by nesmyslné např. na základě tržní poptávky v 5. kole a tržní nabídky v 6. kole počítat tržní cenu). Proto je u třídy Trh nutné definovat kolo již při vytvoření

instance. Stejně jako nazev_trhu musí zůstat nezměněna po celou dobu existenci instance.

private $nabizene_mnozstvi_pri_trzni_cene

Po výpočtu tržní ceny lze zjistit, jak velké množství komodity budou všichni hráči nabízet. Získaná informace je uložena v této proměnné. Celočíselná hodnota.

private $poptavane_mnozstvi_pri_trzni_cene

Lze vypočítat i množství komodity, kterou budou hráči poptávat, rovněž jde o celočíselnou hodnotu. Rozdíl předcházející a této hodnotu označuje převis nabídky nad poptávkou respektive poptávky nad nabídkou.

10.6.4.2 Metody třídy

function __construct($nazev_trhu, $kolo)

Konstruktor třídy. Instanci je přidělen nazev_trhu a kolo, které instanci identifikují.

Dalším metodám třídy Trh již nejsou předávány žádné parametry, protože veškerá další potřebná data jsou již načtena přímo z databáze, případně si je třída Trh sama zjistí vytvořením a voláním metod dalších tříd.

public urci_trzni_cenu()

Nejprve vytvoří tržní nabídku a poptávku pro konkrétní trh, které jsou nutné k určení tržní ceny. Ty jsou uloženy do proměnných instance trzni_poptavka a trzni_nabidka, aby byly k dispozici metodě vypocti_trzni_cenu(). Metoda vypocti_trzni_cenu() je následně zavolána.

private vypocti_trzni_cenu()

Tato metoda je volána metodami urci_trzni_cenu() a zobchoduj_polozky_na_trh(). Od instancí tříd a získá pole_trzni_nabidky a pole_trzni_poptavky. Klíčem u těchto polí je tržní cena (celočíselná hodnota) a hodnotou je nabízené resp. poptávané množství, které budou všichni hráči při dané ceně nabízet resp. poptávat.

Při využití metody maximálního obratu, která je popsána ve čtvrté kapitole, jsou obě pole procházena od 1 do nejvyšší ceny, za kterou je tržní poptávané množství nenulové. Poté již nemá procházení pole smysl.

Pokud je minimálního rozdílu dosaženo pro více hodnot, je tržní cena rovna ceně ve středu tohoto intervalu. Pokud střed intervalu nemůže být určen celočíselně (interval má

v tom případě sudý počet položek), je cena získána zaokrouhlením ceny ve středu intervalu nahoru. Z pole_trzni_nabidky a pole_trzni_poptavky metoda určí a uloží proměnné třídy nabizene_mnozstvi_pri_trzni_cene a poptavane_mnozstvi_pri_trzni_cene a také určenou tržní cenu do proměnné trzni_cena.

public zobchoduj_polozky_na_trh()

Zodpovídá za provedení transakcí na trhu. Nejdříve se zjišťuje, zda při vypočtené tržní ceně dochází k převisu poptávky nad nabídkou nebo nabídky nad poptávkou. V prvním případě nebude část hráčů uspokojená (neboli dojde ke krácení jejich požadavků) v plné výši dle svých příkazů, ve druhém případě se tak stane části nabízejících hráčů.

Výši zkrácení požadavků určí proměnná prepocitavaci_koeficient. V případě, že nabizene_mnozstvi_pri_trzni_cene je menší než poptavane_mnozstvi_pri_trzni_cene, bude prepocitavaci_koeficient roven podílu proměnných nabizene_mnozstvi_pri_trzni_cene a poptavane_mnozstvi_pri_trzni_cene. Poté jsou získány, pomocí proměnných trzni_nabidka a trzni_poptavka, instance tříd Individualni_nabidka a Individualni_poptavka, které odpovídají danému trhu v daném kole. Pro každou z těchto instancí je zjištěno nabízené a poptávané množství při dané tržní ceně. Na základě těchto zjištění jsou vytvářeny příkazy nákupu a příkazy prodeje.

Instance třídy prikaz_prodeje jsou vytvářeny s parametrem prodane_mnozstvi ve výši nabízeného množství hráčem. U poptávek je využita funkce urci_pokracene_poptavane_mnozstvi_na_trhu(), která vrátí individuální poptávané množství při dané tržní ceně násobené proměnnou prepocitavaci_koeficient a zaokrouhlenou směrem dolů.

Celkové množství musí být rovno nabízenému, což nám dosavadní postup nezajišťuje. Suma krácených poptávaných množství by byla rovna poptávanému množství, ale vlivem zaokrouhlování se objem nakoupeného množství sníží. Proto jsou z pole příkazů prodeje náhodně vybírány položky a u nich je zvýšena hodnota proměnné mnozstvi o 1, dokud nedojde k vyrovnání obou hodnot.

Poté všechny vygenerované příkazy provedou změny v databázi.

Postup při převisu nabídky nad poptávkou je analogický, ale namísto poptávaných jsou krácena nabízená množství.

public vypis_tabulku()

Vypíše tabulku, kde pro každou cenu je uvedeno nabízené a poptávané množství.

10.6.5 Prikaz_nakupu

Lze si jej představit jako např. na jedné straně úhradu zboží nakupujícím a současně zapsáním nakoupeného statku do evidence. Zajistí zaznamenání provedení tržní transakce u nakupujícího úpravou příslušných dat v databázi.

10.6.5.1 Vlastnosti třídy

private $nazev_trhu

Označuje trh, na kterém je nákup uskutečněn. Z něj lze snadno získat název příslušné tabulky v databázi.

private $kolo

Kolo, ve kterém je nákup uskutečněn.

private $mnozstvi

Nakoupené množství.

private $trzni_jednotkova_cena

Jednotková cena, za kterou je nákup uskutečněn.

private $kupujici

Login kupujícího. Spolu s proměnnou nazev_trhu a kolo jednoznačně identifikují každý příkaz.

10.6.5.2 Metody třídy

function __construct($nazev_trhu, $kolo, $mnozstvi, $trzni_jednotkova_cena, $kupujici)

Konstuktor, který nastaví hodnoty všech proměnných třídy. Tyto hodnoty se (s výjimkou množství) nebudou měnit.

public function zvys_nakoupene_mnozství_o_jednotku()

Zvýší hodnotu atributu mnozstvi a jednotku. Je využíváno při náhodném rozdělování zbylých komodit na trhu.

public function vygeneruj_a_proved_prikaz_pro_upravu_databaze()

Vygeneruje a provede SQL příkazy pro vhodnou úpravu dle příslušného trhu.

private function vygeneruj_a_proved_prikaz_pro_snizeni_mnozstvi_kapitalu()

Jelikož SQL příkaz pro snížení množství kapitálu je u trhů spotřebního a kapitálového zboží a práce stejný, je umístěn ve zvláštní metodě (aby nedocházelo k opakování identického kódu). Metoda je volána metodou vygeneruj_a_proved_prikaz_pro_upravu_databaze(), z vnějšku není přístupná.

10.6.6 Prikaz_prodeje

Obdobně jako Prikaz_nakupu, tato hodnota reprezentuje akce provedené o transakci na trhu – připsání hotovosti a odečtení prodaných položek z evidence.

10.6.6.1 Vlastnosti třídy

private $nazev_trhu

Označuje trh, ke kterému se příkaz vztahuje.

private $kolo

Číslo kola, ve kterém k prodeji dochází.

private $mnozstvi

Definuje množství nakoupeného zboží, počet hodin nakoupené práce nebo velikost poskytnutého úvěru.

private $trzni_jednotkova_cena

Jednotková cena, za kterou je prodej uskutečněn. Je určena trhem.

private $prodavajici

Login prodávajícího.

10.6.6.2 Metody třídy

function __construct($nazev_trhu, $kolo, $mnozstvi, $trzni_jednotkova_cena, $prodavajici)

Konstruktor, nastaví všechny vlastnosti třídy.

public function zvys_prodane_mnozství_o_jednotku()

Metoda zvyšuje vlastnost mnozstvi o 1. Používá se při náhodném rozdělování při převisu nabídky nad poptávkou.

public function vygeneruj_a_proved_prikaz_pro_upravu_databaze()

Vygeneruje a provede dotazy databázi, které zaznamenají nutné informace o uskutečněné transakci.

10.6.7 Prikaz_produkce

Zastupuje příkaz k produkci definovaný hráčem. Přímo souvisí s nastavením produkce v uživatelském rozhraní a obsahuje tedy informaci o vlastní práci a druhu zboží, které bude vyráběno. Na rozdíl od výše popsaných příkazů je vytvořen již v průběhu hry. Hráč totiž již v průběhu přesně ví, kolik hodin bude pro sebe pracovat a které zboží bude vyrábět (naproti tomu neví, kolik práce nakoupí, protože neví, jaká bude tržní cena práce).

10.6.7.1 Vlastnosti třídy

private $hodin_prace

Počet hodin práce, které hráč využije pro vlastní výrobu.

private $druh_zbozi

Druh zboží, které bude hráč vyrábět. 1 značí spotřební zboží a 2 kapitálové zboží. Stejné označení využívá i třída Individualni_produkcni_funkce.

10.6.7.2 Metody třídy

function __construct($hodin_prace, $druh_zbozi)

Konstruktor, nastaví všechny vlastnosti třídy.

public function vytvor_SQL_definici($hrac, $kolo, $vkladani)

Vytvoří SQL definici pro záznam příkazu produkce do databáze. Tento příkaz je návratovou hodnotou metody a není metodou vykonán.

10.6.8 Individualni_produkcni_funkce

Reprezentuje produkční funkci jednoho hráče.

10.6.8.1 Vlastnosti třídy

private $login

Login hráče, kterému vytvořená produkční funkce náleží.

private $mnozstvi_prace

Počet hodin práce, které do produkční funkce vstupuje.

private $mnozstvi_kapitaloveho_zbozi

Množství kapitálového zboží, které do produkční funkce vstupuje.

private $kolo

Kolo, ve kterém dochází produkci. Tento parametr je nutný z důvodu exogenního technologického vývoje.

private $druh_zbozi

Číselná hodnota označuje vyráběné zboží. 1 pro spotřební zboží a 2 pro kapitálové zboží.

private $velikost_produkce

Vypočtená velikost produkce pro zadané hodnoty.

10.6.8.2 Metody třídy

function __construct($login, $mnozstvi_prace, $mnozstvi_kapitaloveho_zbozi, $kolo, $druh_zbozi)

Konstruktor trvale nastaví všechny vlastnosti třídy s výjimkou velikosti produkce, kterou si funkce sama vypočítá. Výpočet je proveden při vytvoření instance – v konstruktoru je volání metody vypocti_velikost_produkce().

private function vypocti_velikost_produkce()

Provede výpočet velikosti produkce. K výpočtu využije statickou metodu třídy produkcni_funkce, jako parametry jí předá vlastnosti instance.

public function vygeneruj_a_proved_dotaz_na_upravdu_databaze()

Zapíše výsledky produkce do databáze.

public static function produkcni_funkce($mnozstvi_prace,
$mnozstvi_kapitaloveho_zbozi, $kolo)

Statická metoda obsahuje definici produkční funkce a ze zadaných parametrů vypočítá velikost produkce. Může být volána i bez vytvoření instance třídy, což je využíváno např. při vykreslování grafu produkce.

10.6.9 Spravce_informaci_pro_prehled

Úkolem této datové třídy je získávat z databáze informace, které jsou pak zobrazeny uživatelským rozhraním. Tato třída poskytuje informace o transakcích, výrobě, výsledné bilanci finanční hotovosti a získaných bodech pro minulé kolo a dále může být využita pro prognózu kapitálových toků pro aktuální kolo. V tom případě však musí být třídě dodána data, na kterých je prognóza založena.

10.6.9.1 Vlastnosti třídy

private $hrac

Hráč, kterého se týkají zjišťované informace.

private $cislo_aktualniho_kola

Číslo aktuálního kola.

10.6.9.2 Metody třídy

function __construct(hrac $hrac, $cislo_aktualniho_kola)

Hodnoty obou vlastností třídy jsou získány z parametrů konstruktoru.

public function get_data_o_nakupech()

Tato metoda zjistí, kolik zboží, práce a finančních prostředků bylo v minulém kole získáno na všech aktivních trzích. Dále jsou pak zjištěny tržní ceny na všech trzích v minulém kole. Zjištěná data jsou vrácena jako pole, kde klíčem je identifikátor trhu a hodnotou další pole. V tomto poli je pod klíčem mnozstvi uložen údaj o nakoupeném množství a pod klíčem cena tržní cena na trhu.

public function get_data_o_prodejich()

Provádí obdobné operace jako metoda get_data_o_nakupech, ale analyzuje data o prodejích na všech trzích. Zjištěné údaje jsou vráceny v poli o stejném formátu, jako u metody get_data_o_nakupech.

public function get_data_o_vyrobe()

Zjistí data o výrobě hráče v minulém kole. Data jsou navrácena v poli, kde klíčem je řetězec nazev a název vyrobeného zboží a hodnotou množství vyrobeného zboží.

public function get_data_o_kapitalu($data_pro_prognozu)

Parametr data_pro_prognozu je logická hodnota, která určuje, zda jsou data o kapitálu počítána pro přehled transakcí z minulého kola nebo pro prognózu finančních toků. V případě, že je hodnota parametru true, jedná se o zpracování dat pro prognózu a je počítáno s aktuálním kolem. V opačném případě je počítáno s minulým kolem.

Vypočítá změnu hotovosti jako rozdíl stavu hotovosti v předminulém a minulém kole (reps. v minulém a aktuálním v případě prognózy). Jedná se tedy o postup obdobný výpočtu cash-flow přímou metodou. Dále jsou zjištěny splátky úvěrů, úroků a přijaté úroky a navrácené úspory. Tyto údaje jsou zjištěny z tabulky získaných úvěrů a zapůjčených úspor. Jelikož neexistuje možnost insolvence dlužníka a věřitel má stoprocentní jistotu získání úspor i úroků, pak mohou být data o skutečně vyplacených sumách získány na základě informací o plánovaných platbách.

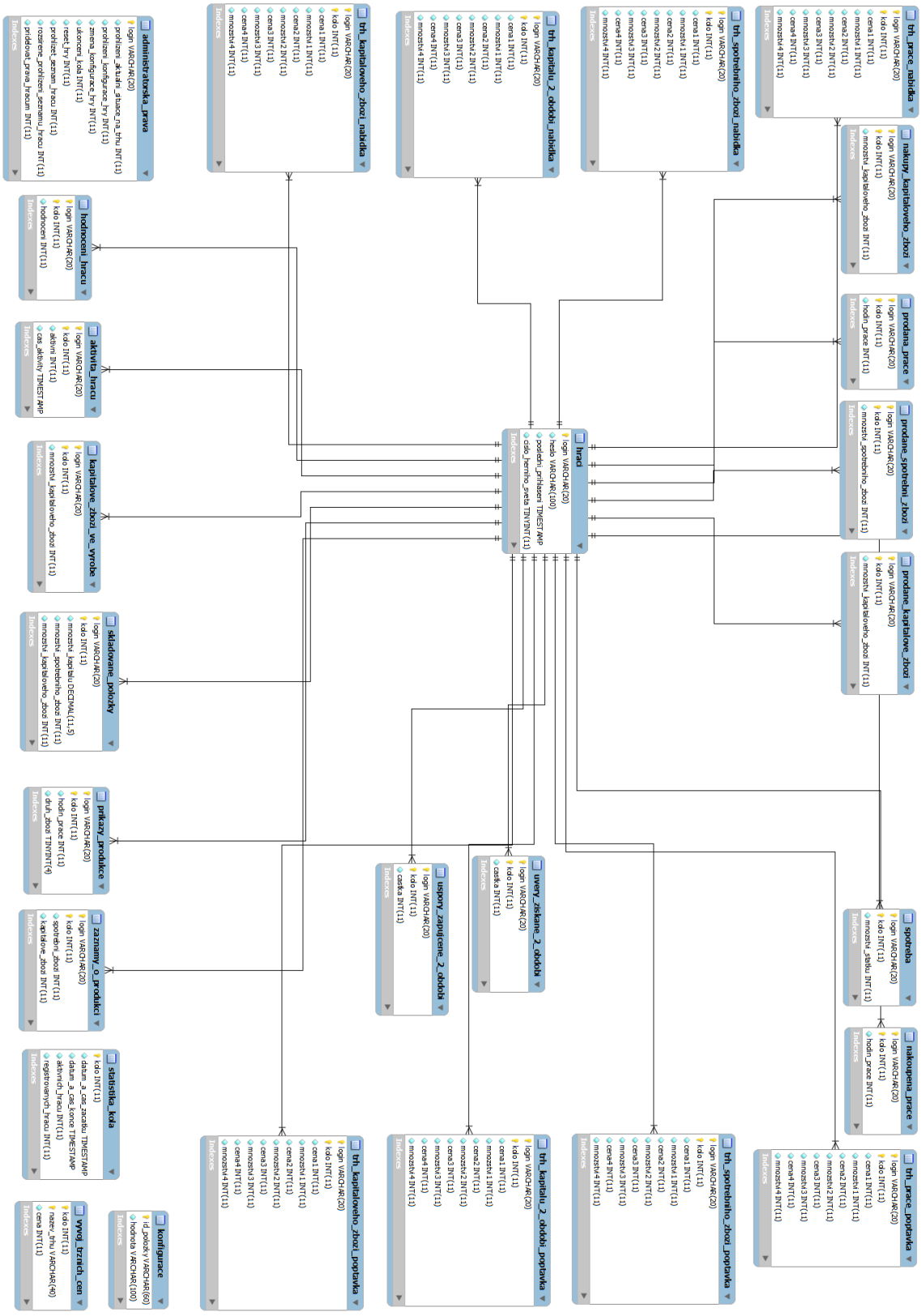
Data jsou navrácena v poli, kde klíčem je identifikátor údaje a hodnotou suma peněz v nominálním vyjádření. Úroky jsou vraceny zaokrouhleny na celá čísla dolů, aby byly údaje přehlednější pro hráče.

public function get_data_o_hodnoceni()

Zjistí počet získaných bodů v minulém kole. Data jsou vrácena v poli. Pod klíčem ziskane_body je uložen počet bodů získaných za minulé kolo.

11 Příloha B: ERA model

Model obsahuje všechny entity databáze, jejich atributy a relace mezi nimi.



12 Příloha C: Uživatelská dokumentace

Pro hraní hry stačí některý ze současnosti běžně používaných webových prohlížečů – např. Microsoft Internet Explorer, Mozilla Firefox, Opera, Google Chrome, Safari nebo Konqueror.

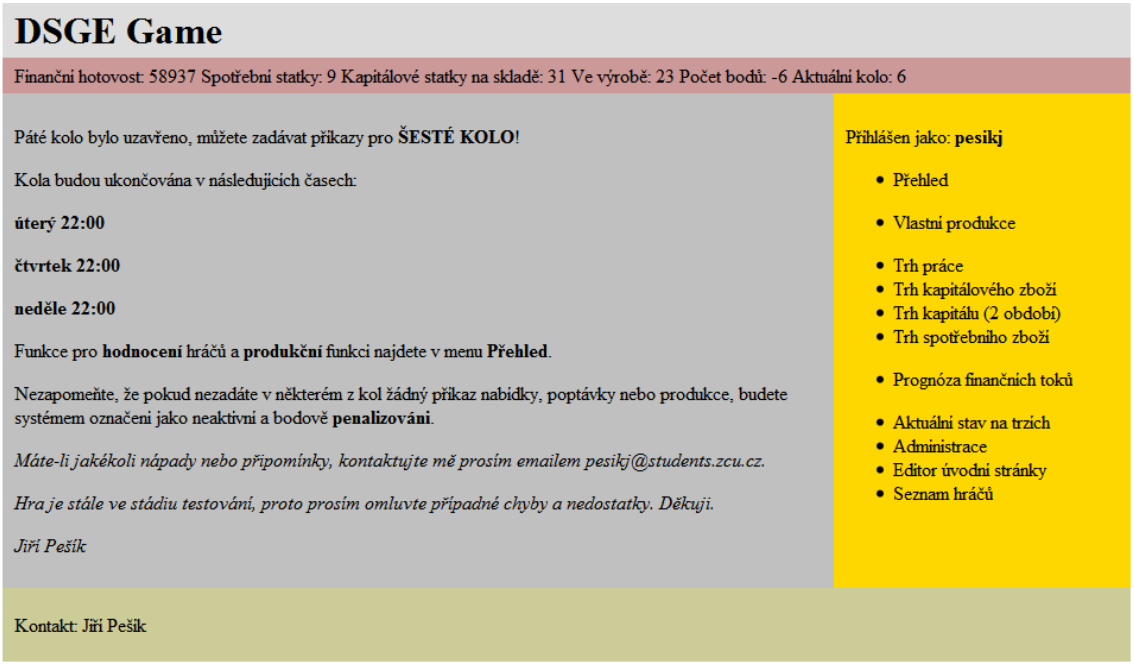
Pro přihlášení zadejte adresu serveru, na kterém je hra spuštěná, do webového prohlížeče (např. www.dsgegame.zcu.cz). V případě, že jste požádáni o své přihlašovací jméno a heslo, tyto údaje zadejte.

Tato dokumentace obsahuje přehled ovládacích prvků hry a způsob zadávání příkazů do hry. K plnému pochopení hry je pak třeba prostudovat popis ekonomického modelu hry.

12.1 Přihlášení

Po přihlášení do hry se zobrazí obrazovka s informacemi pro hráče, které sestavuje administrátor hry. Mohou se zde objevit např. informace o časech, kdy jsou uzavírána jednotlivá kola hry.

V růžovém proužku v horní části obrazovky se zobrazují informace o vašem aktuálním stavu hotovosti, skladovaném spotřebním a kapitálovém zboží, kapitálovém zboží ve výrobě a aktuálním počtu bodů. V pravé části obrazovky je menu, pomocí kterého můžete přepínat mezi jednotlivými obrazovkami hry.

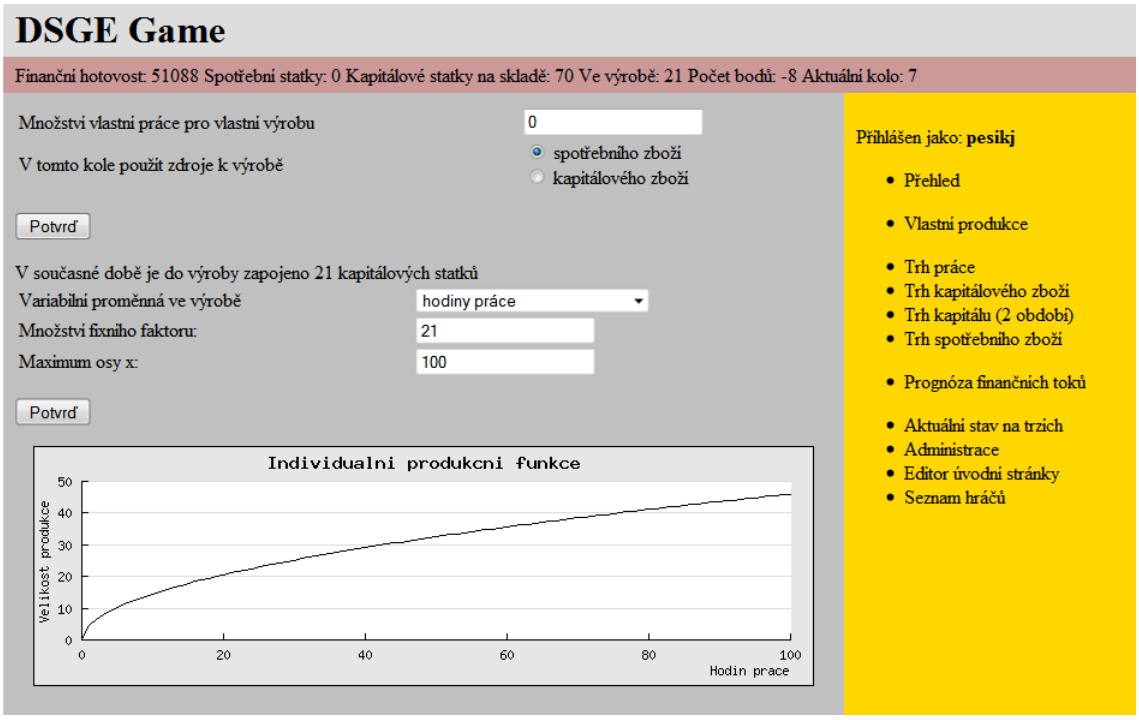


12.2 Přehled

Obrazovka poskytne informace o průběhu minulého kola. Informuje o nákupech a prodejích na jednotlivých trzích, množství vyrobeného zboží a finančních transakcích.

Součástí přehledu jsou i produkční a hodnotící funkce, podle které se počítá objem výroby a počet získaných bodů každého hráče.

12.3 Vlastní produkce



Obrazovka Vlastní produkce umožňuje zadat rozhodnutí o vlastní výrobě – množství časových jednotek, které chcete použít pro vlastní výrobu a druh vyráběného zboží. V případě, že jste zadali více časových jednotek, než je povoleno, pole formuláře se po odeslání obarví červeně. Příkaz se v takovém případě neuloží.

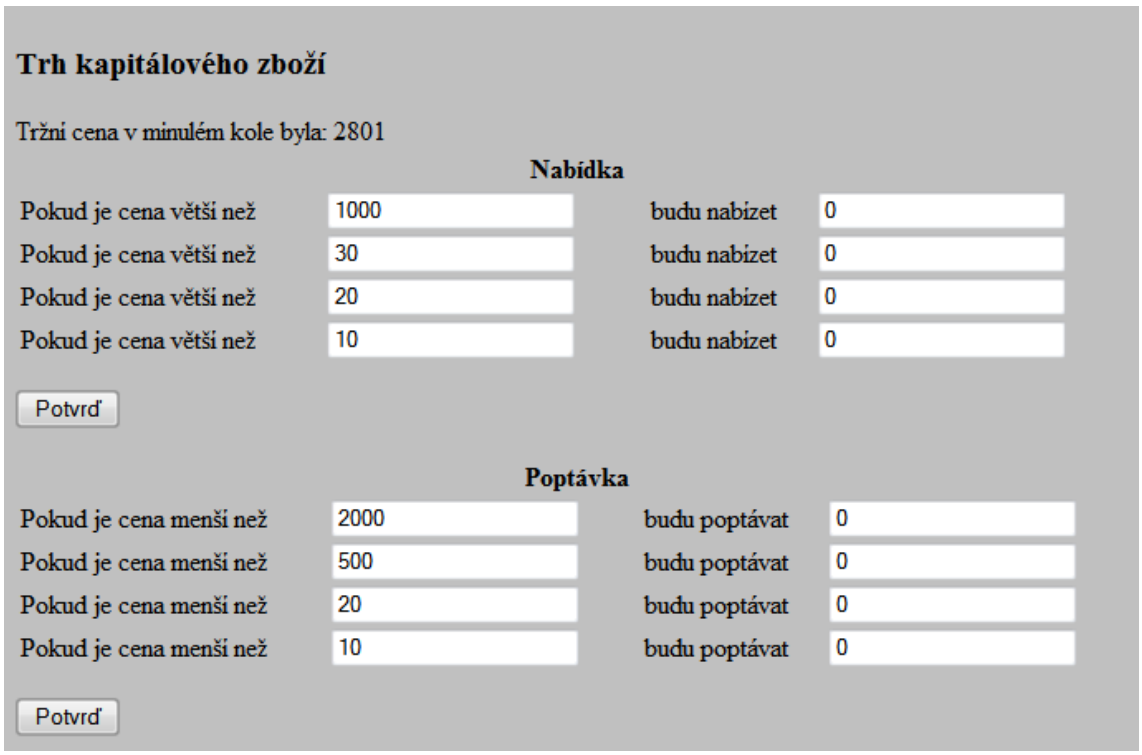
Dále je na ni zobrazen text informující o množství kapitálového zboží, které je aktuálně zapojeno do výroby. Graf v dolní části obrazovky umožňuje vymodelování průběhu produkční funkce při zafixování jednoho z výrobních faktorů (při obou faktorech variabilních by graf musel být trojrozměrný).

12.4 Trhy

Nabídky a poptávky na jednotlivých trzích můžete zadávat po kliknutí na název trhu v menu. Nabídky i poptávky se zadávají pro intervaly cen.

Při zadávání poptávky je příkaz platný pro ceny, které jsou nižší než zadaná cena. Jinými slovy specifikujete cenu, od které jste ochotni na trhu nakupovat dané množství zboží nebo práce. Pro všechny ceny vyšší než je nejvyšší zadaná cena je poptávané množství nulové.

V případě nabídky je pak příkaz platný pro ceny, které jsou vyšší než zadaná cena. Specifikujete tedy cenu, od které jste ochotni dané množství zboží nebo práce nabízet. Pro všechny ceny nižší než nejnižší zadaná cena je nabízené množství nulové.



Součástí obrazovky je pak dále graf s vývojem tržní ceny.

12.5 Prognóza finančních toků

Vzhledem k penalizaci za zápornou peněžní hotovost hra poskytuje možnost odhadu peněžních toků. Při kliknutí na odkaz v menu se dosazují tržní ceny z minulého kola, ale formulář v dolní části obrazovky umožňuje změnit ceny libovolně.

Je nabídnut výpočet peněžních toků a také počet získaných bodů při specifikovaných cenách. Výpočet bodů zahrnuje i penalizaci za zápornou hotovost a neaktivitu.

Tato obrazovka slouží pouze jako pomocná a hráči ji nemusejí používat a nemusejí se získanými výpočty řídit.

Zde si můžete určit přehled finančních toků podle zadaných cen.

Trh	Tržní cena	Příjmy	Výdaje
Trh práce	400	0	0
Trh spotřebního zboží	975	0	0
Trh kapitálového zboží	2801	0	0
Trh kapitálu na 2 období	10	0	0

Dále dojde k následujícím finančním transakcím:

- vyplacené úroky: 0
- přijané úroky: 176
- splacené úvěry: 0
- navrácené úspory: 1760

Vaše hotovost po skončení kola by při zadaných cenách byla 53024

Váš bodový zisk při daných cenách by byl 0

Trh práce	400
Trh spotřebního zboží	975
Trh kapitálového zboží	2801
Trh kapitálu na 2 období	10

Přepočítej prognózu

13 Příloha D: Popis administrátorského rozhraní

Administrátorské rozhraní umožňuje uživateli specifikovat parametry hry. Obvyklé je zadání všech voleb před zahájením prvního kola hry, volby je však možné měnit kdykoli v průběhu hry

Výchozí hodnoty

Tato část obsahuje 4 volby:

- množství spotřebního zboží na skladě
- množství kapitálového zboží na skladě
- množství kapitálového zboží ve výrobě
- finanční hotovost

Tyto volby určují, kolik z těchto položek bude mít hráč k dispozici v prvním kole hry.

Amortizační faktor

Určuje opotřebení kapitálu, neboli množství kapitálového zboží ve výrobě, které je hráčům na konci každého kola odečtena.

Rovnice produkční funkce

Produkční funkce určí, kolik zboží hráč v daném kole vyrobí. Spotřební i kapitálové zboží mají stejnou produkční funkci. Funkce se zadává textově s množství využití znaků pro sčítání (+), násobení (*), dělení (/), odečítání (-) a umocňování (^). Do rovnice se vkládají následující klíčová slova:

- mnozstvi_prace – práce, kterou hráč zapojil do výroby (vlastní i nakoupené na trhu práce)
- mnozstvi_kapitaloveho_zbozi – kapitálové zboží, které hráč v daném kole využíval ve výrobě
- kolo – číslo aktuálního kola

Rovnice hodnotící funkce

Podle hodnotící funkce je každému z hráčů na konci kola vypočítáno a přiděleno bodové hodnocení. Hodnotící funkce se zadává stejným způsobem jako produkční. Do rovnice hodnotící funkce se zadávají následující klíčová slova:

- kolo – číslo aktuálního kola
- mnozstvi_spotrebovanych_statku – množství statků, které hráč v daném kole spotřeboval
- mnozstvi_volneho_casu – rozdíl disponibilního času pro jedno kolo a počet hodin, které hráč v daném kole odpracoval (na trhu práce nebo ve vlastní výrobě)
- neaktivni – při výpočtu je dosazena 1, pokud byl hráč v daném kole neaktivní, v opačném případě je dosazena 0
- zaporny_penezni_zustatek – při výpočtu je dosazena 1, pokud měl hráč na konci kola záporný peněžní zůstatek, v případě nulového nebo kladného zůstatku je dosazena 0

Uložit změny

Uloží změny provedené v nastavení hry.

Konec kola

Tlačítkem Konec kola se ukončí aktuální herní kolo, provedou veškeré potřebné výpočty a začne nové kolo. Doba výpočtu závisí na počtu hráčů a hardwarových možnostech serveru, na kterém je hra spuštěna.