



Bakalářská práce

Modování Minecraftu

Studijní program:

Studijní obor:

Autor práce:

Vedoucí práce:

B0613A140005 – Informační technologie

Aplikovaná informatika

Jiří Růta

Ing. Jana Vitvarová

Liberec 2024

Tento list nahradte
originálem zadání.

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

19. 5. 2024

Jiří Růta

0.1 slovník

blok(block)

vykopat(mine). Hráč drží levé tlačítko myši, dokud mu blok nevypadne jako předmět. Předmět se poté přidá do jeho inventáře.

inventář (inventory) předmět (item)

1 Minecraft

1.1 blok

V Minecraftu se celý svět skládá z bloků. Hráč tyto bloky může vykopat a poté je položit, anebo z nich vyrobit vybavení.

1.2 Příklad začátku hry

Na začátku hry hráč vykope dřevo. Poté otevře UI svého inventáře. Zde jsou všechny předměty, která má hráč u sebe. Kromě toho je zde UI ve kterém hráč může vyrábět předměty. Hráč si vyrobí dřevěnou sekeru, která mu umožní kopat dřeva rychleji.

1.3 kopání

Hráč může vykopat každý blok ve hře. Vykopaný blok se mu poté přidá do inventáře. Doba kopání závisí na druhu bloku a na vybavení hráče.

1.4 inventář

Hráčův inventář obsahuje všechny předměty, které má u sebe. Hráč si může otevřít UI svého inventáře kdykoli chce. Inventář v minecraftu je rozdělen na políčka. Každé políčko může obsahovat pouze jeden druh předmětu a pouze omezené množství.

1.5 výroba

Součástí UI hráčova inventáře je i UI pro výrobu nových předmětů. Hráč v něm může vyrábět nové vybavení anebo bloky z věcí v jeho inventáři. Některé bloky mohou hráči povolit vyrobit si věci, které bez nich nemohl. Například pec umožní hráči předělat surovou rudu na ingoty.

2 Minecarft z programátorské stránky

2.1 Blok

Minecraft server musí mít v paměti minimálně 2097152 bloků pro každého hráče připojeného na serveru. Každý blok je v paměti skladován jako odkaz na jeho objekt v registru bloků a odkaz na jeho blockstate. V registru bloků jsou uchovávané všechny neměnné vlastnosti a v blockstate uchovává všechny vlastnosti které se pro blok mohou měnit.

2.1.1 registr bloků

Při startu hry se pro každý druh bloku zaregistruje objekt třídy *Block* do registru. Tento objekt je neměnný. Veškeré instance daného bloku odkazují na tento jeden objekt. Jelikož je tento objekt společný pro všechny instance, pokud chceme ukládat data specifická pro jeden block musíme použít blockstate anebo tile entity.

2.1.2 blockstate

Blockstate je neměnná třída obsahující pouze proměnné. Pro každou kombinaci proměnných je vygenerován singleton. Každý blok odkazuje na jeden ze singletonů. Blockstate šetří paměť, tím že dovoluje několika blokům držet pouze jednu proměnnou (odkaz na blockstate) místo několika.

Například máme blok lampy. Každý blok lampy odkazuje na stejný objekt v registru bloků, ale pokud chceme určit zdali je lampa zapnutá musíme se na který blockstate odkazuje.

2.1.3 tile entities

Pokud v bloku potřebujeme skladovat velké množství proměnných, můžeme využít *tile entity*. Objekt třídy *tile entity* je samostatný pro každý blok.

Například blok bedna umožňuje hráči odložit si předměty. Které předměty jsou uloženy v bedně je uloženo v *tile entity*. Bedna také odkazuje na blockstate, který udává kterým směrem je otočena. A každá bedna také odkazuje na svůj objekt v registru bloků.

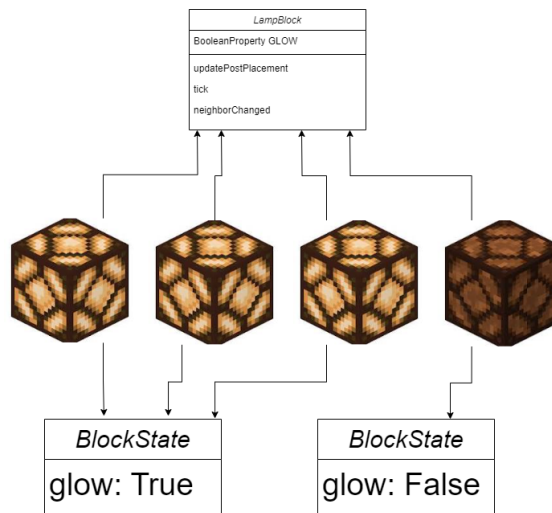


Figure 2.1: blockstate

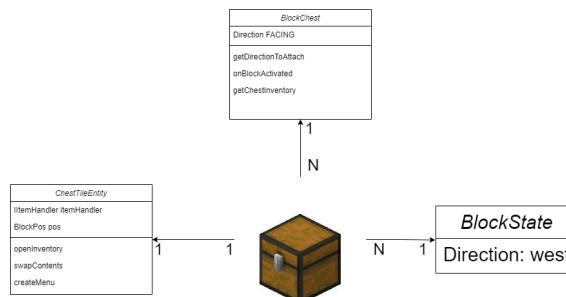


Figure 2.2: příklad vztahů

2.2 Inventář

Inventář v minecraftu je rozdělen na políčka. Každé políčko může obsahovat pouze jeden druh předmětu a pouze omezené množství. V programu je každé políčko reprezentováno objektem *itemstack*.

2.2.1 Itemstack

Třída obsahuje 2 důležité proměnné, jaký předmět obsahuje (odkaz na jeho třídu) a jeho množství. Třída obsahuje převážně metody na porovnávání objektů Itemstack mezi sebou, předávání informací o předmětu, ukládání a načítání dat.

2.2.2 Item

Každý druh předmětu má svůj singleton, který dědí třídu *Item*. Tato třída obsahuje proměnné a metody pro daný předmět. Například obsahuje metodu "onUse", která je volána pokud se hráč pokusí použít předmět. Jelikož je tato třída společná pro všechny itemy pokud chceme ukládat data specifická pro jeden předmět musíme

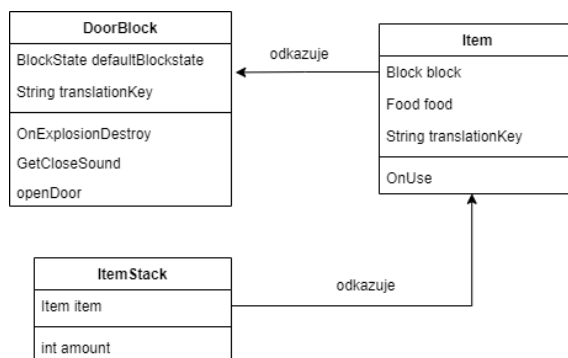


Figure 2.3: ItemStack vztahy

použít NBT.

2.3 Výroba

Velká část minecraftu je vytváření nových předmětů z těch které už hráč vlastní. Z čeho se předměty vyrábějí je zapsáno v JSON souborech.

2.3.1 Recept

Ve hře je několik druhů receptů, každý druh receptu má svojí třídu implementující interface IRecipe. Každý JSON soubor ve složce s recepty má hodnotu "type", která říká kterou třídou ho má hra přečíst. Při zapnutí se načtou veškeré recepty z JSON souborů a pro každý json soubor se vytvoří jeden objekt třídy implementující IRecipe. Tyto objekty obsahují veškerá data z JSON souborů a implementují metody na porovnání zdali je možné recept vyrobit. Práce těchto tříd není vyrobit daný předmět, ale pouze říct zdali je možné daný předmět vyrobit a předat vyžádané informace.

2.4 Klient a Server

Minecraft kód vždy běží dvakrát, jednou jako server a jednou jako klient. I když hráč hraje sám na vlastním počítači, minecraft pořád běží rozdělen na stranu serveru a klienta. Na obou stranách běží stejný kód. Na rozlišení se používá metoda "world.isRemote", která vrací *true* pokud kód běží na klientu. Některé operace musí běžet pouze na jedné straně a druhou stranu upozornit pomocí paketů. Například vytváření nových předmětů by se musí provádět na serveru, jinak je možné že server nebude o předmětu vědět a když se ho hráč pokusí použít tak nebude fungovat, protože o něm server neví. Na druhou stranu, vykreslování grafiky se musí provádět pouze na klientu.

2.5 UI

Některé bloky ,předměty otevřou hráči UI. Například když hráč klikne na pec, otevře se mu UI ve kterém může odložit předměty do pece, aby je pec předělala na jiné. Ui se v Minecarftu skládá ze dvou částí: Container a Screen.

2.5.1 Container

Container obstarává logicku a data. Komunikuje s ostatními částmi hry. Běží na serveru i na klientu.

2.5.2 Screen

Určuje jak bude UI vypadat. Zobrazuje text a obrázky. Komunikuje pouze s Container. Může obsahovat logicku, ale měla by být jen okolo grafiky v UI. Běží pouze na klientu.

3 Vytváření modu

Minecraft v základu nepovolujeme módování, pokud chceme musíme vytvořit mód musíme použít zavaděč módů, který se pustí při startu hry a načte kód módu do hry za pomoci *Java Class Loader*. Zavaděče módu také poskytují vývojářům módů eventy, které jsou volány během hry. Existují 2 zavaděče módu *Forge* a *Fabric*. Tyto zavaděče nemohou být spuštěny společně. *Forge* se zaměřuje na kompatibilitu mezi módy. *Forge* má větší množství eventů a jsou k němu zabalené abstraktní třídy a rozhraní pro většinu věcí, které vývojáři módu často implementují. Díky tomu mají vývojáři normalizované rozhraní, která mohou používat ke komunikaci s ostatními módy. *Forge* je také starší, takže většina módů je dělaná pro *forge*. *Fabric* se zaměřuje na výkon. *Fabric* má menší API a na rozdíl od *Forge* načítá pouze kód který je potřeba. *Fabric* také dovoluje upravovat soubory základní hry. Z pohledu hráče jsou zavaděče pouze omezení, které omezuje jaké módy hráč může využívat společně.

Pro svůj mod jsem si vybral *Forge*, jelikož se v něm jednodušeji řeší kompatibilita mezi ostatními módy a více módu je dělaných pro *Forge*.

Minecraft vyšel v roce 2009, od té doby vyšlo 20 velkých aktualizací. Minecraft dovoluje hráčům hrát jakoukoli verzi. Jelikož se s každou verzí minecraftu mění kód základní hry, musí pro každou verzi udělat nová verze zavaděče módu. Módy napsané pro jednu verzi minecraftu nejsou kompatibilní s ostatními verzemi.

Všechny módy dodržují hierarchii souborů. Ve složce *java* se nachází veškerý *java* kód. Ve složce *resources/assets* se nachází textury a *json* soubory modelů. S těmito soubory by se mělo pracovat pouze na straně klienta. *resources/data* obsahuje *json* soubory receptů.

Named binary tag je jednoduchý hierarchický formát pro ukládání dat.

Zavaděč umožňuje přidat nové objekty do příslušných registrů, kde s nimi může komunikovat základní hra i veškeré módy.

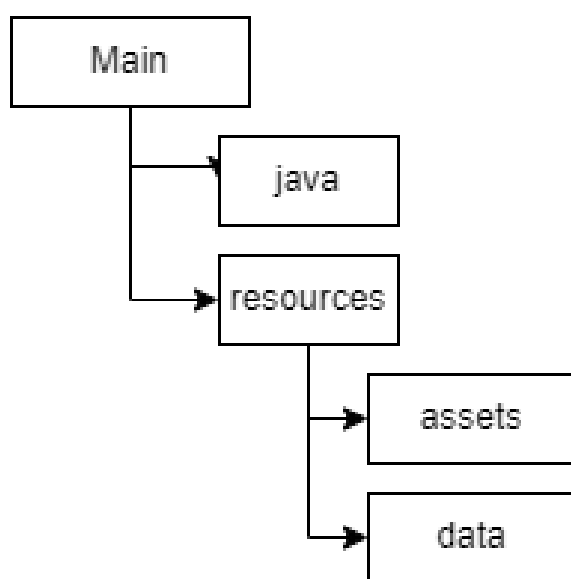


Figure 3.1: hierarchie souborů

4 Můj mód z pohledu hráče

Módy často přidávají bloky které představují různé stroje, jako třeba elektrická pec anebo Hydraulický lis. Tyto bloky většinou mají vlastní inventář do kterého dá hráč předměty a stroj z nich vyrobí jiné. Tyto stroje většinou potřebují elektřinu, aby fungovali, což je věc která se v základní hře nevyskytuje a je exkluzivní pouze pro módy. Těmto módům se říká technické módy. Dobrý technický mód funguje s ostatními technickými módy. Například pokud mód přidává generátor tak mělo být možno z generátoru odvádět elektřinu i kabely z cizího módu a napájet s ní stroje z cizích módů. Můj mód přidává několik strojů. Přidává síta, lis a kotlík. Síto přesívá tekutiny na rudy. Lis umí předměty na tekutiny. V kotlíku může hráč kombinovat tekutiny a předměty, aby získal nové tekutiny. Všechny fungují s trubkami a kabely z populárních technických módů. Můj mód je inspirován módem *Ex Nihilo*. Jedná se o populární mód vydaný v roce 2014. Tento mód přidává síta ve která přesívají hlínu na rudy a sud ve kterém může hráč kombinovat tekutiny a předměty, aby získal nové tekutiny. Tento mód je neudržovaný a funguje pouze pro velmi starou verzi minecraftu, ale má několik neoficiálních pokračování. *Ex Nihilo 2* je jediné oficiální pokračování k módu *Ex Nihilo* od jeho autora. Bylo vydáno roku 2015, ale od roku 2016 je opuštěné. Obsahově je podobné původnímu *Ex Nihilo*, ale některé věci chybí, mód zůstal nedodělán. *Ex Nihilo Omnia* je neoficiální pokračování vydané 4.9. 2016. Tento mód přidává vše co původní *Ex Nihilo* pouze s malými úpravami a obsahem navíc. Tento mód dostal svoji poslední aktualizaci v roce 2019. *Ex Nihilo Adscensio* je neoficiální pokračování vydané 29.11 2016. Tento mód je první který přidal více úrovní sít. Poslední aktualizaci dostal 6.5 2017. *Ex Nihilo Creatio* je fork *Ex Nihilo Adscensio*. Je to první mód který přidává sívky, které hráč musí napájet elektřinou, ale za odměnu vyrábějí bez hráčovi přítomnosti. Mód byl vydán 14.8. 2017 a poslední aktualizace byla vydána 27.8 2019. *Ex Nihilo: Sequentia* začalo jako *Ex Nihilo Creatio* předělané na vyšší minecraft verze. Tento mód obsahuje veškerý obsah ze všech předchozích *Ex Nihilo* inspirovaných módů. Byl vydán 4.8. 2020 a je stále podporován. Má několik módů které dodávají další obsah k tomuto módu anebo přidávají kompatibilitu s ostatními módy. *Fabricae Ex Nihilo* neoficiální pokračování módu *Ex Nihilo* pro zavaděč Fabric. Vydáno 21.3. 2022 a stále aktualizováno.

jméno módu	podporované minecraft verze	zavaděč	počet stažení
Ex Nihilo	1.6 a 1.7	Forge	8 058 815
Ex Nihilo 2	1.8	Forge	61 152
Ex Nihilo Omnia	1.10 a 1.12	Forge	2 611 291
Ex Nihilo Adscensio	1.10	Forge	6 236 591
Ex Nihilo: Creatio	1.12	Forge	15 638 553
Ex Nihilo: Sequentia	1.15 až 1.20 (nejnovější)	Forge	6 761 281
Fabricae Ex Nihilo	1.18 až 1.20 (nejnovější)	Fabric	168 881
Ex Aqua	1.16	Forge	432

5 Můj mód z pohledu programátora

Začal jsem tím že jsem do hry přidal blok pro sívku. Nejprve jsem vytvořil třídu pro blok která dědí ze třídy *Block*. Poté jsem registroval objekt s touto třídou do registru bloků. Registrace vyžaduje jméno pro tento objekt a konstruktor třídy *Block*. Podle jména hra pak vyhledává stejnojmenné soubory v podsložkách *resources*. Pro blok je potřeba do přidat soubor do složek : *blockstates*, *models*, *textures*. Do složky *blockstates* přidám json soubor který obsahuje, který model má být použit v závislosti na proměnných v *blockstate*. Do složky *models* přidám json soubor, který obsahuje model bloku a odkaz na jeho textury. Soubor také může dědit z ostatních souborů, pomocí parametru *parent*. Pro model krychle je zde předpřipravený soubor *cube_all.Vesložcetexturessenacházejívšechnytextury*.

*Pro upravení modelu bloku je potřeba přepsat metodu `getShape` ve třídě bloku ,aby vracela model sestavený z kvádrů a také upravit soubor ve složce *models*. K vytvoření modelů pro moje blocky jsem použil program *Blockbench*.*

Dále jsem k bloku přidal *tile entity*. Nejprve jsem vytvořil třídu dědící od třídy *TileEntity*. V této třídě jsem přepsal metody k ukládání a načítání dat z *nbt*. Poté jsem *tile entity* registroval. Při registraci se předá registru constructor pro *tileentity* a odkaz na všechny registrované blocky kterým *tileentity* patří. Poté přepíšu ve třídách těch bloků metodu na vrácení *tile entity* a propojení bloku a *tile entity* je kompletní. Chtěl jsem v *tile entity* skladovat tekutiny tak jsem si vytvořil novou třídu *MyLiquidTank* která dědila ze třídy *FluidTank*. Implementoval jsem všechny její metody a přidal pár vlastních abych zabránil duplikacy kódu. Objekt této třídy jsem poté přidal do své *tile entity*. Pro práci s ním jsem vytvořil interface *IMyLiquidTankTile* . Jelikož potřebuju synchronizovat *MyLiquidTank* mezi server a klientem vytvořil jsem si třídu *MyFluidStackPacket* která je poslána ze serveru klientovy a obsahuje momentální stav daného *MyLiquidTank*. Ve třídě *MyLiquidTank* jsem implementoval odeslání a přijmutí tohoto packetu. Také jsem chtěl aby ostatní módy mohli komunikovat s *MyLiquidTank* tak jsem vytvořil třídu *WaterFluidTankCapabilityAdapter* co slouží jako adaptér. Která překládá můj *MyLiquidTank* na *IFluidTank* což je interface zabudovaný v *Forge*. Testoval jsem tento kód s nejznámějšími módy a se všemi fungoval. Poté jsem to samé udělal pro *MyEnergyStorage*. Vytvořil jsem *IMYEnergyStorageTile* pro *tile entity* obsahující *MyEnergyStorage*. Vytvořil jsem *MyEnergyPacket* pro synchronizaci klienta a serveru. Vytvořil jsem *EnergyStorageAdapter* který překládá *MyEnergyStorage* na *IEnergyStorage* z *Forge* pro komunikaci s ostatními módy. Poté jsem měl *tile entity* která je schopná skladovat tekutiny, elektrinu a předměty Dále jsem chtěl aby v *tile entity* bylo možné vyrábět předměty, k tomu jsem potřeboval recept který bude uvádět co bude potřeba jako

ustup do stroje a co vyjde jako výstup. Vytvořil jsem třídu SieveRecipe implementující interface ISieveRecipe. Tato třída přečte data z json souboru a poté vytvoří jednu svoji instanci pro každý úspěšně načtený soubor. Jedna instance této třídy představuje jeden recept. Tato třída obsahuje metody které určí zdali jsou všechny potřebné ingredience a vrátí předmět, který má být vyroben. Ke každému druhu receptu jsem musel napsat třídu která překládá daný recept pro její mód a přidat mu UI. Také jsem pro tile entity přidal container a screen. container bere informace přímo z tile entity a předává je pro screen. container také obsahuje logicku.