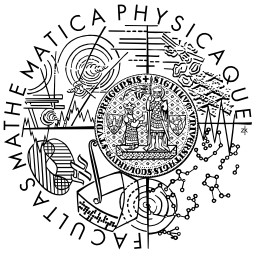


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Pavel Halbich

Tau Ceti f 2 – budovatelská počítačová hra se strategickými prvky

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Pavel Ježek, Ph.D.

Studijní program: Informatika

Studijní obor: Programování a softwarové systémy

Praha 2017

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Děkuji mému vedoucímu Pavlu Ježkovi za pomoc s touto prací, mým rodičům za podporu a pevné nervy, mé přítelkyni Veronice taktéž za podporu a pomoc s 2D grafikou a Jiřímu Kurčíkovi za laskavé poskytnutí práv na použití jeho hudební tvorby v mé hře.

Název práce: Tau Ceti f 2 – budovatelská počítačová hra se strategickými prvky

Autor: Pavel Halbich

Katedra: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Pavel Ježek, Ph.D., Katedra distribuovaných a spolehlivých systémů

Abstrakt: Abstrakt.

Klíčová slova: klíčová slova

Title: Tau Ceti f 2 – A Creative Computer Game with Strategic Elements

Author: Pavel Halbich

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Pavel Ježek, Ph.D., Department of Distributed and Dependable Systems

Abstract: Abstract.

Keywords: key words

Obsah

1. Úvod

V době vzniku této práce jsou velice populární hry s otevřeným světem. Lákají hráče na obsáhlost světa a možnost nelineárního řešení problémů a herních úkolů. Her s otevřeným světem najdeme nepřeberné množství v různých herních žánrech. My se zaměříme na podmnožinu her, které kromě otevřeného světa nabízí také možnosti budování struktur a vyžadují od hráče netriviální styl hraní, který mu umožňuje ve hře přežít. Průmyslově se tyto hry často označují jako *Sanboxové*, *S budováním*, *S průzkumem prostředí*, *Survival*. Autor této práce má tento typ her v oblibě a rád by touto prací představil svoji vizi dalšího možného rozvoje her tohoto žánru. Cílem práce by měla být implementace nového herního principu stavění, které současné herní tituly nenabízí.

1.1 Charakteristika her

V práci se budeme zabývat několika různými hrami, které však mají několik společných vlastností. Jedním ze základních konceptů je využívání herních bloků. Dalším význačným prvkem je způsob integrace herních bloků do herního prostření. Některé hry jsou celé tvořeny bloky, jiné se snaží dosáhnout vyššího stupně realismu ve hře a bloky využívají pouze pro konstrukci různých herních objektů. Ústředním tématem této práce tedy bude rozbor systému bloků a práce s nimi a popis hráčských problémů způsobených danými koncepty. V další části práce pak navrhneme a implementujeme vlastní řešení.

1.1.1 Hry kompletně blokové

Začněme hrami, které využívají bloků jako základního elementu celé hry. Bloky zde tvoří doslova celý svět. Mezi nejpopulárnější a širokou veřejností nejznámější bychom měli zařadit hru *Minecraft*. Jak je zřejmé z obrázku 1.1, celá hra včetně herní postavy, nehratelných postav - Non-playable character (NPC), nebo třeba mraků, je stylizovaná do bloků ve tvaru kostky. Většina bloků je stejně velká a má hranu o délce 1 metru [1], [2].

Obrázek 1.1: Hra Minecraft - hrad na skále

Mezi dalšími hrami bychom mohli zmínit například *Terraria*. Ta je o něco mladší než *Minecraft*, ale je častým zdrojem diskusí, zda je lepší new *Minecraft*, nebo ne. Pravdou je, že obě hry mají svůj svět kompletně složený z kostek (*Terraria* je však 2D hra), ale každá si klade trochu jiné cíle. *Terraria* je více orientovaná na příběh, obsahuje více NPC i bossů, *Minecraft* je orientován spíše na stavění. (Porovnání [3])

1.1.2 Hry s prvky realismu

Mezi tyto hry bychom mohli zařadit třeba hry *Space Engineers* či *Medieval Engineers*, využívají kombinaci herních bloků s voxelovou reprezentací světa. a tím dosahují vyššího stupně realismu ve hře. Skvělým příkladem je obrázek 1.2 převzatý ze serveru Gamespot [4].

Je vidět, že asteroid ani zdaleka není kostičkovaný. Přesto je povrch ve hře editovatelný voxelovými nástroji. Samotná základna i vesmírná plavidla jsou na první pohled tvořeny bloky. *Space Engineers* umožňuje stavět pohyblivé stroje, které si hráč postaví z herních bloků, ale po ukončení editace se nadále chovají jako jedna entita. Stále je na ně však aplikována fyzika, takže je možné plavidlo poškodit, nebo dokonce zničit. Tento stupeň realismu od naší hry vyžadovat nebudeme.

Obrázek 1.2: Hra Space Engineers - základna na asteroidu

1.1.3 Hry s maximálním důrazem na simulaci reality

Do této sekce bychom měli zařadit například vesmírný simulátor *Take on Mars*. // TODO popis, obrázek

bloky na stavění, ale třeba vozidla kompletní

1.1.4 Ostatní - zařadit TODO

Můžeme však nalézt i další příklady her (// TODO *Take on Mars*, *Novus Inceptio*, *Planet Nomads*, *ARK Survival Evolved*, *No man's sky*).

1.2 Herní bloky

Obvykle je ve hře definován jeden základní rozměr bloku, který je neměnný. (*Space Engineers* definuje více velikostí – ty však nelze vzájemně kombinovat). To však může být problémem, pokud se hráč rozhodne postavit v herním světě nějakou větší a komplexnější strukturu podle reálné či fiktivní předlohy. Pro příklad uveďme některé výtvory ze hry *Minecraft*:

- King's landing z knih Píseň ledu a ohně
- Minas Tirith - hlavní město Gondoru z univerza J.R.R. Tolkiena

Autoři těchto výtvorů museli volit takové měřítko, aby byly výtvory dostatečně detailní, ale zároveň aby bylo možné výtvor postavit v nějakém rozumném čase. Obecně ale můžeme říct, že čím větších detailů chtěli autoři dosáhnout, tím větší musel celý výtvor být, ale za ceny toho, že velké plochy trvaly o to déle.

1.2.1 Náš návrh úpravy

Chtěli bychom dát hráči k dispozici možnost ovlivňovat velikost bloku. Tím by mohl rychleji stavět rozsáhlejší struktury a přitom se věnovat i drobným či estetickým detailům.

1.3 Inventář

Dalším společným prvkem tohoto druhu her je inventář bloků, které může hráč umístit do herního světa. Hráč přes celé herní okno vidí Head-Up Display (HUD)[5], ve kterém má zobrazenou kromě jiného nabídku bloků, které má na rychlé volbě, může je snadno zvolit a daný blok umístit do herního světa. Navíc hry mohou definovat skupiny bloků (*Space Engineers*, *Medieval Engineers*), mezi kterými hráč může přepínat a tím rychle kompletně změnit sadu rychlé nabídky. Vidíme však limitaci v tom, že hráč musí ručně spravovat tyto seznamy a jednotlivé bloky (či nástroje) umisťovat do příslušných pozic.

Rádi bychom navrhli jiný způsob správy těchto sad, aby hráč jednou definoval, jaké prvky chce mít v příslušných sadách. Při vytvoření nového bloku by pak nemusel ručně editovat sadu, ale automaticky by měl tento nový blok k dispozici. // TODO aplikovat do hry - typy bloků jako tagy

1.4 Herní nepřítel

Protože samotné stavění bez nějakého cíle či překážky není úplně zábavné, musíme hráči připravit nějakou překážku, komplikaci, kterou musí překonávat. Zde neexistuje jednoznačné řešení — to je závislé na celkovém prostředí hry, zamýšlené cílově skupině a mnoha dalších faktorech. Cílem našeho hráče bude přežít kyselé deště. Ty budou přicházet v náhodných intervalech a budou sloužit jako překážka v rozvoji hry. Zároveň to ale bude pro hráče nástroj, jak získávat prostředky pro ochranu před dalšími dešti a rozvoj svých staveb.

1.5 Cíle práce

Tato práce by měla naplnit následující cíle:

- Bloky
- Inventář
- TODO

2. Analýza zadání

V této části provedeme rozbor toho, jak různé hry v současné době přistupují k řešení jednotlivých součástí hry. Tím si připravíme prostor pro specifikaci toho, jak by naše hra mohla vypadat a co všechno by měla umět.

// TODO remove me: Úkol zněl jasně: Cílem bakalářské práce je implementace budovatelské hry se strategickými prvky, hranou z pohledu třetí osoby. Hra se odehrává na nehostinné planetě, kde je hráčův úhlavní nepřítel nedostatek zdrojů a superkyselé deště. Hráč začíná v menší budově – zbytek přistávacího modulu kosmické lodi. Dochází mu elektrická energie i kyslík a je na hráči, aby takticky využíval dostupné zdroje, hledal nové možnosti výroby energie a přežil kyselé bouře. Cílem práce není vytvořit dohratelnou hru, spíše proof-of-concept, zda je tento typ hry s uvedenými mechanikami zábavný a má smysl v jejím vývoji pokračovat i nadále.

2.1 Stávající implementace mechanismů

V následujících podkapitolách si rozebereme jednotlivé části her a jak je implementují ostatní.

2.1.1 Bloky

různé druhy, velikosti, jejich vizuální reprezentace, rozšiřovatelnost, obecně co všechno by měly umět

Základní vlastnosti

rozměry

Součásti bloků

komponenty elektřiny, inventáře

Speciality

Multiblocks, náhled inventáře (*Medieval Engineers*)

2.1.2 Herní svět

jaký je herní svě

Reprezentace

MC - bloky, chuncks

Bloky v herním světě

do gridu, start-free grid

Denní / noční cyklus

obvykle tam je

Herní překážky

počasí, NPC, atributy avataru

(Ne)fyzikální chování

MC - bloky stojí ve vzduchu, ale třeba písek při updatu začne padat

2.1.3 Inventář

mc pevné sloty

2.1.4 Avatar hráče

avatar má nějaké vlastnosti, HUD, 1St / 3rd person view, zdraví, stamina, hlad, O2

2.2 Co bychom chtěli implementovat

V následujících podkapitolách si rozebereme naše požadavky na hru

2.2.1 Bloky

různé druhy, velikosti, jejich vizuální reprezentace, rozšiřovatelnost, obecně co všechno by měly umět

Standardní bloky

- ty různé krychle

Kvádr

- všechny velikosti
- má elektriku

Kvádr seříznutý stranou

- všechny velikosti
- má elektriku

Kvádr říznutý tělem

-všechny velikosti
- má elektriku

Kvádr – základový

- velikost v ose Z omezena na 4 základní bloky
- má elektriku

Kvádr seříznutý stranou – základový

- velikost v ose Z omezena na 4 základní bloky
- má elektriku

Kvádr – polykarbonát

- všechny velikosti
- má elektriku

Speciální bloky

- popsat speciality

Napájené okno

- minimální velikost 2 x 1 x 2, maximální velikost 20 x 1 x 20 základních bloků
- má elektriku

Terminál

- speciální, pevná velikost 1 x 8 x 5 bloků
- má elektriku

Dveře

- speciální, pevná velikost 7 x 7 x 11 bloků
- má elektriku

Kyslíková nádoba

- speciální, pevná velikost 2 x 2 x 2 bloků
- má kyslíkovou komponentu

Plnička kyslíkové nádoby

- speciální, pevná velikost 4 x 3 x 4 bloků
- má elektriku, kyslíkovou komponentu

Baterie

- speciální, pevná velikost 3 x 3 x 3 bloků
- má elektriku

Generátor bloků

- omezená velikost v ose Z na 2 bloky, jinak 3 x 3 až 20 x 20 v ostatních osách
- má elektriku

Generátor energie

- omezená velikost v ose Z na 2 bloky, jinak 3 x 3 až 20 x 20 v ostatních osách
- má elektriku

Světlo

-velikost omezena na 1 x 1 x 1 blok
- má elektriku

Spínač

-velikost omezena na 1 x 1 x 1 blok
- má elektriku

Základní vlastnosti

rozměry

Součásti bloků

komponenty elektřiny, inventáře
- dále také propojovatelnost

2.2.2 Herní svět

jaký chceme herní svět

Reprezentace

bude nám stačit nějaký tree, definovat rozměry, na chuncky kašlem

Bloky v herním světě

do gridu

Denní / noční cyklus

dáme ho

Herní překážky

počasí, , atributy avataru

(Ne)fyzikální chování

nebudeme hrotit

2.2.3 Inventář

chceme volné sloty, rozšiřitelnost

2.2.4 Avatar hráče

avatar má nějaké vlastnosti, HUD, 1St / 3rd person view, zdraví, O2, energie

2.3 Backlog

???

3. Detailní analýza

V této kapitole podrobně rozebereme cíle práce.

3.1 Herní engine

Nyní už víme, čeho bychom chtěli dosáhnout a je na čase vyřešit, jak toho dosáhneme. V první řadě bychom se měli zamyslet nad tím, jaký herní engine použijeme. Díky tomu budeme moct počítat s možnostmi a omezeními danými touto volbou. V zásadě máme několik možností:

- Implementace vlastního enginu
- Použití existující nízkoúrovňový framework
- Použití existující herní engine

Každá volba má své pro a proti, což podrobně rozebereme v následujících odstavcích.

3.1.1 Vlastní engine

Hlavní výhodou, ale zároveň nevýhodou této volby je to, že bychom si všechny potřebné součásti enginu (třeba renderování) museli napsat sami. Tím bychom měli naprostou kontrolu nad celým produktem, ale zabralo by nám to netriviální množství času. Vzhledem k rozsahu plánované funkcionality by tato volba byla nepraktická a tedy touto cestou se nevydáme.

3.1.2 Nízkoúrovňový framework

Máme na výběr z více druhů frameworků postavených na různých platformách. Mezi známějšími bychom mohli uvést například XNA (`C#`) nebo Ogre (`C++`). Oba frameworky jsou k dispozici zdarma, nicméně jejich podpora stagnuje. Implementace hry s použitím některého z těchto frameworků by byla rychlejší než v předchozím případě, ale stále bychom museli spoustu funkcionality implementovat sami.

3.1.3 Existující herní engine

V této kategorii máme nejvíce možností jak rychle implementovat celou práci. Zástupců je opět mnoho, nicméně mezi nejoblíbenější se řadí Unity (`C#`) a Unreal Engine (UE)(`C++`), které jsou oba zdarma. Díky práci komunity pro oba enginy existuje i kvalitní dokumentace. Navíc jsou enginy obvykle multiplatformní a tedy existuje zde snadný postup distribuce na různé typy herních zařízení. Při rešerši jsme zjistili následující klady a zápory:

Unity

Výhoda (ale i nevýhoda) Unity je celkově v jednoduchosti. Engine nabízí dostatečné možnosti i pro tvorbu AAA herních titulů, ale za cenu toho, že si toho autoři musejí dost napsat sami (oproti UE).

- Klady

  – Pokud bychom chtěli modifikovatelný či nějakým způsobem dynamický terén, Unity implementuje podporu modifikace terénu.

- Zápory

  – Při rešerši jsme řešili podporu dynamického navigačního meshe, která v Unity nebyla příliš dobrá. Při změně prostředí docházelo k lagům během přepočtu navmeshe. To by byl při umisťování bloků zásadní problém.

  – Další nevýhodou je podpora materiálů, kdy bychom snadným způsobem nedosáhli vizuálně přitažlivého prostředí.

Unreal Engine

Oproti Unity je UE podstatně komplexnější a pochopení všech vztahů a závislostí může být pro začínajícího herního programátora obtížné. Dalším zdrojem problémů může být i programování v `C++`, které je navíc díky technologii Unreal Build Tool (UBT) trochu jiné než klasické `C++` a je potřeba dodržovat standardy definované UE.

- Klady

  – Komplexnější engine (UE je primárně určen pro vývoj AAA titulů)

  – Oproti Unity lepší podpora materiálů — i negrafik může snadno vytvořit vizuálně přitažlivé povrchy objektů a nemusí se přitom zabývat psaním vlastních shaderů

  – Rychlý dynamický navmesh

- Zápory

  – Komplexnější engine

Nakonec jsme zvolili poslední možnost - Unreal Engine. Autorovy znalosti především z oblasti `C#` sice hovořily pro použití Unity, nicméně výhody použití UE převážily nad nevýhodami i všemi výhodami Unity. // TODO tohle chce vyladit

3.2 Bloky

Zde by měl být popis možností jak definovat a následně implementovat bloky. jaké jsou výhody a nevýhody jednotlivých implementací

- externě editovatelné formáty (+ - modding, - těžší implementace, parsing, validace) - binární formát

- xml

- interní formát - specifické subclassy pro bloky včetně specifických vlastností přímo na - definiční struktura

3.3 Vlastnosti bloků

Popis toho, co blok umí

3.3.1 Energie

- popis energie, co to umí (např. výkon)

3.3.2 Energetická síť

-způsob zapojení do sítě

3.3.3 Kyslík

- to je podobný jako energie
- mít možnost uchování kyslíku, v případě použitíí elektirky pak i generování

3.3.4 Označovatelnost

- hráč může avatarem zamířit na blok a ten se označí červeně, žlutě zeleně

3.3.5 Možnost vzít do inventáře

- bloky mohou být sebratelné, tedy hráč si je může dát do svého inventáře. vlastnosti jako třeba uchovaná hodnota kyslíku, pak zůstávají zachované

3.3.6 Interakce

- vypínač, světla - vlastní UI
- bloky mohou být použitelné, tj. hráč s nimi může nějakým zůsobem interagovat

3.3.7 Zapojení do rozpoznávání tvarů

- generátor bloků
jaké byly možnosti - abecné rozpoznávání (původní implementace, rozvést nutnost rozpadu tvarů na menší (slope) + doplnění kvádry

3.4 Komponenty bloků

popis jednotlivých komponent dle předchozího, co všechno umí (např. přidání / odebrání hodnoty energie za použité zámku (není transakce))

3.5 Bloky v herním světě

- je více mmožností. Uchování pole 50000 x 50000 x 25000 // todo ověřit je nesmysl.

- nepotřebujeme otevřený svět bez mřížky (pozdější aktualizace ME, jinak SE), takže budeme hledat nějakou variantu stromové struktury

- nabízí se možnost clustorování budov a shlukování do skupin, s následnou optimalizací počtu hladin

- my jsme zvolili K-D strom kombinovatný s AABB. (proč?)

- náš strom má optimalizaci jedinného potomka, v případě potřeby se dogeneruje do úrovně níže, případně rozpadne na podčásti a rekurzivně se přidá.

- díky této variantě se můžeme snadno dotazovat na sousedy, což je hlavní cíl (proto)

3.6 Počasí

- počasí chceme proměnlivé ale s tím, že gamedesignéři mohou snadno ovlňovat výsledné počasí, případně aby šlo snadno rozšířit varianty pro různé herní módy

-budeme mít ve světě umístenou entitu (Pawn) ovládaný AI Controllerem - to z toho důvodu, že pro AI Controller můžeme použít BehaviorTree

-popsat ideu BT

-další možnosti by byly, že bychom prostě použili update smyčku nějakého Actora - není potřeba, tohle se vyřeší updatem na komponentě

3.7 Hráčova postava

- pohled 1st person, 3rd person
- má komponenty kyslíku, energie
- může stavět, interagovat s bloky
- může zařvat

3.8 Inventář

- je to vlastnost hráče
- v inventáři má několik přepínatelných banků
- bank může být se stavitelnými bloky nebo s intentárními předměty
- bank je možné filtrovat
- důvod pro použití banku - rychlé přepnutí při stavění (minecraft složitá organizace při stavění a použití 10+ druhů bloků)

3.9 Ukládání hry

- vše se musí korektně uložit
- ?? specifikace binárního formátu zde, nebo v programátorský?

3.10 Doplňující vlastnosti

3.10.1 DLC?

- můžeme dodávat extra bloky (ukázka!)
// toto budeme zmiňovat? - byly problémy s inicializací elektrické komponentyy, takže leda tak statické meshe, co umí prd

3.10.2 Lokalizace

- použití lokalizace

3.10.3 Hudba

- atmosférický hudební doprovod

3.11 Backlog

- Popsat, že bychom chtěli nějaké UI + nabídky menu

4. Programátorská dokumentace

4.1 Počáteční inicializace projektu

Pokud je cílem spustit projekt hry ze zdrojových kódů, je potřeba si stáhnout Unreal Engine ve verzi 4.15 (TODO link!). Použití novější verze je možné, ale běžnému uživateli to nedoporučujeme. Mezi verzemi se mohou projevit nekompatibility v kódu, které je případně nutné řešit zásahy přímo do zdrojových kódů hry. Dále je potřeba mít k dispozici zdrojové kódy ať už z DVD, nebo z tohoto release na GitHubu (TODO link na public repo, release commit).

Dále je zapotřebí vygenerovat solution pravým klikem na uproject file (TODO img!) Pokud tato možnost v kontextové nabídce není, je potřeba provést FIX . Ze zkušenosti autora - toto se mnohdy nemusí podařit. Pokud se nepodaří vygenerovat solution, může stačit otevřít projekt a dát zkompilovat chybějící binárky (TODO img!). Je zapotřebí mít VS 2015 alespoň ve verzi Community.

Pokud i toto selže, ověřte si, prosím, že je možné založit nějaký projekt založený na C++ (todo font), zkompilovat jej a taktéž vygenerovat solution. Pokud se to povede s template, mělo by to fungovat i s tímto projektem.

Dalším krokem je v případě úspěšného otevření ve VS (todo abrreviation) nastavení TCF2 jako výchozího projektu a následné spuštění. Dále by měl následovat krok spuštění Play in Editor v UE.

Pokud vše selže, je možné nalézt příčinu chyby v logu (TODO) v Saved/Logs

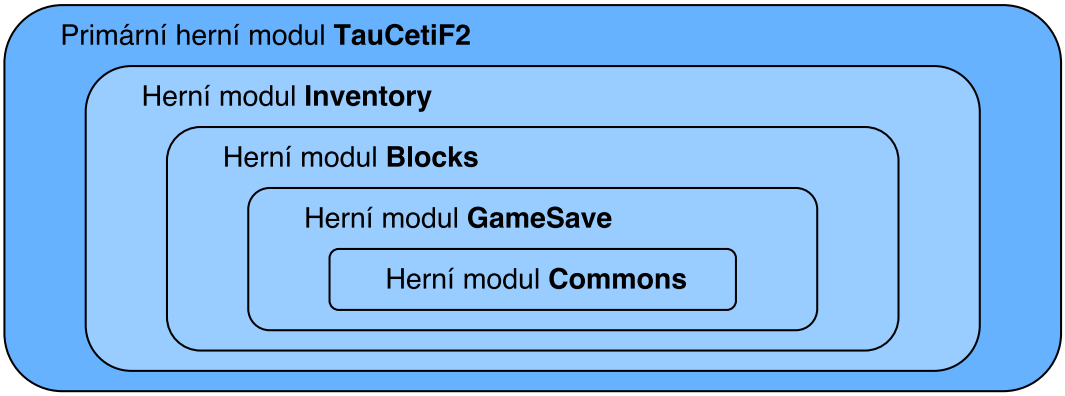
4.2 Struktura kódu

Protože jsme zvolili implementaci práce v UE, můžeme využít toho, že engine umožňuje rozdělit celý herní projekt do jednotlivých herních modulů[6]. Tím docílíme modularity, nebudeme mít celý projekt v jednom kuse a zároveň tím urychlíme překlad projektu při kompilaci.
Každý UE projekt musí definovat právě jeden primární herní modul. Pokud využijeme možnosti vytvoření nového projektu založeného na C++, editor tento modul automaticky vytvoří za nás. My jsme pojmenovali náš projekt `TauCetiF2` a tak se jmenuje i náš primární modul.

Jednotlivé části projektu jsme rozdělili do několika herních modulů:

1. TauCetiF2 (primární modul)
2. Inventory
3. Blocks
4. Game Save
5. Commons

Herní moduly jsme seřadili dle jejich závislosti tak, že každý modul závisí na všech modulech s vyšším očíslováním. Tedy primární modul využívá všech ostatních modulů a poslední modul (Commons) není závislý na žádném dalším modulu. Pro lepší představu můžeme tyto souvislosti vyjádřit obrázkem 4.1:



Obrázek 4.1: Diagram závislostí modulů projektu.

4.2.1 Struktura modulu

Všechny moduly dodržují společnou strukturu. Obsahují:

- složku `Public`
- složku `Private`
- soubor `[název_modulu].Build.cs`
- soubory `[název_modulu].h`, `[název_modulu].cpp`

Každý modul pak má hlavičkové soubory ve složce Public, implementaci tříd pak ve složce Private. Poslední tři soubory jsou kvůli UBT a použitím herních modulů v rámci UE

4.2.2 Modul Commons

Tento modul je základním modulem, který na jednom místě definuje všechny potřebné informace, které využívají ostatní moduly. Jedná se zejména o definici herních konstant (GameDefinitions.h), či definice vších sdílených enumerátorů (Enums.h). Najdeme zde také prapředka použité herní instance

// TODO link na ue docs

Tuto vlastní implementaci herní instanci využijeme pro ukládání nalezených bloků.

4.2.3 Modul Game Save

Modul GameSave slouží k ukládání a načítání informací o probíhající hře do binárního formátu. K tomu používáme streamové operátory <<, které jsou v tomto případě implementovány tak, že je možné je použít jak pro ukládání, tak pro načítání. // TODO link na tutorial

Díky tomuto přístupu tak můžeme definovat celou strukturu výsledného binárního souboru na jednom místě a tedy rozšiřování uložené hry je triviální. Co si ovšem musíme pohlídat je to, abychom si drželi informaci o verzi uloženého souboru. V našem případě, pokud se bude lišit verze načteného souboru a uložená konstanta v programu, save prostě odmítneme (a dokonce smažeme). V

produkčním prostředí bychom si mazání nemohli dovolit, ale museli bychom save ignorovat a uživateli zobrazit nějakou hlášku o tom, že verze souboru není podporovaná. My jsme se však v tomto případě rozhodli save mazat, protože jsme očekávali, že během vývoje hry se bude binární struktura savu často rozšiřovat. Po každé iteraci jsme si savy prostě vytvořili nové.

Co by se stalo, kdybychom se snažili načíst save jiné verze? Celá hra by nejspíše byla ukončena s chybou, protože by se pokoušela číst neplatná data a/nebo by očekávala nějaká data tam, kde žádná nejsou. Tím bychom četli z neplatné lokace.

- popsat save game carrier (+ výsledný formát)

- zdůraznit, že se jedná o holá data, UObjekty si pak vytváří každý modul sám

- popsat NewSaveGameHolder, rozšiřování pevných savů

- popsat *Archive helpers

4.2.4 Modul Blocks

Modul bloků obsahuje podstatné informace o tom, jak hra pracuje s bloky, jak se tyto bloky skládají do herního světa, jaké jsou jejich komponenty atd.

- základní definice bloku je v (Block.h)

- block.h definuje hromadu společných věcí tak, aby nějaké základní bloky bylo možné implementovat třeba komplet v BP a neřešit vůbec kód. (To neplatí pokud blok má třeba Electricity component - díky odložené inicializaci by se správně nepropisovaly infa apod)

Komponenty bloků

- pak máme komponenty bloků a nějaké interfaces

Definice bloků

- popsat způsob definice bloků

Nalezení bloků

- popsat block holder a co všechno pro nás znamená

Ukládání

- ukládání - máme něco jako block saving helpers

(Jednotlivé implementace)

-popsat vstrvení)to se použíá i v BP - strom základní tvary / speci. impl (elektr, kyslík) apod

Základní tvary

- 3 varianty krychle

Speciální

- popsat speciální bloky + nějaké speciality co umějí (showableWidget)

Hurá na stromy

popsat stromové struktury, které tam mám

4.2.5 Modul Inventory

Modul inventáře byl vyčleněn do samostatné části. Je to hlavně jako ukázka možného členění do modulů. Navíc časem by se mohl tento modul rozšířovat jak by rostla kompelxicita správy inventáře.

Nejdůležitější inventory component

4.2.6 Modul TauCetiF2

- primární modul

popsat co všechno obsahuje (widgety, gamemodes, weather apod)

- popsat synchronize widget (// TODO link na důvod, proč to tam mám), popsat object widget, napsat důvody

- popsat to stackování

- popsat komponenty (weather, game electricity)

4.3 Struktura projektu v Unreal Enginu

- ukázat jak se to dělí v UE editoru

4.3.1 Backlog

Zde popsat jak jsem to celé implementoval a proč

Popsat jednotlivé moduly a nakreslit diagram vztahů mezi nimi

Popsat strukturu save gamu + důvod proč jsem to tak udělal +popsat načítání savů + systémových savů

Popsat jednotlivé C++ třídy a jejich odvozené Blueprintové deriváty + přidat případné obrázky z BL kódu (např. BlueprintImplementable event, který se zavolá jak na C++ tak i na BP)

Udělat rozbor BT počasí + mechaniku počasí + denního cyklu popsat řízení osvětlení dle počasí

Udělat rozbor bloků, škálování, konfigurace, datovou strukturu, implementaci dynamických textur, zvýraznění

Popsat mechaniku Selector - SelectTarget + napojení na Builder

Popsat mechaniku používání objektů + zvýraznění

Popsat mechaniku Inventáře

// TODO vymyslet vhodné pořadí, abych neskákal mezi prvky, toto pořadí dodržet i v předchozích kapitolách

-> Mám svět, ten má v sobě bloky, ty jsou v nějaké stromové struktuře, bloky mají komponenty, které přes tuto strukturu mohou na sebe vázat Svět má také

počasí se svojí vlastní strukturou, vuyžívající podobnosti s bloky (2D KD strom s Heapem na listech)

-> hráč může to a tamto, díky inventáři se dostane na bloky, a díky selectoru je pak můževložit do světa skrz World controller (zmíněno v předchozím) ->zároveň jsou všechny entity savovatelné

-> Popsat struktury Widgetů, zmínit použití Synchronize Widgetu, implementaci mechaniky stackovatelných widgetů

-> popsat implementaci hudby

-> TODO otestovat možnost nového bloku v rámci DLC ->Zmínit zároveň, že s tímto by šlo tweakovat nastavení hry

// TODO obrázky s konfiguračními ukázkami do příloh (např. jak se definuje Blok z UE

5. Uživatelská dokumentace

Tato část obsahuje informace o tom, jak hru spustit a jaké jsou požadavky ke spuštění. Dále jsou zde uvedeny obrázky ze hry a popis toho, co znamenají.

5.1 Požadavky pro spuštění hry

5.1.1 Hardwarové požadavky

Doporučená minimální sestava (na ní byla hra vyvíjena):

Procesor:	Intel i7-2630QM @ 2.00GHz
RAM:	12 GB (8 GB by mělo také stačit)
Grafika:	ATI Radeon HD 6700M
OS:	Win 10 x64 (7 a vyšší by měly být v pohodě)

Výše uvedneou konfiguraci je potřeba brát jako orientační. Hru jsme úspěšně spustili i na notebooku s procesorem Intel i5, integrovanou grafickou kartou a 8 GB operační paměti. Bylo však nutné nastavit grafické vlastnosti na minimální možnou konfiguraci.

5.1.2 Softwarové požadavky

Pro spuštění zkompilované hry není potřeba nic speciálního. Je zapotřebí mít stroj s minimální uvedenou konfigurací. Dále je dobré mít nainstalované poslední verze ovladačů HW komponent (hlavně grafiky). Taktéž je zapotřebí mít nainstalovanou poslední verzi DirectX.

// TODO povolit obrázky

6. Závěr

6.1 Zhodnocení práce

6.2 Budoucí práce

- dynamičtější mřížka? 20cm je nejspíše dost málo a vyžaduje to dost preciznosti // TODO zkusit pro test 25 či 30 cm a patřičným způsobem upravit velikosti modelů? (nejspíše to musí zůstat hardcoded, ale zkusím se nad tím zamyslet, pokud bude čas)
- vlastní sortování v seznamech

TODO dotazník?

Seznam použité literatury

[1] Minecraft Wiki. Block, . URL http://minecraft.gamepedia.com/Block.

[2] Minecraft Wiki. Tutorials/units of measure, . URL http://minecraft.gamepedia.com/Tutorials/Units_of_measure.

[3] dillonsup. Minecraft vs terraria (facts). URL http://www.minecraftforum.net/forums/minecraft-discussion/discussion/178129-minecraft-vs-terraria-facts. internetové fórum.

[4] GameSpot. Space engineers. URL https://static.gamespot.com/uploads/original/1365/13658182/2626082-space-engineers1.jpg.

[5] Erik Fagerholt; Magnus Lorentzon. Beyond the hud - user interfaces for increased player immersion in fps games. Master's thesis, 2009. URL http://publications.lib.chalmers.se/records/fulltext/111921.pdf.

[6] Inc Epic Games. Gameplay modules. URL https://docs.unrealengine.com/latest/INT/Programming/Modules/Gameplay/.

Přílohy