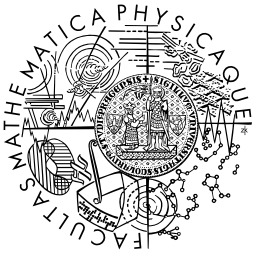


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

**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

# Úvod

V době vzniku této práce jsou velice populární hry s otevřeným světem a velkými možnostmi stavění. Autor této práce si je taktéž rád zahraje a rád by touto prací představil svoji vizi dalšího možného rozvoje tohoto žánru.

Když se podíváme na hry jako například Minecraft, Space Engineers (nebo její odnož Medieval Engineers) tak zjistíme, že hra od hráče vyžaduje nejen stavitelské a konstruktérské schopnosti, ale také taktické dovednosti, které hráči umožňují v daném světě přežít. Mnohdy hráč může využít nevšedních technik daných mechanikami dané hry. Příkladem budiž farma na golemy ve hře Minecraft, která využívá bloky lávy, které jsou drženy cedulemi(Minecraft Wiki).

V současné době jsou velikosti bloků omezeny na konstantné velikost. Ve hře Minecraft je blok hranově omezen na 1m, hra Space Engineers bloky omezuje dle kategorií od 0.5 m do 2.5 m (Space Engineers WIKI). My bychom se v této práci chtěli zabývat myšlenkou proměnlivé velikosti stavitelných bloků. Hráči by tak mohli ovlivňovat detailnost svých výtvorů, aniž by to muselo mít nutně negativní vliv na dobu nutnou k postavení komplexního, ale zároveň detailního výtvoru. Tato myšlenka však přináší spoustu problémů k řešení, které se však v této práci snažíme vyřešit.

Aby hra byla nebyla pro hráče nudná, měla by také obsahovat nějakého nepřítele - tedy něco, co bude hráče nutit se zlepšovat, překonávat nástrahy a posouvat se dále. I tomuto elementu hry se v této práci budeme věnovat. Zároveň nám to nabízí nové možnosti, jak do hry dodat prvky strategických her. Aby to hráč neměl tak jednoduché, bude muset taktizovat, aby svého nepřítele porazil, nebo alespoň v dané chvíli neprohrál, byť za cenu nějakých ztrát.

- V poslední době jsou kromě klasických her typu FPS také různé strategie a hry o přežití
- Typ: Minecraft, Space Engineers, Medieval Engineers, Take on Mars, ARK Survival Evolved, novější No man's sky
- autor tyto hry má též v oblibě a rád by představil svoji vizi budovatelské hry s prvky strategie a přežití // TODO tohle by chtělo hodně učesat

// TODO zmínit příběh, který jsem si vymyslel? (cesta za záchranou lidstva atd...)?

// možná bych to mohl dát do hry na úvod, poté zobrazit textový tutorial

# 1. Analýza zadání

## 1.1 Stávající implementace mechanismů

- v dalším textu budeme vycházet z her:

Minecraft
Space Engineers / Medieval Engineers

- popsat, jak je to v jednotlivých zmíněných hrách (musel jsem je nutně hrát všechny?)
- popsat velikosti bloků, nějaké zákonitosti, fyziku
- popsat strategické mechaniky

## 1.2 Rozbor zadání

Zde bychom měli popsat, co by se nám ve hře líbilo a stanovit reálnost implementace

Nejspíše se text bude prolínat s kapitolou - dalším vývojem?
Měl bych si tu vysnit celou hru, nebo to spíše seškrtat?

- tedy že bychom chtěli panďuláka, jaké pohledy
- a že bychom s ním chtěli chodit a stavět a bourat
- ale že nám taky může umřít - počasí, kyslík
- popsat svět, bloky, co by asi měly umět

## 1.3 Cíle práce

# 2. Detailní analýza

## 2.1 Použitý herní engine

- máme několik možností:
- napsat si vlastní (ne, moc práce)
- použít low level (XNA) - opět ne, moc práce
- Unity nebo Unreal Engine
- Unity mělo alespoň v době analýzy této práce problémy s dynamickým navmeshem, oproti tomu mělo editovatelný terén. Další nevýhoda je absance editorů materiálu tk jak je tomu v UE
- Unreal je prostě nej

## 2.2 Bloky

- Herní svět je složen z bloků
- velikost bloků je omezena, minimální velikost bloku je 20 na 3 cm
- maximální velikost bloku je 20ti násobek
- mám různé tvary - krychle, stranově seříznutá krychle, tělně říznutá krychle (obrázky)
- pak jsou zde i speciální tvary - ty bylo nutné vymodelovat v Cinemě4D
- speciální tvary definují svoji pevnou velikost, od této definice se pak odvíjí další vlastnosti (výpočet zdraví, energie bloku)

### 2.2.1 Standardní bloky

- ty různé krychle

### 2.2.2 Speciální bloky

- popsat speciality

## 2.3 Vlastnosti bloků

- bloky mohou mít několik vlastností:
- mít možnost elektriky a zapojení do elektrické sítě
- mít možnost uchování kyslíku, v případě použitií elektirky pak i generování
- bloky mohou být použitelné, tj. hráč s nimi může nějakým zůsobem interagovat
- bloky mohou být sebratelné, tedy hráč si je může dát do svého inventáře. vlastnosti jako třeba uchovaná hodnota kyslíku, pak zůstávají zachované
- bloy mohou být zákadem pro rozpoznávání tvarů (TODO )

### 2.3.1 Elektrika

### 2.3.2 Kyslík

### 2.3.3 Označovatelnost

### 2.3.4 Možnost vzít do inventáře

### 2.3.5 Interakce

## 2.4 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)

## 2.5 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

## 2.6 Hráčova postava

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

## 2.7 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)

## 2.8 Ukládání hry

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

## 2.9 Doplňující vlastnosti

### 2.9.1 DLC

- můžeme dodávat extra bloky (ukázka!)

### 2.9.2 Lokalizace

- použití lokalizace

### 2.9.3 Hudba

- atmosférický hudební doprovod

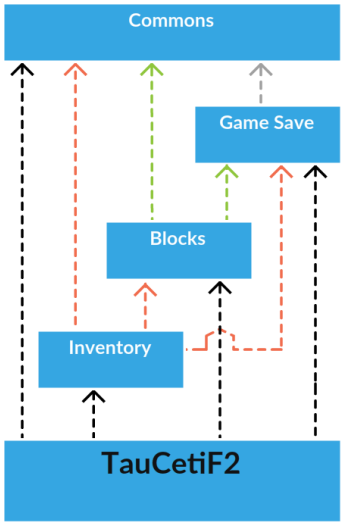
# 3. Backlog

- Svět + jak vypadá + jak je reprezentován (K-D tree zde, nebo v programátorské?)
- Popsat bloky, velikosti
- Popsat komponenty bloků (že je něco jako komponenta elektriky, komponenta vzduchu
- Popsat hratelnou postavičku (že má taky možnost elektriky a kyslíku
- Popsat počasí - že má taky svoji blokovou reprezentaci a že je na pozadí Behavior Tree, který to celé řídí (ovlivňování konfigurace v programátorské části)
- Popsat implementaci elektriky // TODO dodělat ve hře
- Popsat implementaci rozpoznávání tvarů // TODO dodělat ve hře
- Popsat způsob ukládání a načítání hry (to je možná až do Progr. sekce?)
- Popsat, že bychom chtěli nějaké UI + nabídky menu
- Popsat, že bychom chtěli základní hudbu
- Popsat, že máme něco jako inventář s možností nějaké správy bloků
- Stejně tak pro builder + herní terminály //TODO doimplementovat

# 4. Programátorská dokumentace

## 4.1 Struktura kódu

Program se dělí do několika modulů. Jejich struktura je zachycena na obrázku 4.1.




Obrázek 4.1: Diagram závislostí modulů projektu.

Když si otevřeme zdrojovou sln projektu, tak uvidíme, že každý modul má několik podřízených věcí:
- složku Private
- složku Public
- soubory .Build.cs, .h, .cpp

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

### 4.1.1 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.

### 4.1.2 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

### 4.1.3 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

### 4.1.4 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

### 4.1.5 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)

## 4.2 Struktura projektu v Unreal Enginu

- ukázat jak se to dělí v UE editoru

# 5. 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

# 6. Uživatelská dokumentace

Obrázky s UI nabídkami, obrázky ze hry

# 7. Závěr

## 7.1 Zhodnocení práce

## 7.2 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

Minecraft Wiki. Iron golem farming. Web. URL http://minecraft.gamepedia.com/Tutorials/Iron_golem_farming.

Space Engineers WIKI. Blocks. URL http://spaceengineers.wikia.com/wiki/Category:Blocks.

# Seznam obrázků

# Seznam použitých zkratek

TODO seznam zkratek

# Přílohy