



# 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 takéž 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

<b>1</b>	<b>Úvod</b>	<b>3</b>
1.1	Herní bloky . . . . .	3
1.2	Inventář . . . . .	3
1.3	Herní nepřítel . . . . .	4
1.4	Cíle práce . . . . .	4
<b>2</b>	<b>Analýza zadání</b>	<b>5</b>
2.1	Stávající implementace mechanismů . . . . .	5
2.1.1	Blokы . . . . .	5
2.1.2	Herní svět . . . . .	5
2.1.3	Inventář . . . . .	6
2.1.4	Avatar hráče . . . . .	6
2.2	Co bychom chtěli implementovat . . . . .	6
2.2.1	Blokы . . . . .	6
2.2.2	Herní svět . . . . .	8
2.2.3	Inventář . . . . .	9
2.2.4	Avatar hráče . . . . .	9
2.3	Backlog . . . . .	9
<b>3</b>	<b>Detailní analýza</b>	<b>10</b>
3.1	Herní engine . . . . .	10
3.1.1	Vlastní engine . . . . .	10
3.1.2	Nízkoúrovňový framework . . . . .	10
3.1.3	Existující herní engine . . . . .	10
3.2	Blokы . . . . .	11
3.3	Vlastnosti bloků . . . . .	12
3.3.1	Energie . . . . .	12
3.3.2	Energetická síť . . . . .	12
3.3.3	Kyslík . . . . .	12
3.3.4	Označovatelnost . . . . .	12
3.3.5	Možnost vzít do inventáře . . . . .	12
3.3.6	Interakce . . . . .	12
3.3.7	Zapojení do rozpoznávání tvarů . . . . .	12
3.4	Komponenty bloků . . . . .	12
3.5	Blokы v herním světě . . . . .	13
3.6	Počasí . . . . .	13
3.7	Hráčova postava . . . . .	13
3.8	Inventář . . . . .	13
3.9	Ukládání hry . . . . .	13
3.10	Doplňující vlastnosti . . . . .	14
3.10.1	DLC? . . . . .	14
3.10.2	Lokalizace . . . . .	14
3.10.3	Hudba . . . . .	14
3.11	Backlog . . . . .	14

<b>4 Programátorská dokumentace</b>	<b>15</b>
4.1 Struktura kódu . . . . .	15
4.1.1 Struktura modulu . . . . .	15
4.1.2 Modul Commons . . . . .	16
4.1.3 Modul Game Save . . . . .	16
4.1.4 Modul Blocks . . . . .	16
4.1.5 Modul Inventory . . . . .	17
4.1.6 Modul TauCetiF2 . . . . .	18
4.2 Struktura projektu v Unreal Enginu . . . . .	18
4.2.1 Backlog . . . . .	18
<b>5 Uživatelská dokumentace</b>	<b>19</b>
5.1 Hlavní menu . . . . .	19
5.2 Herní menu - ukládání, nahrávání . . . . .	21
5.3 Inventář . . . . .	23
5.4 Terminál . . . . .	27
5.5 Stavební akce . . . . .	31
5.6 Umístitelné předměty . . . . .	33
5.7 Plnička kyslíkových bomb . . . . .	34
5.8 Přepínač . . . . .	36
5.9 Světlo . . . . .	37
5.10 Kyselý déšť . . . . .	37
<b>6 Závěr</b>	<b>41</b>
6.1 Zhodnocení práce . . . . .	41
6.2 Budoucí práce . . . . .	41
<b>Seznam použité literatury</b>	<b>42</b>
<b>Seznam obrázků</b>	<b>43</b>
<b>Seznam použitých zkratek</b>	<b>44</b>
<b>Přílohy</b>	<b>45</b>
A Minimální HW konfigurace . . . . .	45
B Požadavky na zprovoznění hry . . . . .	46

# 1. Úvod

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

Tyto hry se vyznačují tím, že hráč využívá stavebních bloků k tomu, aby rozšiřoval své herní možnosti v těchto světech. Dalším společným prvkem je přítomnost zdraví hráčova avatara (postavy, za kterou hráč hraje) a dalších survival atributů. Dokonce můžeme tvrdit, že je hráč nucen využívat alespoň určité minimum bloků, protože jinak by v herním světem nepřežil a tím hra obvykle končí.

## 1.1 Herní bloky

Některé hry (kupříkladu *Minecraft* nebo *Terraria*) využívají stavebních bloků jako základního konceptu a tyto bloky pak tvoří celý herní svět. Tím je dosažena jednotná stylizace herního světa. Jiné (*Space Engineers*, *Medieval Engineers*) využívají kombinaci herních bloků s voxelovou reprezentací světa a tím dosahují vyššího stupně realismu ve hře. Můžeme však nalézt i dalsí příklady her (// TODO *Take on Mars*, *Novus Inceptio*, *Planet Nomads*, *ARK Survival Evolved*, *No man's sky*).

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 uvedeme některé výtvory ze hry *Minecraft*:

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

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.

V této práci se chceme zaměřit na to, aby hráč mohl ovlivnit velikost stavěných bloků. Tedy aby mohl snadno stavět rozsáhlé struktury, ale zároveň aby se mohl plípat s detaily, pokud je to hráčovým cílem.

## 1.2 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), 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 rychlých nabídek. 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

## 1.3 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.

## 1.4 Cíle práce

Tato práce by měla naplnit následující cíle:

- Bloky
- 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

##### 2.1.1.1 Základní vlastnosti

rozměry

##### 2.1.1.2 Součásti bloků

komponenty elektřiny, inventáře

##### 2.1.1.3 Speciality

Multiblocks, náhled inventáře (*Medieval Engineers*)

#### 2.1.2 Herní svět

jaký je herní svět

##### 2.1.2.1 Reprezentace

MC - bloky, chuncks

##### 2.1.2.2 Bloky v herním světě

do gridu, start-free grid

### **2.1.2.3 Denní / noční cyklus**

obvykle tam je

### **2.1.2.4 Herní překážky**

počasí, Non-playable character (NPC), atributy avataru

### **2.1.2.5 (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

#### **2.2.1.1 Standardní bloky**

- ty různé krychle

#### **2.2.1.2 Kvádr**

- všechny velikosti  
- má elektriku

#### **2.2.1.3 Kvádr seříznutý stranou**

- všechny velikosti  
- má elektriku

#### **2.2.1.4 Kvádr říznutý tělem**

-všechny velikosti  
- má elektriku

### **2.2.1.5 Kvádr – základový**

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

### **2.2.1.6 Kvádr seříznutý stranou – základový**

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

### **2.2.1.7 Kvádr – polykarbonát**

- všechny velikosti
- má elektriku

### **2.2.1.8 Speciální bloky**

- popsat speciality

### **2.2.1.9 Napájené okno**

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

### **2.2.1.10 Terminál**

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

### **2.2.1.11 Dveře**

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

### **2.2.1.12 Kyslíková nádoba**

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

### **2.2.1.13 Plnička kyslíkové nádoby**

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

### **2.2.1.14 Baterie**

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

### **2.2.1.15 Generátor bloků**

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

### **2.2.1.16 Generátor energie**

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

### **2.2.1.17 Světlo**

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

### **2.2.1.18 Spínač**

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

### **2.2.1.19 Základní vlastnosti**

rozměry

### **2.2.1.20 Součásti bloků**

- komponenty elektřiny, inventáře
- dále také propojovatelnost

## **2.2.2 Herní svět**

jaký chceme herní svět

### **2.2.2.1 Reprezentace**

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

### **2.2.2.2 Bloky v herním světě**

do gridu

### **2.2.2.3 Denní / noční cyklus**

dáme ho

### **2.2.2.4 Herní překážky**

počasí, , atributy avataru

### **2.2.2.5 (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 záporu:

### 3.1.3.1 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 navmeshu. To by byl při umisťování bloků zásadní problém.
  - Další nevýhodou je podpora materiálů, kdy bychom snadným způsobem nedosáhli vizuálně přitažlivého prostředí.

### 3.1.3.2 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 použití 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 zásobem 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 množností. 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 clustrování 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 rekursivně 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 ovládat výsledné počasí, případně aby šlo snadno rozšířit varianty pro různé herní módy
  - budeme mít ve světě umístěnou 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 stavěními 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 komponenty, 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 Struktura kódu

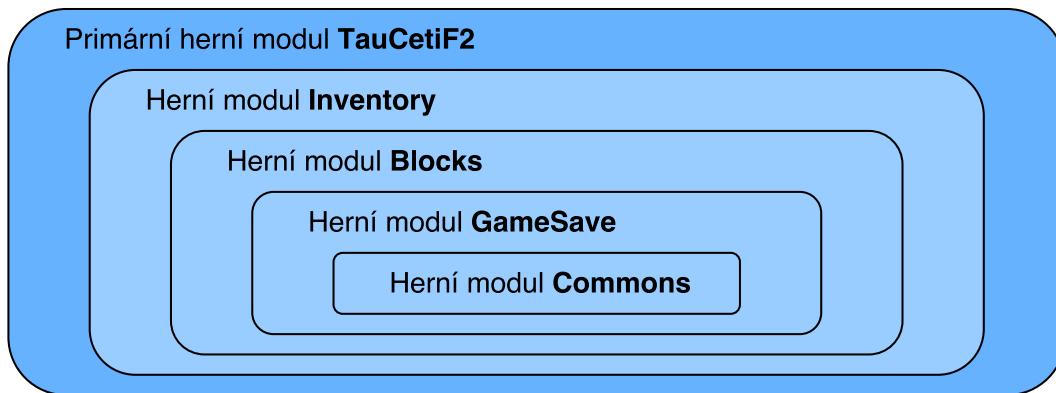
Protože jsme zvolili implementaci práce v UE, můžeme využít toho, že engine umožňuje rozdělit celý herní projekt do jednotlivých herních modulů[1]. Tím docílíme modularity, nebudeme mít celý projekt v jednom kuse a zároveň tím urychlíme překlad projektu při komplikaci.

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.

### 4.1.1 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

#### 4.1.2 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 všech 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.

#### 4.1.3 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

#### 4.1.4 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 infy apod)

#### 4.1.4.1 Komponenty bloků

- pak máme komponenty bloků a nějaké interfaces

#### 4.1.4.2 Definice bloků

- popsat způsob definice bloků

#### 4.1.4.3 Nalezení bloků

- popsat block holder a co všechno pro nás znamená

#### 4.1.4.4 Ukládání

- ukládání - máme něco jako block saving helpers

#### 4.1.4.5 (Jednotlivé implementace)

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

#### 4.1.4.6 Základní tvary

- 3 varianty krychle

#### 4.1.4.7 Speciální

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

#### 4.1.4.8 Hurá na stromy

popsat stromové struktury, které tam mám

### 4.1.5 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 rozširovat jak by rostla komplexita správy inventáře.

Nejdůležitější inventory component

#### 4.1.6 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
  - napsat to stackování
  - napsat komponenty (weather, game electricity)

### 4.2 Struktura projektu v Unreal Enginu

- ukázat jak se to dělí v UE editoru

#### 4.2.1 Backlog

Zde napsat 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 +napsat 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 napsat 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, využívající 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

-> napsat implementaci hudby

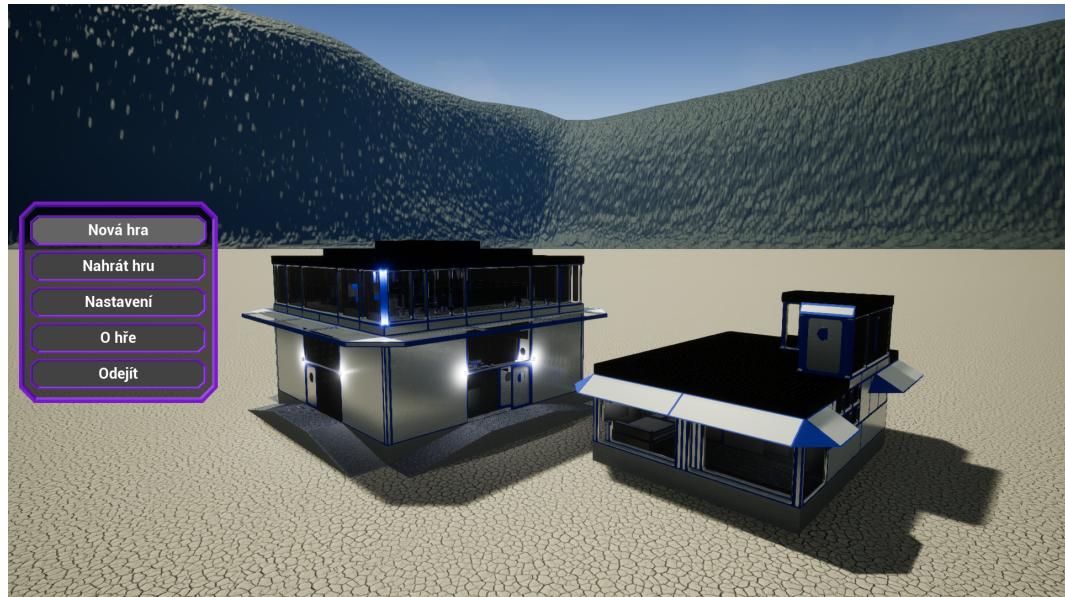
-> TODO otestovat možnost nového bloku v rámci DLC ->Zmínit zároveň, že s tímto by šlo tweakovat nastavení hry

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

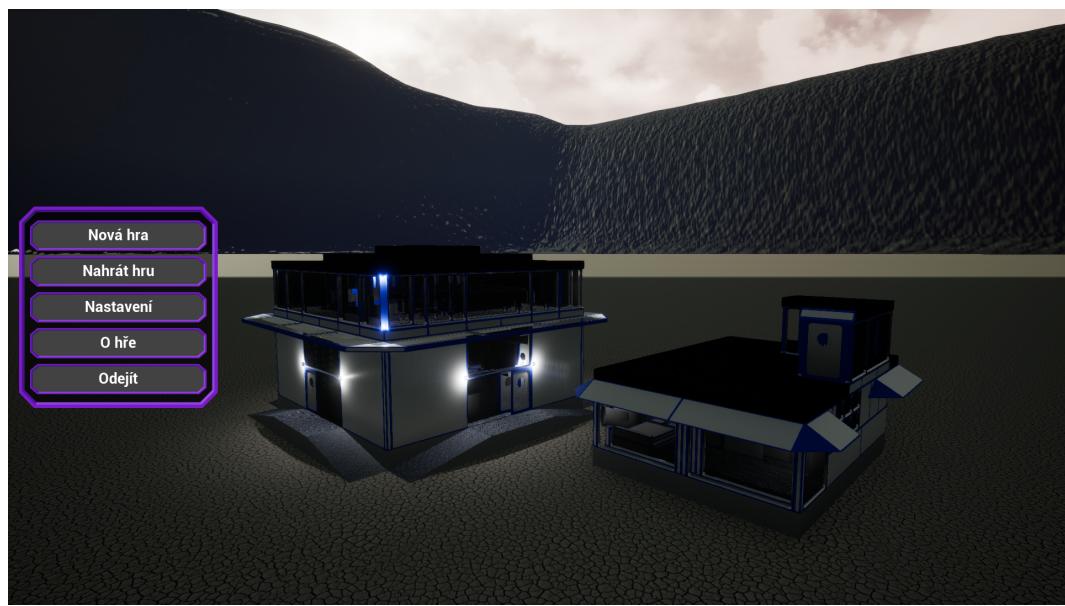
# 5. Uživatelská dokumentace

## 5.1 Hlavní menu

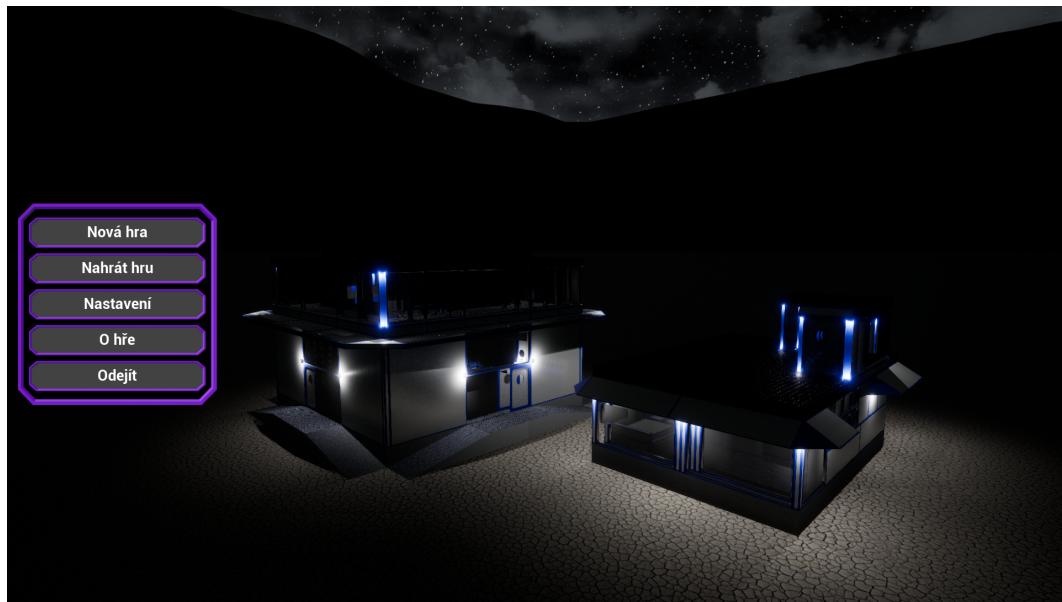
Po načtení hry hráč vidí hlavní menu. Denní doba i počasí se zvolí náhodně.



Obrázek 5.1: Obrazovka hlavního menu - den

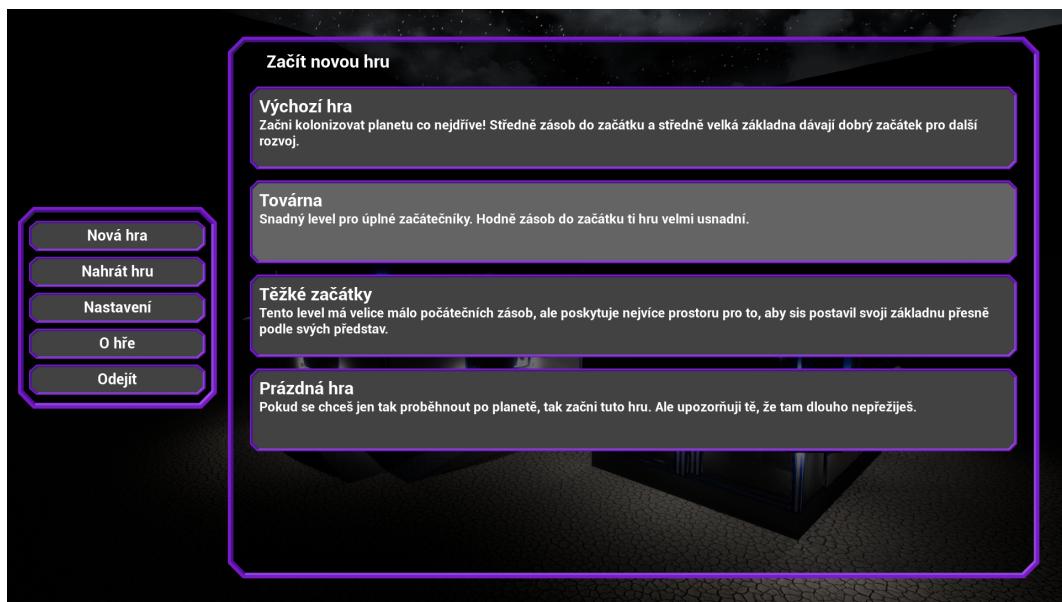


Obrázek 5.2: Obrazovka hlavního menu - poledne, zataženo



Obrázek 5.3: Obrazovka hlavního menu - noc

První volba, kterou je možné zvolutit, je výběr nové hry. K dispozici je několik variant s různými obtížnostmi.



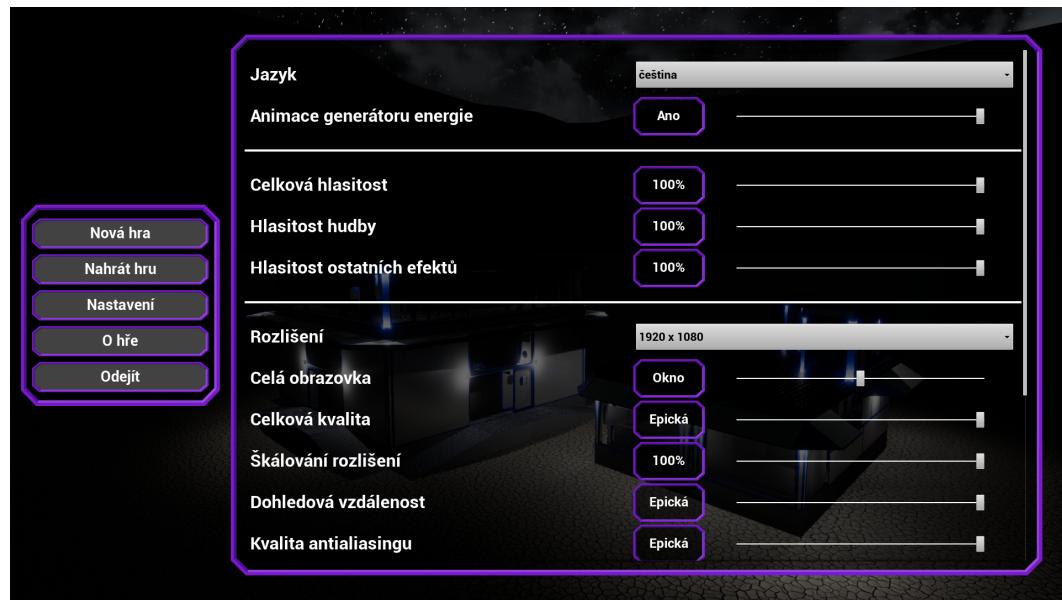
Obrázek 5.4: Obrazovka hlavního menu - Nová hra

Pokud hráč má nějaké uložené hry, může si je nahrát kliknutím na tlačítko **Nahrát hru**. V tomto případě žádné uložené hry k načtení k dispozici nejsou.



Obrázek 5.5: Obrazovka hlavního menu - Nahrát hru

Pod položkou **Nastavení** může uživatel měnit herní, zvuková a grafická nastavení hry. Nastavení se aplikují a ukládají okamžitě, výjimkou je pouze položka *Animace generátoru energie*, která se projeví až po změně levelu. V případě konfigurace z hlavní nabídky se nastavení projeví okamžitě, v případě konfigurace během rozehrané hry je potřeba vyvolat opětovné načtení uložené hry, nebo znova spustit novou hru.

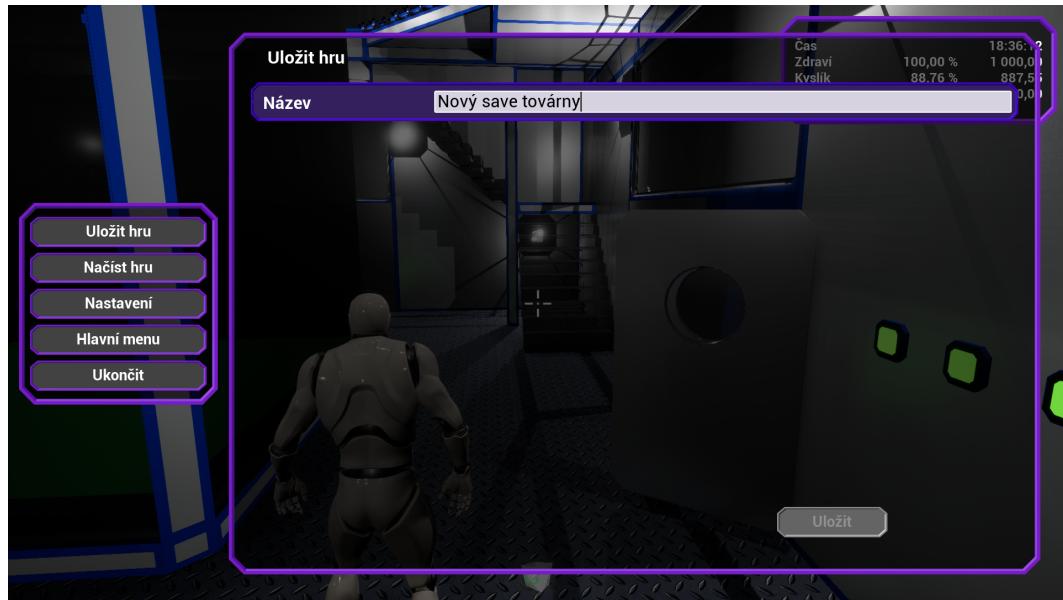


Obrázek 5.6: Obrazovka hlavního menu - Nastavení

## 5.2 Herní menu - ukládání, nahrávání

V průběhu rozehrané hry je možné použít **Rychlé uložení / načtení**, nebo si hru uložit z herní nabídky.

Pro uložení hry je možné vytvořit nový save nebo přepsat stávající, pokud nějaký existuje. Uložené hry je též možné mazat.



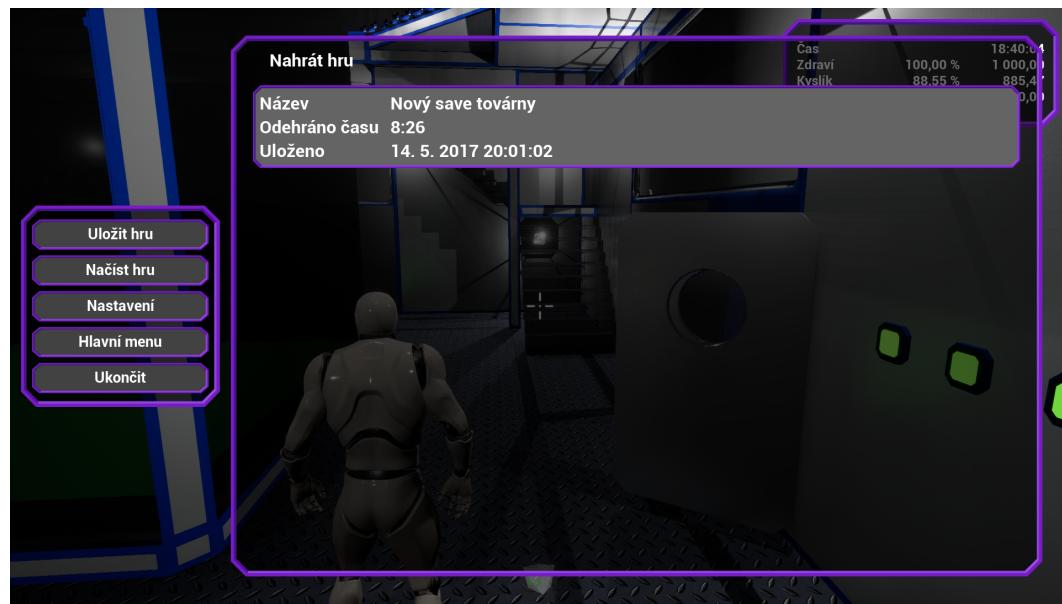
Obrázek 5.7: Ukládání - Nový save

Pokud bylo uložení úspěšné, hráči se zobrazí následující hláška:



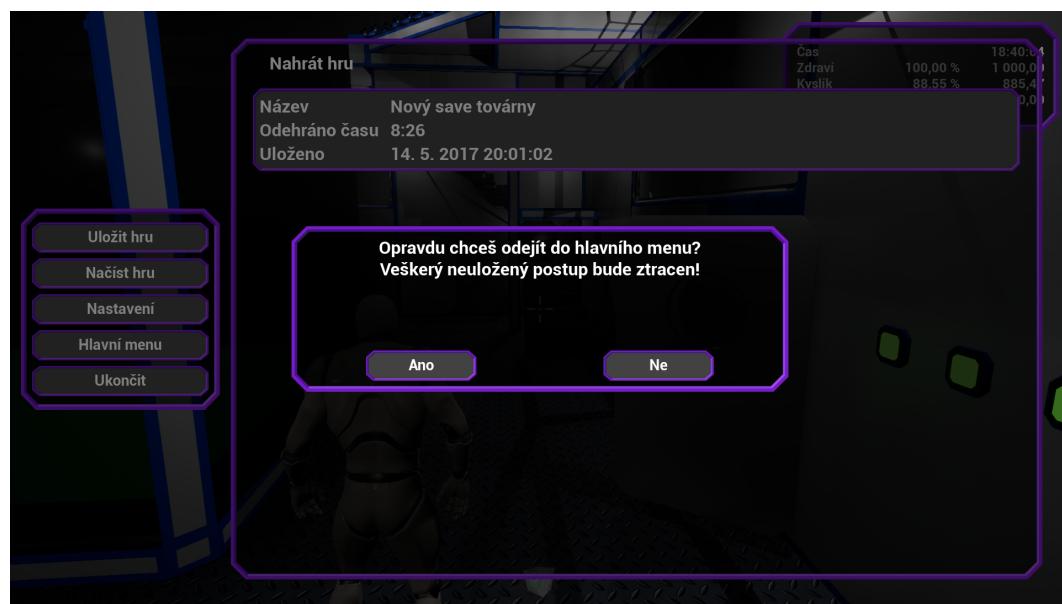
Obrázek 5.8: Ukládání - po uložení

Nahrávat je možné ze všech uložených pozic, včetně rychlého uložení. Opět je zde možné vybraný save smazat.



Obrázek 5.9: Ukládání - nahrát hru

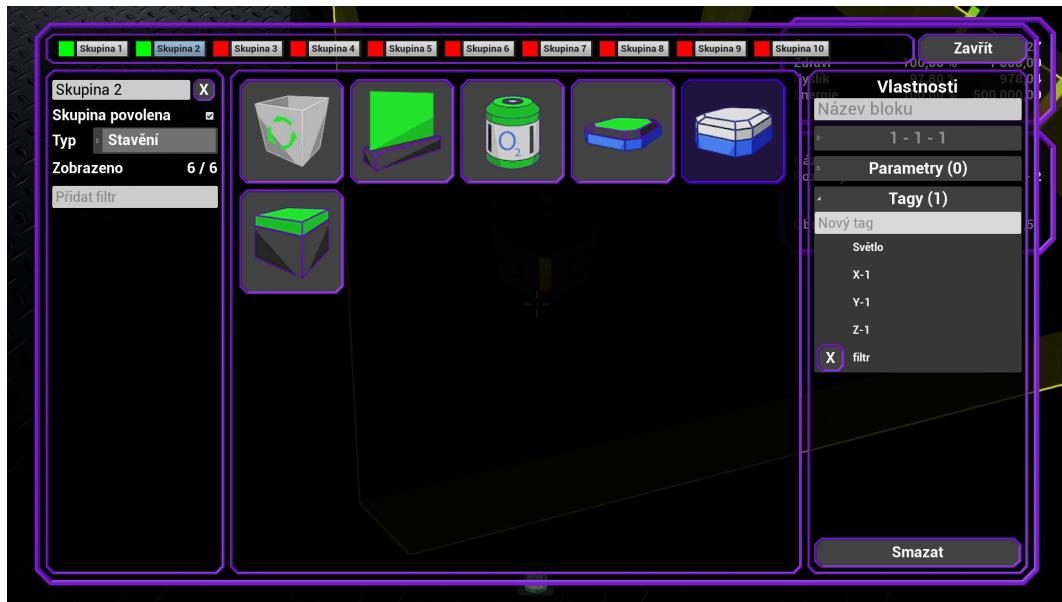
Pokud má hráč rozehranou hru, je pro jistotu vyžadováno potvrzení.



Obrázek 5.10: Ukládání - potvrzení nahrání hry

### 5.3 Inventář

Inventář se vyvolá klávesou **E**. Zobrazí se následující obrazovka:



Obrázek 5.11: Inventář - přehled

V horní části je výběr skupin. Je možné vybírat z 10 skupin, které si lze pojmenovat dle libosti. Dále je možné je (de)aktivovat buď kliknutím na zelený / červený čtverec, nebo skrze příslušný checkbox v editaci skupiny. Je možné používat standardní změnu skupiny klávesami **ú** a **)**, editační okno se pak příslušným způsobem změní.

V levé části je **editor skupiny**, jehož funkce budou popsány v textu dále. *Uprostřed* je možné vidět postavitelné či umístitelné položky, které jsou dofiltrovaný dle právě nastaveného filtru. Výběrem položky lze editovat její vlastnosti v *pravé* části okna

Každá skupina umožňuje definovat své filtry. Matematicky bychom mohli popsat filtrování jako vyhodnocení formule v *CNF*. Pokud do pole *Přidat filtr* napíšeme tagy oddělené mezerou, do skupiny se přidá **nová** skupina s výchozím pojmenováním.



Obrázek 5.12: Inventář - přidání skupiny

Po přidání se seznam dostupných prvků dofiltruje podle nastavených tagů - ve výsledku bude každá položka splňovat alespoň jeden tag z každé skupiny. Shoda nemusí být přesná, tag objektu musí obsahovat podřetězec definovaný filtrem.



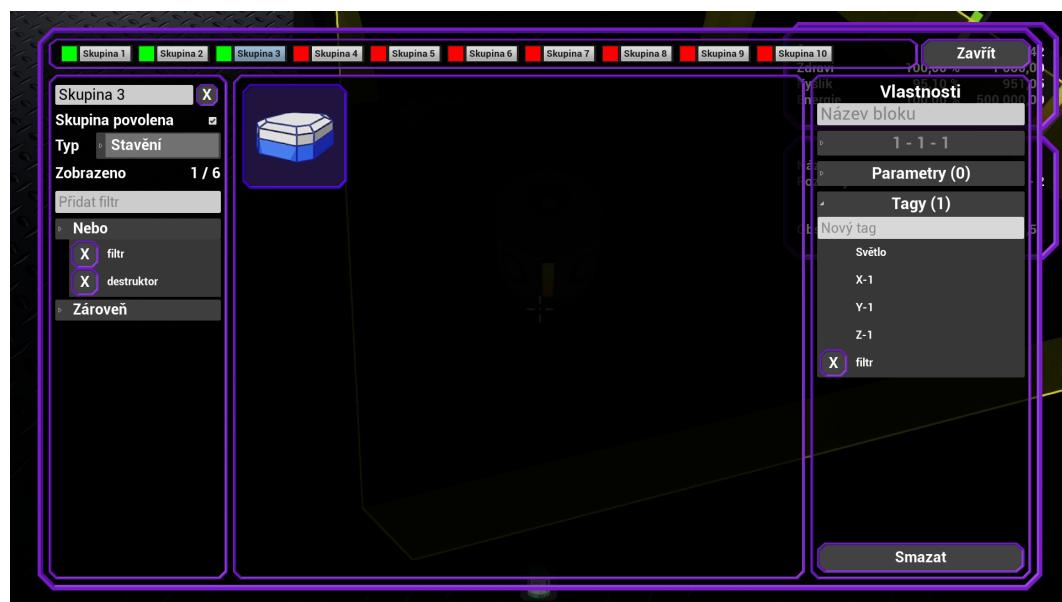
Obrázek 5.13: Inventář - výsledek s filtrovanými položkami

Na následujícím obrázku je možné vidět filtry s rozbalenými vlastnostmi. Změřme se nyní na část *Nesbalovat*. Po zaškrtnutí této volby je zobrazena pouze hlavička skupiny.



Obrázek 5.14: Inventář - Nová hra

Výsledný seznam se sbalenými vlastnostmi filtrov:



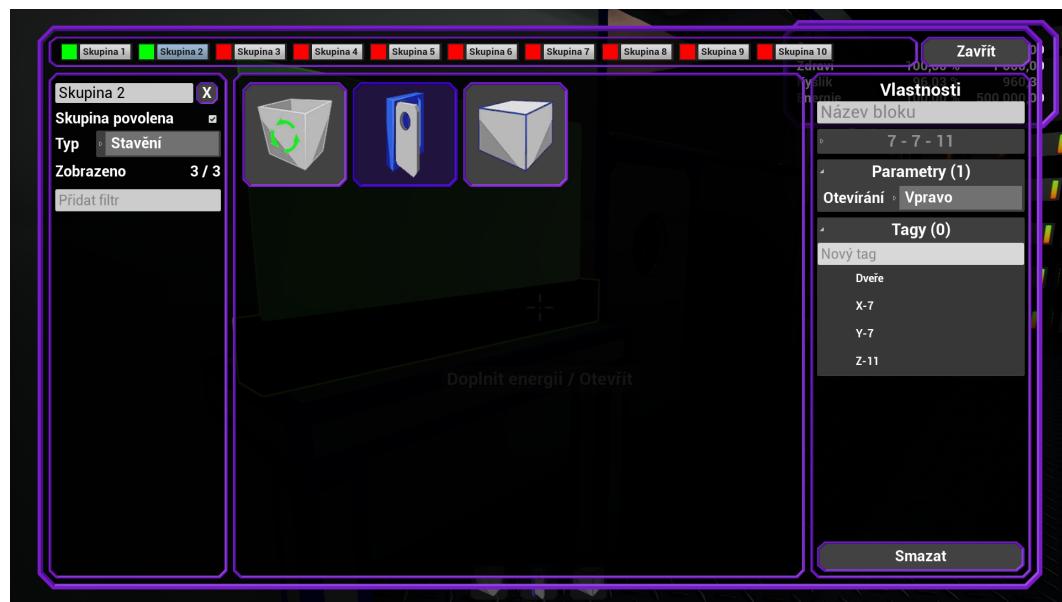
Obrázek 5.15: Inventář - Nahrát hru

Označené položky je také možné smazat z inventáře. Umístitelné položky tak mohou být reálně zničeny.



Obrázek 5.16: Inventář - Nastavení

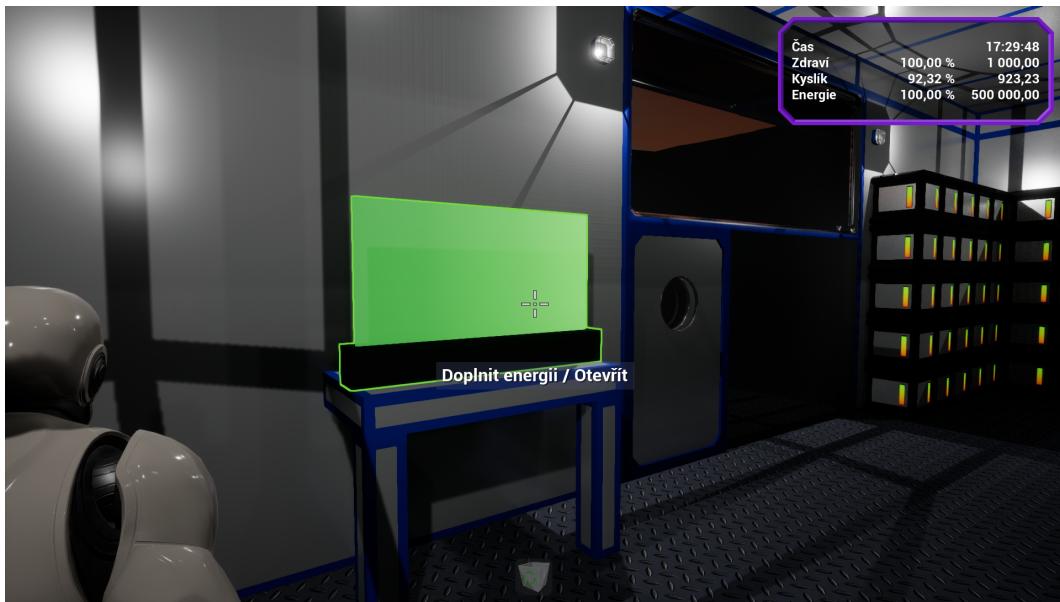
V případě, že je zvolena položka s dodatečnými parametry, je možné jejich hodnoty vidět v editoru vlastností bloku



Obrázek 5.17: Inventář - Nastavení

## 5.4 Terminál

Pro informaci o aktuálním stavu sítě je nutné použít **Terminál**. Levým tlačítkem je možné rychle doplnit energii hráče, pravým pak otevřít ovládací obrazovku.



Obrázek 5.18: Terminál - ve hře

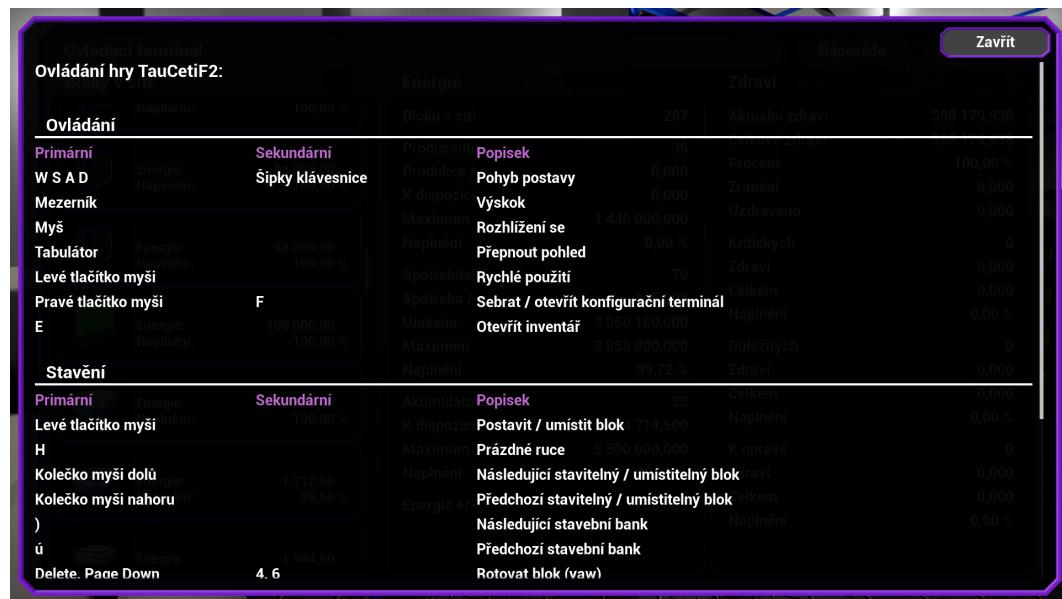
V horní části je vidět selektor obrazovky. Jeho rozkliknutím (více v obrázku 5.21) je možné zvolit výchozí obrazovku, nebo jeden z **Konstruktoru objektů**, které jsou v dispozici v síti.



Obrázek 5.19: Terminál - výchozí obrazovka

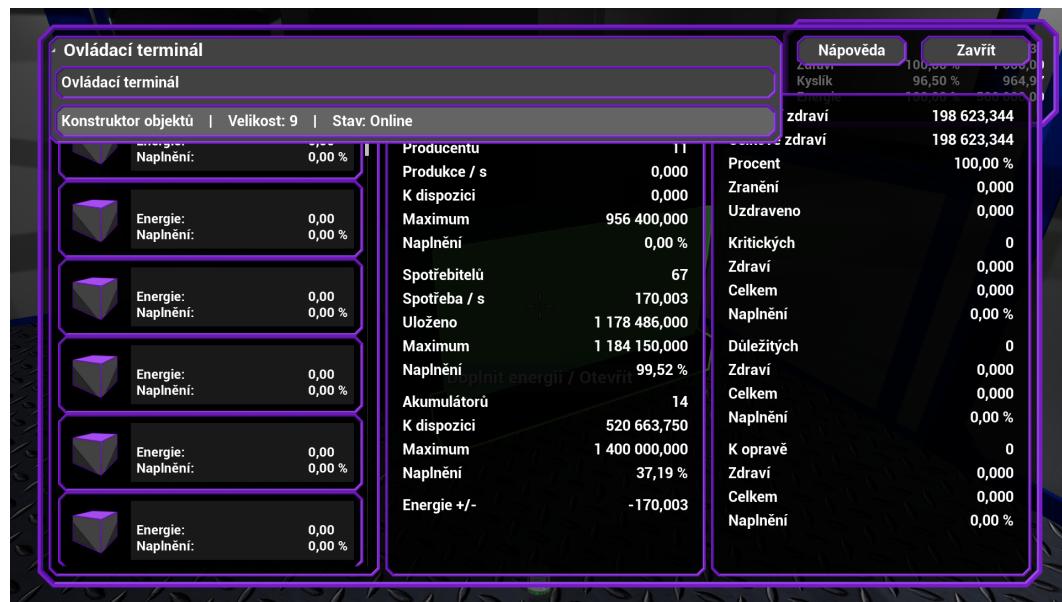
V levé části je vidět seznam význačných bloků. Uprostřed jsou vidět energetické informace o síti. Vpravo je pak možné sledovat aktuální zdravotní stav sítě a bloků v síti.

Tlačítkem **Nápověda** je možné zobrazit informace o ovládání hry a přiřazených klávesách pro jednotlivé úkony.



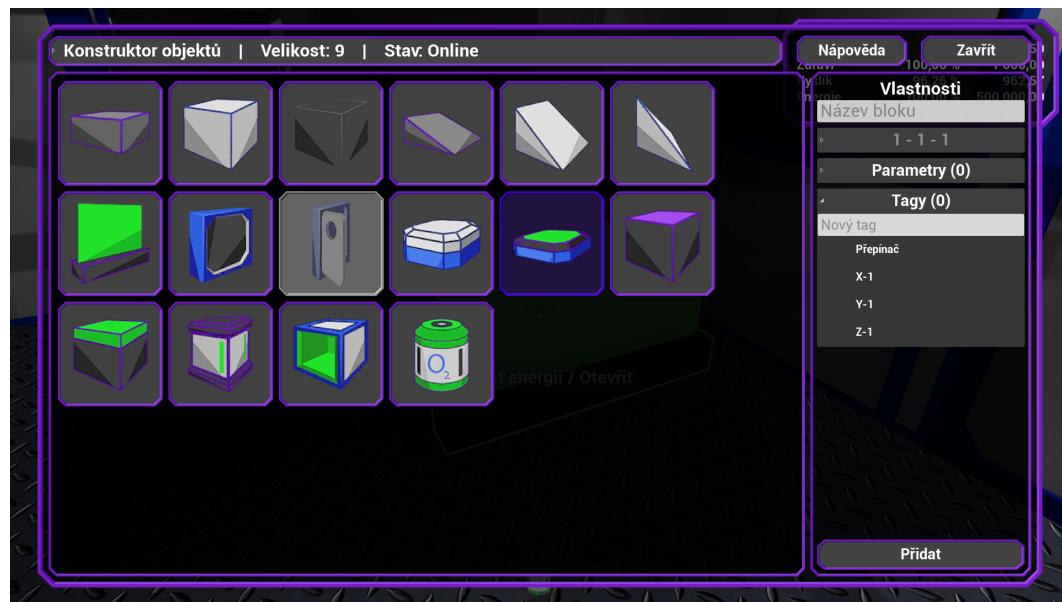
Obrázek 5.20: Terminál - nápověda

Selektor dostupných obrazovek:



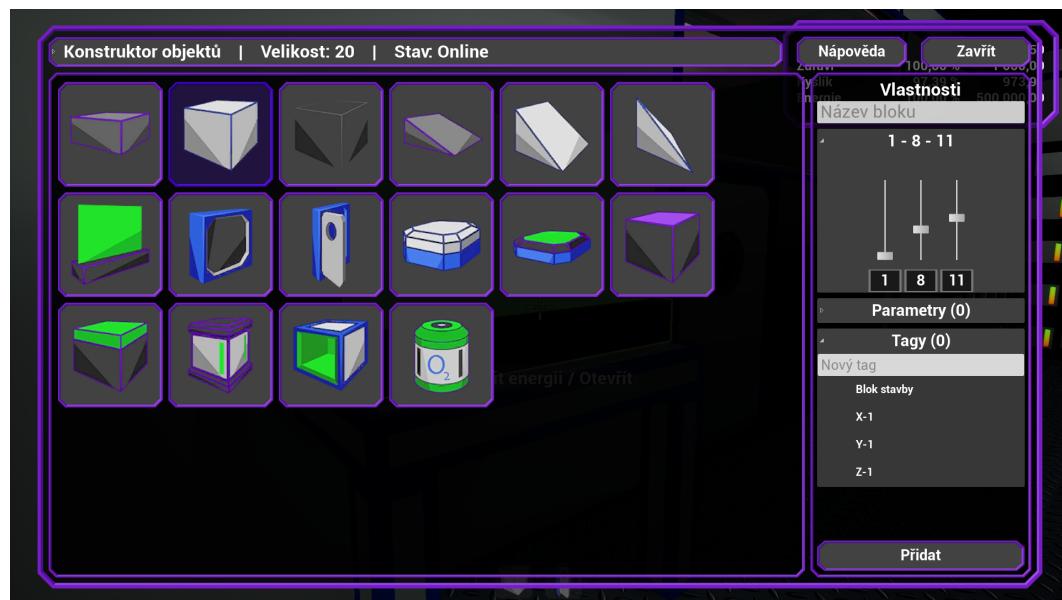
Obrázek 5.21: Terminál - selektor obrazovek

Pokud vybereme **Konstruktor objektů**, vidíme seznam dostupných bloků, které můžeme zkonstruovat. Pokud to pro některé nelze (třeba z důvodu omezení velikosti - konstruktor je na daný objekt příliš malý), blok není aktivní a nelze ho zvolit.



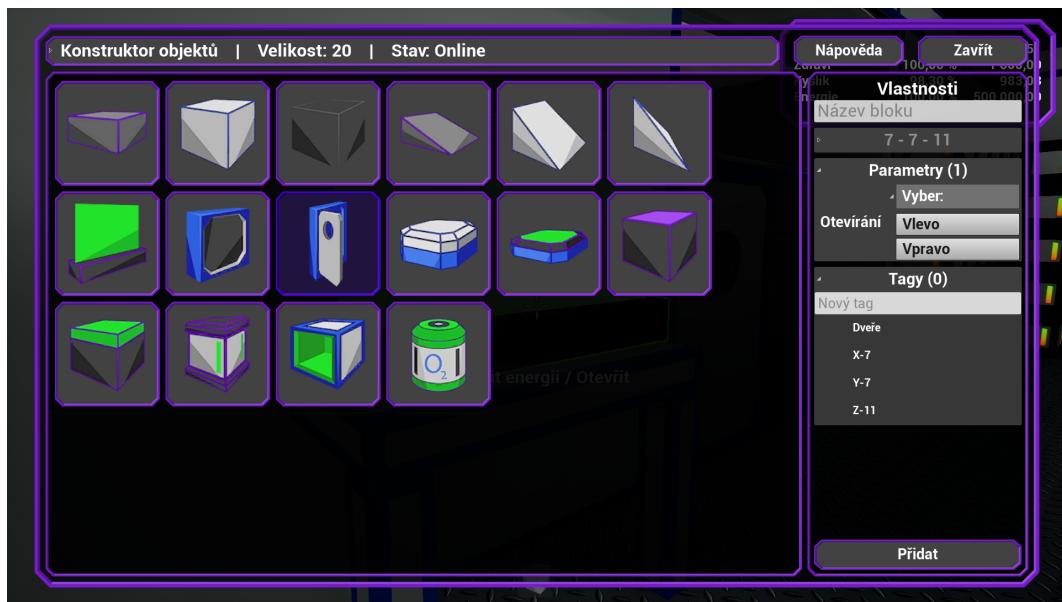
Obrázek 5.22: Terminál - konstruktor objektů

V sekci *Nastavení* je možné kromě tagů definovat i požadovanou velikost a to až do velikosti konstruktoru, nebo globálního omezení 20 násobku základní kostky.



Obrázek 5.23: Terminál - Nastavení velikosti

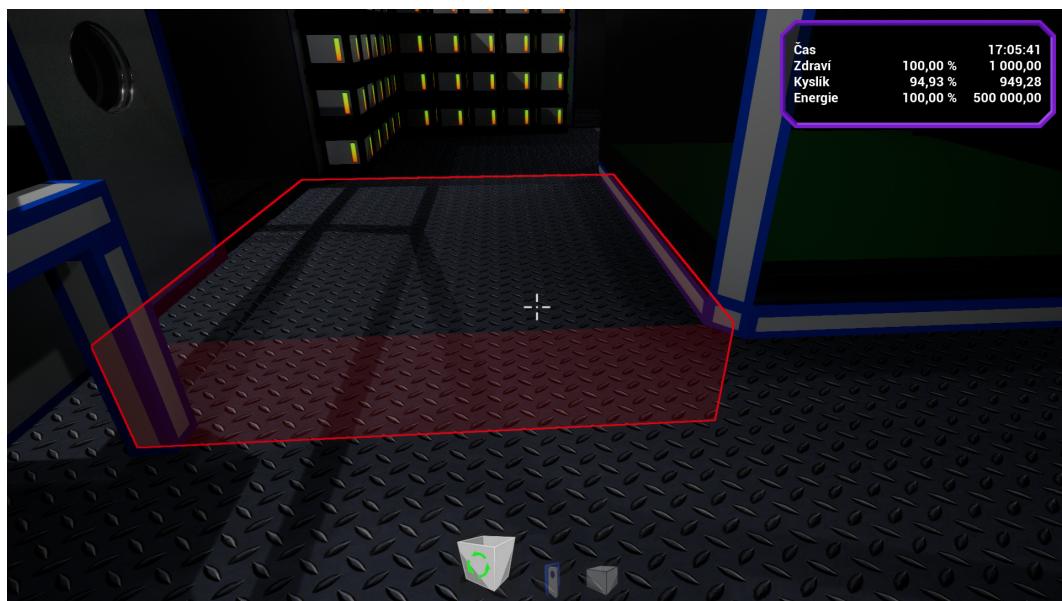
Některé bloky, třeba **Dveře** požadují dodatečné parametry, které ovlivňují jejich výsledné chování. V našem případě to je smysl otevírání dveří při čelném pohledu. Na obrázku je vidět stav po rozkliknutí.



Obrázek 5.24: Terminál - Nastavení parametrů

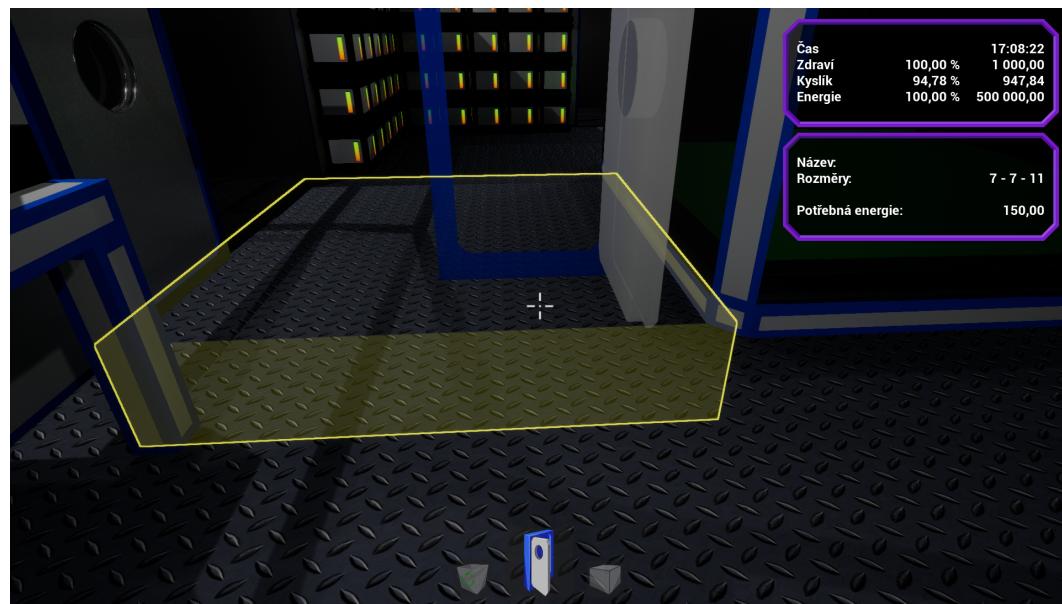
## 5.5 Stavební akce

**Destruktor** umožňuje mazat bloky. Po jeho výběru je vidět červený outline vybraného bloku.



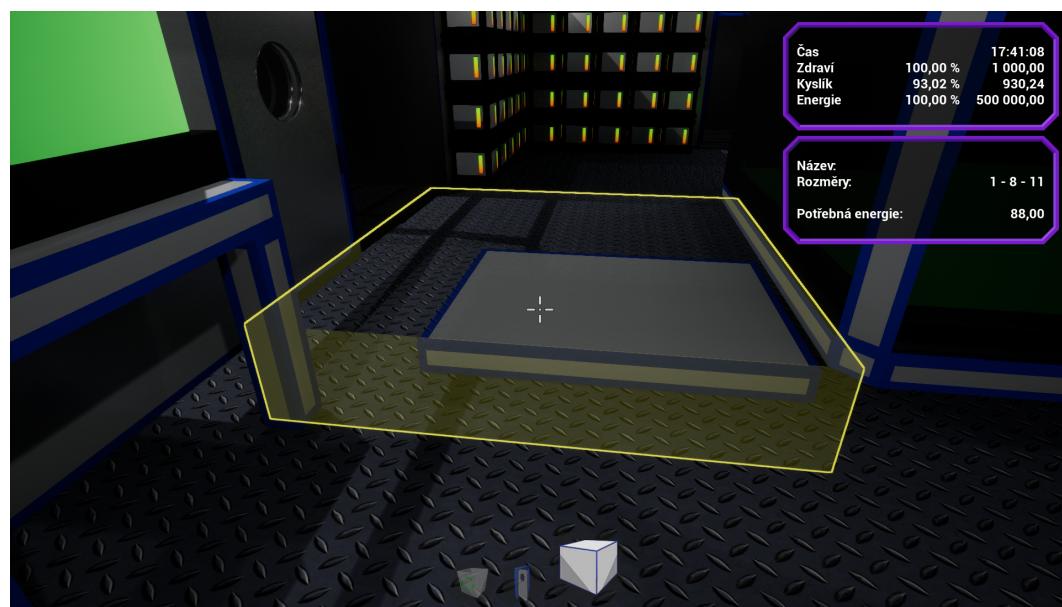
Obrázek 5.25: Stavění - smazat

Pokud vybereme umístění nového bloku, vidíme žlutě hranice sousedního bloku, ke kterému blok přistavujeme. Stavěný / umístovaný blok musí mít dostatečné místo pro svoje umístění. Dále je potřeba mít s sebou dostatečně velkou zásobu energie (dle energetické náročnosti bloku).



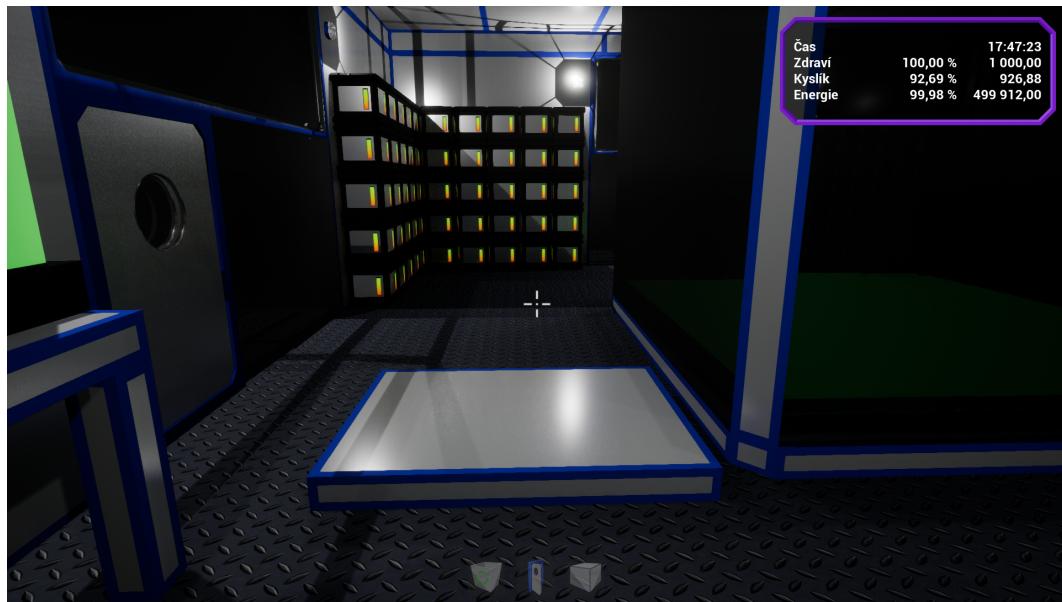
Obrázek 5.26: Stavění - umístit

Blok je též možný rotovat (Klávesy v sekci Insert .. Page Down, případně jejich ekvivalenty 7,8,9,4,5,6).



Obrázek 5.27: Stavění - rotace

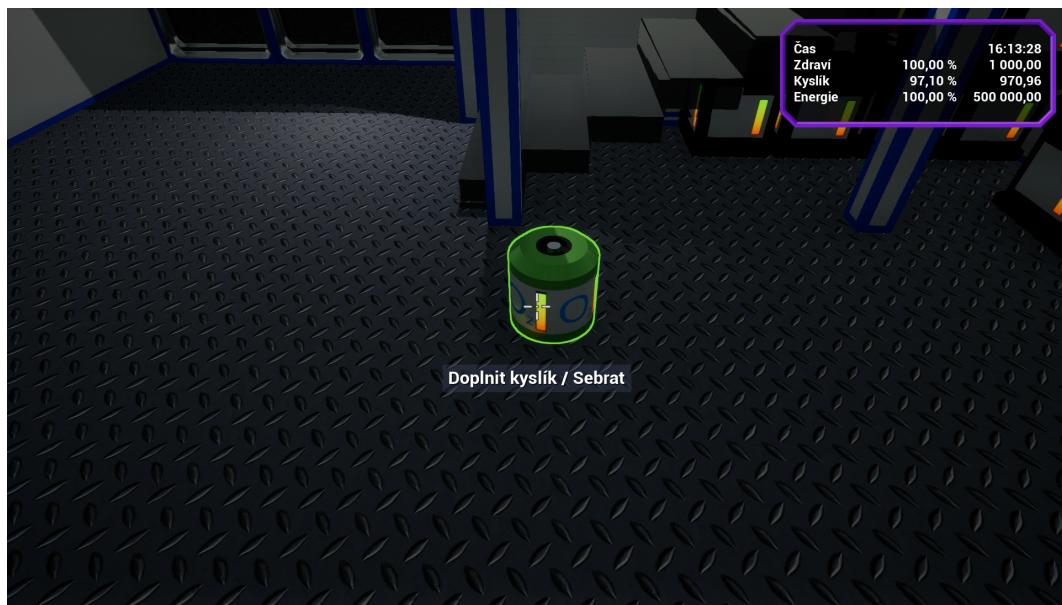
Pokud bylo umístění v pořádku, blok již není nadále průhledný a hráči ubyla energie. Pokud je zapnut kreativní mód, energie samozřejmě neubývá.



Obrázek 5.28: Stavění - po umístění

## 5.6 Umístitelné předměty

Blok **Kyslíková bomba** je možné umístit do světa a pak si ho opět vzít do inventáře. Díky tomu je možné tyto bloky dále používat třeba pro plnění v **Plničce kyslíkových bomb**. Blok je možné buď rovnou použít (levé tlačítko myši). Tento blok zároveň rovnou ukazuje, kolik objemu je využito.



Obrázek 5.29: Umístitelné předměty - plná kyslíková bomba

Na dalším obrázku vidíme již částečně použitou kyslíkovou bombu. Pokud ji pravým tlačítkem myši sebereme a otevřeme si inventář a správnou skupinu (**Typ: Inventář** v nastavení skupiny), uvidíme tento blok v seznamu.

Požité tagy se při přidání do světa a opětovném sebrání zachovávají.



Obrázek 5.30: Umístitelné předměty - použitá bomba

Zároveň v inventáři můžeme vidět přesnou hodnotu naplnění bloku.



Obrázek 5.31: Umístitelné předměty - inventář

## 5.7 Plnička kyslíkových bomb

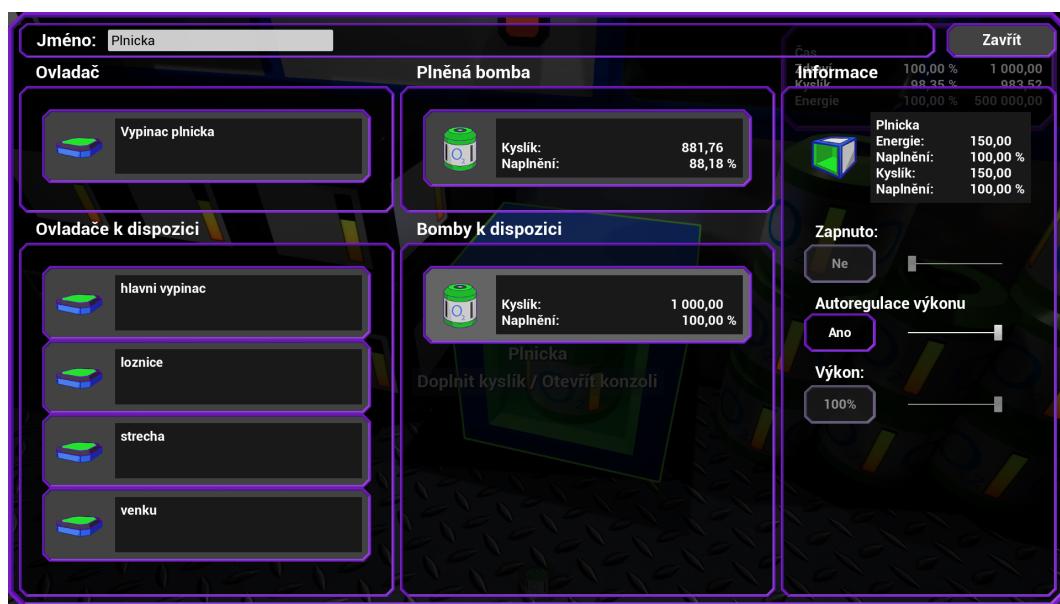
Kyslíkové bomby je potřeba opětovně naplnit, pokud byl jejich obsah spotřebován. Stejně jako u Kyslíkové bomby, levým tlačítkem myši je možné rovnou doplnit zásoby kyslíku hráče. Pravým se pak otevře ovládací obrazovka.



Obrázek 5.32: Plnička kyslíkové bomby - náhled

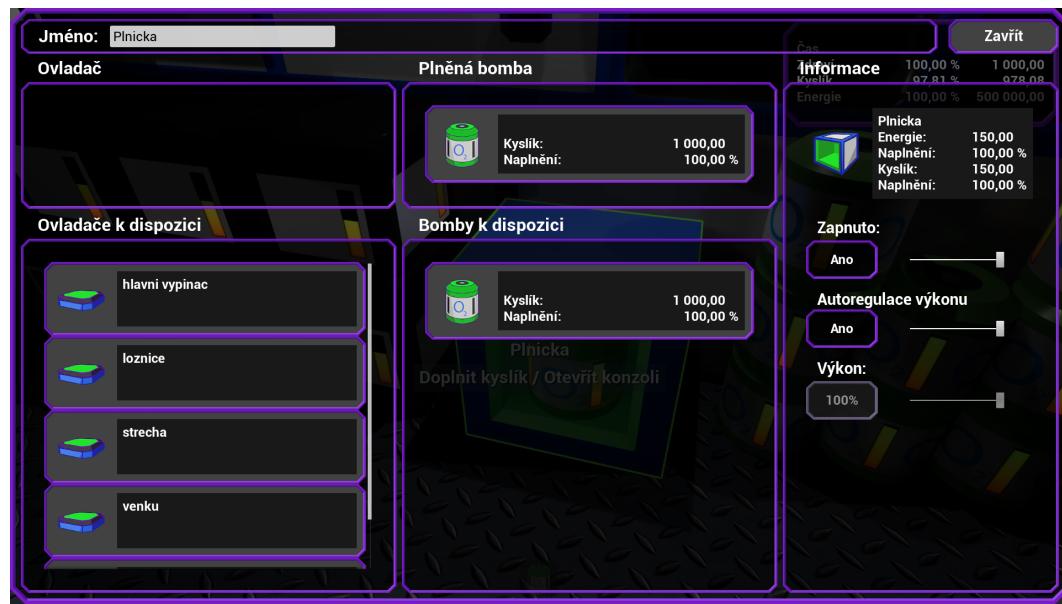
Ta má v levé části přiřazený ovladač, případně je možné nejvýše jeden ovladač ze seznamu ovladačů přiřadit. Pokud je přiřazen ovladač, zapnutí bloku se řídí jeho nastavením. V pravé části je možné regulovat spotřebovanou energii.

Uprostřed je možné vybrat bomby, které jsou v inventáři hráče, k naplnění.



Obrázek 5.33: Plnička kyslíkové bomby - ovládací obrazovka

Pokud je plnička zapnuta, generuje kyslík a ze své zásoby plní přiřazenou kyslíkovou bombu.



Obrázek 5.34: Plnička kyslíkové bomby - naplněno

## 5.8 Přepínač

Přepínač slouží jako ovládání pro světla a plničku



Obrázek 5.35: Přepínač - den

Blok umožňuje reagovat na denní dobu - automatické přepínání na definovaný stav, pokud začne den, nebo začne noc.

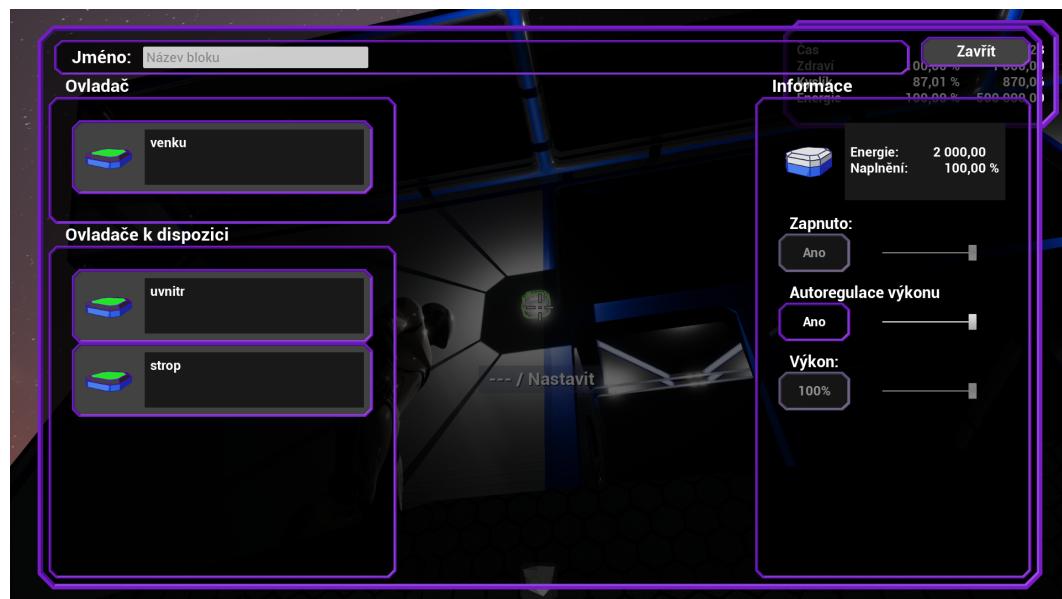
V levé části le možné přiřazovat ovládané bloky



Obrázek 5.36: Přepínač - poledne, zataženo

## 5.9 Světlo

Světlo má podobné rozhraní jako Plnička. V levé části se přiřazuje ovladač, v pravé se edituje výkon bloku.



Obrázek 5.37: Světlo - ovládací obrazovka

## 5.10 Kyselý déšť

Pokud se blíží bouře kyselého deště, nebo právě jedna probíhá, hráč vidí v levé horní části obrazovky zprávu s odhadovaným časem a intenzitou.

To umožňuje strategicky řídit chod svých budov a případně limitovat spotřebovávané zdroje v případě očekávaných dlouhotrvajících bouří. Odhadovaný čas je udán v herním čase.



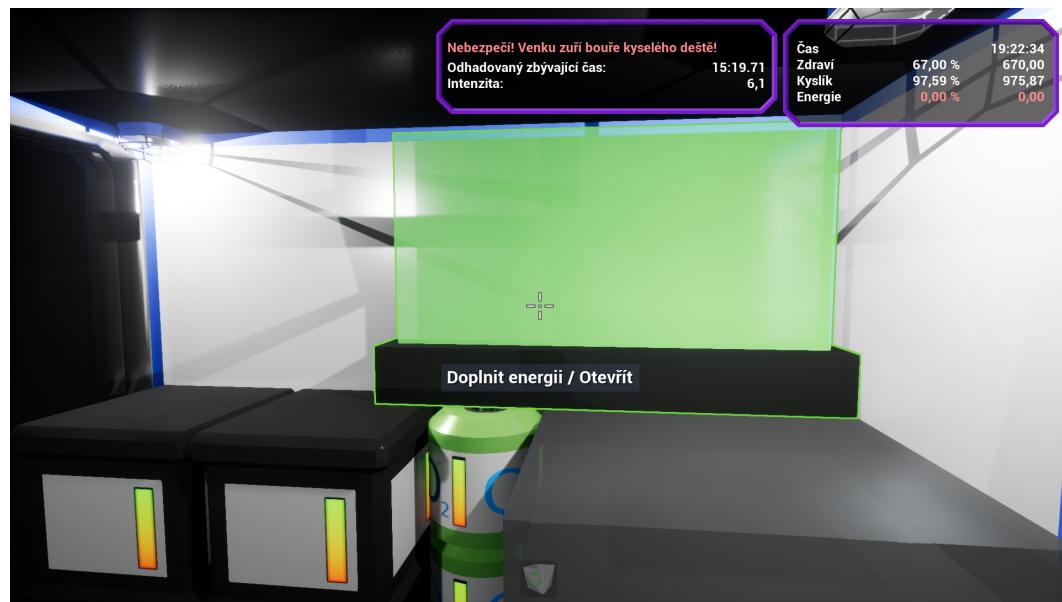
Obrázek 5.38: Kyselý déšť - info

Pokud hráč není ukrytý v budově či pod nějakým blokem, dostává zásahy. Dokud má dostatek energie, je schopen odolávat účinkům bouře, v momentě, kdy mu energie dojde, začne mu ubývat zdraví.

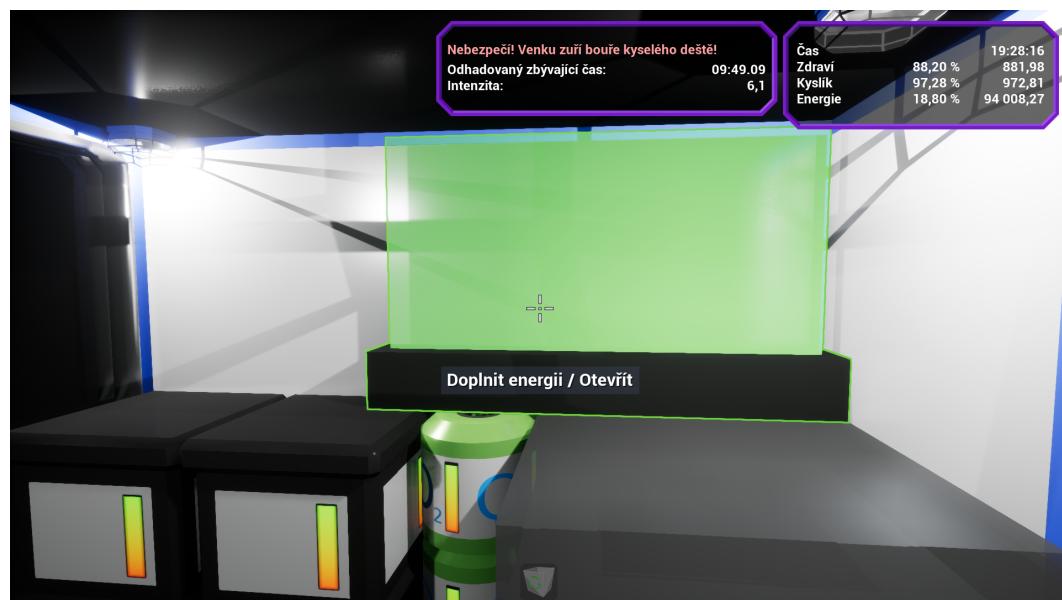


Obrázek 5.39: Kyselý déšť - zásahy

Pokud si hráč doplní energii, začne se mu zdraví obnovovat.

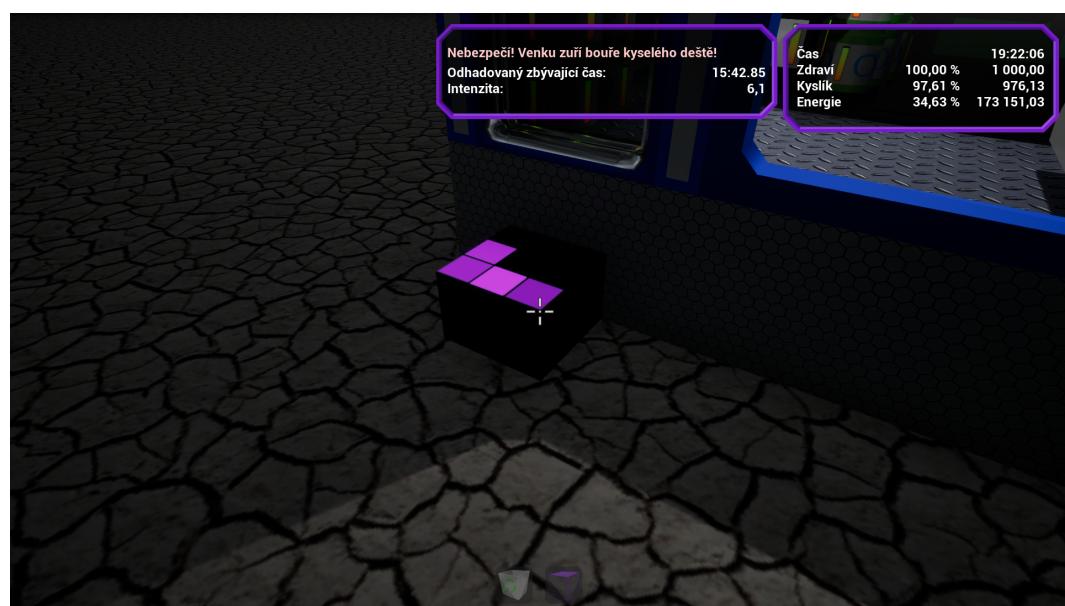


Obrázek 5.40: Kyselý děšť - obnova zdraví



Obrázek 5.41: Kyselý děšť - obnova zdraví

Během bouře je též možné pozorovat animaci generátoru energie, kdy je po každém zásahu rozsvícen příslušný čtverec. Tuto animaci lze z menu vypnout a pokud má uživatel slabší stroj, tak to i doporučujeme.



Obrázek 5.42: Kyselý déšť - animace zásahů

# **6. Závěr**

## **6.1 Zhodnocení práce**

## **6.2 Budoucí práce**

- dynamičtějí 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] Inc Epic Games. Gameplay modules. URL <https://docs.unrealengine.com/latest/INT/Programming/Modules/Gameplay/>.

# Seznam obrázků

4.1	Diagram závislostí modulů projektu.	15
5.1	Obrazovka hlavního menu - den	19
5.2	Obrazovka hlavního menu - poledne, zataženo	19
5.3	Obrazovka hlavního menu - noc	20
5.4	Obrazovka hlavního menu - Nová hra	20
5.5	Obrazovka hlavního menu - Nahrát hru	21
5.6	Obrazovka hlavního menu - Nastavení	21
5.7	Ukládání - Nový save	22
5.8	Ukládání - po uložení	22
5.9	Ukládání - nahrát hru	23
5.10	Ukládání - potvrzení nahrání hry	23
5.11	Inventář - přehled	24
5.12	Inventář - přidání skupiny	25
5.13	Inventář - výsledek s filtrovanými položkami	25
5.14	Inventář - Nová hra	26
5.15	Inventář - Nahrát hru	26
5.16	Inventář - Nastavení	27
5.17	Inventář - Nastavení	27
5.18	Terminál - ve hře	28
5.19	Terminál - výchozí obrazovka	28
5.20	Terminál - nápověda	29
5.21	Terminál - selektor obrazovek	29
5.22	Terminál - konstruktor objektů	30
5.23	Terminál - Nastavení velikosti	30
5.24	Terminál - Nastavení parametrů	31
5.25	Stavění - smazat	31
5.26	Stavění - umístit	32
5.27	Stavění - rotace	32
5.28	Stavění - po umístění	33
5.29	Umístitelné předměty - plná kyslíková bomba	33
5.30	Umístitelné předměty - použitá bomba	34
5.31	Umístitelné předměty - inventář	34
5.32	Plnička kyslíkové bomby - náhled	35
5.33	Plnička kyslíkové bomby - ovládací obrazovka	35
5.34	Plnička kyslíkové bomby - naplněno	36
5.35	Přepínač - den	36
5.36	Přepínač - poledne, zataženo	37
5.37	Světlo - ovládací obrazovka	37
5.38	Kyselý déšť - info	38
5.39	Kyselý déšť - zásahy	38
5.40	Kyselý déšť - obnova zdraví	39
5.41	Kyselý déšť - obnova zdraví	39
5.42	Kyselý déšť - animace zásahů	40

# Seznam použitých zkratek

**HUD** Head-up display

**NPC** Non-playable character

**UBT** Unreal Build Tool

**UE** Unreal Engine

# Přílohy

## A Minimální HW konfigurace

Doporučená minimální sestava (naní 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)

## B Požadavky na zprovoznění hry

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).

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 abbreviation) 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