TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Sociální aplikace pro komunitní server s 3D interaktivním rozhraním

Bakalářská práce

Studijní program:	B2646 – Informační technologie
Studijní obor:	1802T007 – Informační technologie
Autor práce:	Petr Macek
Vedoucí práce:	Ing. Roman Špánek, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr Macek**
Osobní číslo:	**M13000118**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Sociální aplikace pro komunitní server s 3D interaktivním rozhraním**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s prostředím Unity pro tvorbu interaktivního 3D rozhraní.
2. Navrhněte vhodný systém pravidel pro sociální aplikaci, která bude kombinovat reálné a virtuální prostředí.
3. Připravte řešení, které umožní interakci jednolitvých uživatelů přímo v navrženém interaktivním 3D prostředí.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Will Goldstone: Unity 3.x Game Development Essentials, Packt Publishing; 2 edition (December 20, 2011), ISBN-10: 1849691444**

[2] **Robert Nystrom: Game Programming Patterns, Publisher: Genever Benning; 1 edition (November 2, 2014), ISBN-10: 0990582906, ISBN-13: 978-0990582908**

[3] **Joe Hocking: Unity in Action: Multiplatform Game Development in C# with Unity 5, Publisher: Manning Publications; 1 edition (June 29, 2015), ISBN-10: 161729232X, ISBN-13: 978-1617292323**

Vedoucí bakalářské práce: **Ing. Roman Špánek, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2016**

Termín odevzdání bakalářské práce: **15. května 2017**

Technická univerzita v Liberci - 14 -

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2016

Prohlášení

Byl(a) jsem seznámen(a) s tím, 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 (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

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

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15. 5. 2017

Podpis:

Poděkování

Rád bych poděkoval vedoucímu své bakalářské práce, panu Ing. Roman Špánek, Ph.D., za jeho ochotu, rady, vstřícnost a trpělivost.

Děkuji Vám.

Abstrakt

Cílem této bakalářské práce je vytvořit aplikaci pro komunitní server, která bude kombinovat reálné a virtuální prostředí. K vývoji aplikace jsem použil Unity3D a SharpDevelop. Vytvořené řešení poskytuje pohyb více hráčů ve virtuálním prostředí, vzájemnou interakci uživatelů mezi sebou případně s herními prvky. Výsledkem práce je funkční a otestovaná aplikace v beta verzi.

Klíčová slova:

Aplikace, reálné a virtuální prostředí, komunitní server, Unity3D, SharpDevelop

Abstract

The aim of this bachelor thesis is to create a community server application that will combine real and virtual environments. I used Unity3D and SharpDevelop to develop the application. The created solution provides multiplayer movement in virtual environment, interaction of users with each other or with game elements. The result of this work is a functional and tested beta application.

Key words:

Application, real and virtual environment, community server, Unity3D, SharpDevelop

Obsah

Seznam ilustrací

1 Úvod

Předmětem této bakalářské práce, je návrh a vytvoření aplikace pro komunitní server. Dalšími cíli byly návrh a následné sestavení aplikace pro komunitní server, návrh a implementace pravidel pro kombinování reálného a virtuálního prostředí. V neposlední řadě šlo o ověření znalostí jazyka C#, který je použit při vývoji.

Hlavní myšlenkou této bakalářské práce je vytvoření „hry“, která by pro svou funkci využívala dvou různých platforem. První část je aplikace pro mobilní platformu Android a druhá, která je náplní této práce, je vývoj aplikace pro platformu Windows. Další náplní této práce, je práce s SQL databází, která je použita ke komunikaci mezi oběma platformami.

Jak již bylo zmíněno, první část práce, by měla být předmětem práce mého kolegy a je vyvíjena pro systém Andriod. Klíče, které jsou potřebné k odemykání dveří, bludiště se nachází v reálném světě. To znamená, že GPS pozice jednotlivých klíčů jsou uloženy na serveru, odkud jsou staženy do mobilního zařízení a vykresleny do mapy, která se zobrazí po spuštění aplikace. Uživatel tedy vidí kde a přibližně v jaké vzdálenosti od něj se nacházejí a vše co k jejich sebrání musí udělat, je dojít na dané místo a stisknout tlačítko pro sebrání.

Vývoj aplikace pro Windows a databázové části je předmětem mé bakalářské práce. Funguje na poměrně jednoduchém principu. Po spuštění se ocitnete v „labyrintu“ a vaším úkolem je jím projít a dostat se na konec. Překážou v tomto labyrintu jsou zamčené dveře a různé typy hádanek, skákání a nutnosti spolupracovat k úspěšnému nalezení daných dveří. K odemčení každých dveří je potřeba klíč, který lze získat pouze přes mobilní aplikaci. To znamená, že musíte odejít od počítače, abyste se mohli dále posunout v labyrintu.

2 Návrh řešení

2.1 Unity

Unity[1][2][3] je meziplatformní herní engine od společnosti Unity technologies. Je používán pro vývoj aplikací pro počítače, mobily, konzole a webové stránky. Unity je nyní ve dvou verzích, první verze je zcela zdarma a je určena k nekomerčnímu použití, je ale hodně omezena a neposkytuje tolik prostředků pro vývoj jako verze placená. Druhá verze nabízí věci navíc mimo jiné týmové licence, zprávy o herním výkonu aplikace a přístup ke zdrojovému kódu. Unity nabízí možnost programování v příjemném grafickém prostředí, které je přehledné, dobře se v něm orientuje. Dovoluje programovat animace, vytvářet zvuky, objekty, osvětlení a spoustu dalších.

2.2 Peer-to-peer

Peer-to-peer(P2P) nebo také klient-klient, je typ počítačových sítí, kde spolu komunikují uživatelé napřímo. Opakem tohoto typu je klient-server, kde jde veškerá komunikace přes centrální server. Dnes se P2P označení vztahuje hlavně na sítě, pomocí kterých si více uživatelů může vyměňovat data.

Největší výhodou P2P sítí je, že s rostoucím počtem klientů se zvyšuje přenosová kapacita. U typu klient-server je to právě naopak, jelikož všichni uživatelé se musejí dělit o kapacitu serveru, která se nemění.

Nevýhodou P2P sítí je, že je lze snadno napadnout.

- Spam
- Neužiteční uživatelé: pouze získávají data, ale žádná neposkytují
- DoS útoky a další

2.3 uLink

ULink[4] je síťová knihovna navržena pro herní engine Unity. ULink API je navrženo tak, aby bylo velmi podobné síťové knihovně, která je přímo v Unity. ULink plně nahradí tuto síťovou knihovnu v Unity a přidá nové vlastnosti.. Jediné co je potřeba, tak je postavit a vydat klienty a servery v Unity editoru a uLinku.

Hlavní výhodou této knihovny je, že poskytuje rychlou a jednoduchou možnost vývojářům k vytvoření síťové funkčnosti. Další výhodou je vysoká výkonost a rozsáhlá dokumentace.

2.4 Unity networking

Unity Networking[3] je knihovna pro vytváření síťových her přímo od vývojářů Unity. Tato knihovna nabízí dvě možnosti vývoje aplikací. První možností je High Level API (HLAPI) a druhou možností je NetworkTransport API. HLAPI je určeno k vytváření multiplayerových her a je určeno spíše začátečníkům nebo méně zkušeným vývojářům, díky snadnému používání a jednoduchému vývoji. NetworkTransport API je určeno pro síťovou infrastrukturu nebo pokročilé multiplayerové hry.

Výhody a nevýhody Unity networking jsou:

- Úplná integrovanost do vývojového prostředí Unity, která umožňuje snadnou práci se všemi komponenty, které knihovna nabízí.
- Srozumitelná a přehledná dokumentace, popsaná metodou krok za krokem.
- Nevýhodou jsou problémy a komplikace spojené se synchronizací a komunikací mezi více instancemi v projektu, které běží na více zařízeních, které mohou být kdekoliv ve světě.

2.5 Photon Unity Networking

Photon unity networking(PUN)[5] je balíček od společnosti Unity pro multiplayer hry, který poskytuje možnosti ověření, matchmakingu a rychlé komunikace. Aktuálně je PUN ve dvou verzích, první verze je zdarma, ale má omezení. Maximální počet hráčů je omezen na 20 online a je nutné používat verzi Unity 4 PRO (placená verze unity) nebo verzi Unity 5. Druhou verzi je nutné zakoupit, ale poskytuje podporu až pro 100 online hráčů a možnost programovat v Unity 4, bez nutnosti placené verze.

Photon unity networking využívá jeden master server a několik game serverů, které běží ve Photon Cloudu. Master server řeší matchmaking a pamatuje si informace o herních místnostech. Game server řídí několik herních místností. Herní místnosti jsou identifikovány pomocí jedinečného čísla aplikace a verze aplikace, tím je zajištěno, že se nemůžou připojit hráči z jedné hry do druhé. Tento systém má také výhodu v dalším

vývoji aplikace při dalších verzích se jen změní číslo verze a lidé co si aplikaci ještě neaktualizovali, mohou stále hrát, aniž by je aktualizace ovlivnila.

Výhodami Photon Unity Networking jsou: vývoj aplikace pro jakoukoliv zvolenou platformu, úprava a vývoj serverové logiky v .NET, C# nebo C++, kompatibilita s TCP, UDP a HTTP, protokoly, a v neposlední řadě celosvětová podpora pro uživatele, které je dosaženo díky Photon Cloudu. Photon Cloud mají hostován ve všech hlavních světových regionech, díky čemuž se dosáhne nízké odezvy.

2.6 Reálné a virtuální prostředí

V této práci jde o kombinování reálného a virtuálního prostředí. K tomuto kombinování tato práce využívá databáze jako prostředníka, který si uchovává veškeré potřebné informace, a poté dvě aplikace, jednu mobilní, která zastupuje reálné prostředí, a druhou desktopovou, ta zastupuje virtuální prostředí.

2.6.1 Navržená pravidla

Konkrétní navržená pravidla pro kombinování reálného a virtuálního prostředí v této aplikaci jsou založena na systému předmětů. Ve virtuálním prostředí, je systém místností. K pokračování do další místnosti jsou vždy potřeba dvě věci. První z nich je nutnost přijít na způsob jak zničit zeď, která schovává dveře. Tyto způsoby jsou od základního řešení kombinačních zámků, přes spolupráci více uživatelů až po nutnosti skákání a dostání se na určité místo v místnosti. Druhou věcí, která je potřeba pro postup do další místnosti je nutnost odemknout dveře. Každé dveře mají svůj klíč, který k tomu potřebují, pokud ho dotyčný uživatel nemá, tak po pokusu otevřít dveře se pošle dotaz na server, který zajistí, že dotyčný předmět se zobrazí na mapě mobilní aplikace. Pro další postup ve virtuálním prostředí je tedy nutné, aby uživatel došel s mobilem do přibližné oblasti předmětu, kde ho po sebrání bude mít ve svém inventáři ve virtuálním prostředí a následně ho může použít k odemknutí dveří.

2.7 Sociální hry

Jako sociální hry se běžně označuje hraní online her, které umožňují nebo požadují společenskou interakci mezi hráči, na rozdíl od hraní her o samotě.

Do této kategorie mohou spadat:

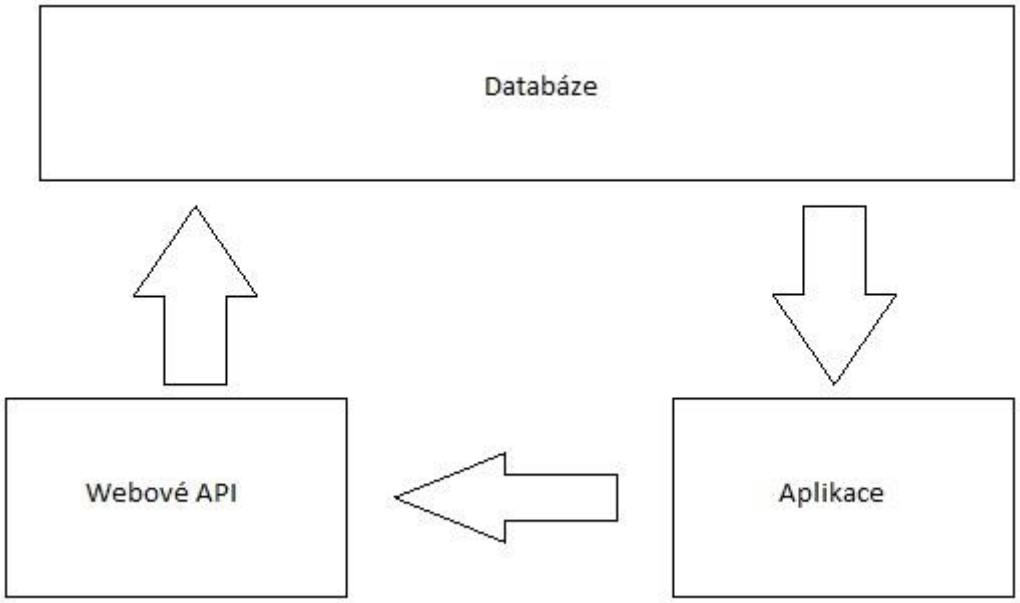
- Karetní hry (Hearthstone, Gwent)

- Deskové hry (Dáma, Šachy)
- Multiplayer hry, kde hraje více hráčů v jednou herním prostředí.
    - LAN, dočasné shromáždění lidí, které zakládají místní síť
    - MMO, označení pro hraní ohromného množství hráčů (World of Warcraft, Diablo 3)
- Hry na hrdiny (Role-playing games)

3 Implementace řešení

3.1 Databáze a webové API

Základní verze databáze a databázového API, byla převzata a rozšířena z mého ročníkového projektu a je základním kamenem k propojení reálného a virtuálního prostředí.

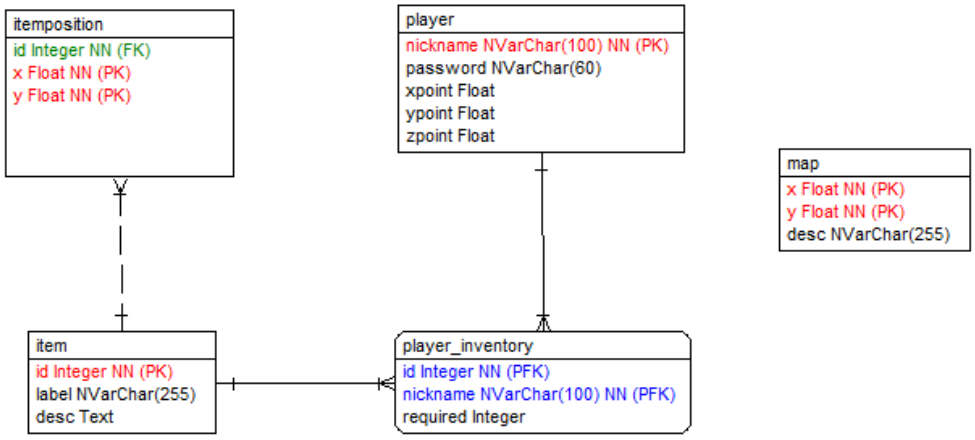


Obrázek 1: Komunikace v rámci projektu

Na obrázku je vidět způsob komunikace. Aplikace zavolá určitou metodu webového API, to tuto žádost zpracuje a následně pošle žádost databázi. Žádost může být jakákoliv, od načtení dat až po uložení dat do databáze. Následně je na základě žádost vytvořena odpověď, která se zobrazí v aplikaci.

3.1.1 Struktura databáze

Jak můžete vidět na obrázku Obrázek *2* je zde 5 tabulek a jejich vzájemné vazby. První tabulkou je ta s názvem „player“. Tato tabulka obsahuje jméno a heslo uživatele, následně obsahuje pozici „x“, „y“ a „z“, která určuje, kde se v labyrintu nachází. Každý uživatel má přidělený jeden vlastní inventář „player_inventory“. Inventáři jsou přiřazeny jednotlivé klíče z tabulky „item“ a každý z těchto klíčů má přidělenou svou pozici pomocí tabulky „itemposition“. Poslední tabulkou je „map“, která obsahuje souřadnice a popis, který je určen k budoucímu rozvoji.

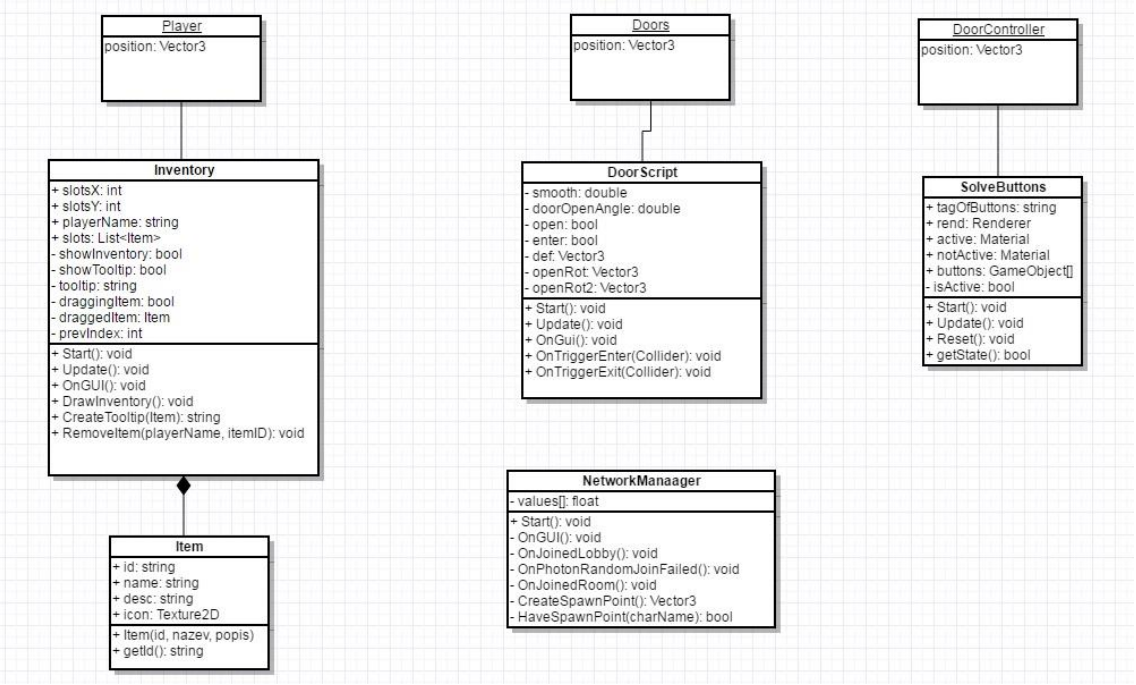


Obrázek 2: Schéma databáze

3.1.2 Webové API

Jelikož aplikace nekomunikuje se serverem přímým spojením, ale využívá k tomu http protokol, je nutné, aby na straně serveru bylo webové API, které je schopné přijímat a zpracovávat dotazy a následně na ně posílat odpovědi. Proto je na straně serveru mimo SQL databáze také server, který umožňuje běh PHP webových aplikací. V jazyce PHP je napsané celé webové API. To poskytuje metody, které jsou vybírány na základě volání s konkrétními parametry. Tyto metody následně vrací data z SQL databáze.

3.2 Aplikace pro Windows

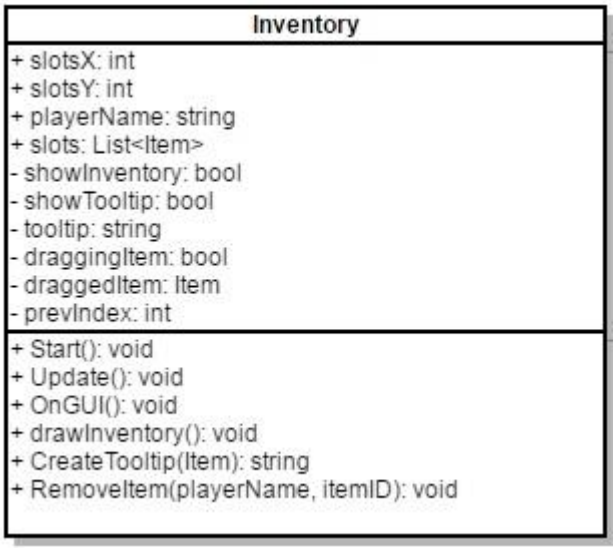


Obrázek 3: Zjednodušený class diagram

Na počátku vývoje aplikace bylo nezbytné si aplikaci navrhnout a následně rozdělit na menší části, které se lépe vytvoří a následně je pospojovat do kompletní funkční aplikace.

První částí mé aplikace bylo vytvoření základní verze labyrintu. Od vstupní místnosti, přes vytvoření dalších místností až po určení koncového místa.

Vytvořil jsem tedy základní herní plochu, kterou jsem ohraničil neprostupnými zdmi, aby se nikdo nemohl dostat mimo tuto oblast. Dále jsem pomocí zdí vytvořil místnosti a v posledním kroku jsem si vytvořil dveře. U samotných dveří bylo nutné naprogramovat jejich chování a kdy se tak mají zachovat. To znamenalo vytvoření skriptu na otevírání dveří, kde je určeno číslo klíče z databáze, následně zda je uživatel v oblasti k otevření daných dveří. Pokud je uživatel v oblasti pro otevření dveří na obrazovce se mu zobrazí hláška „Pro otevření dveří stiskni klávesu E“. Po tomto stisknutí, za předpokladu, že dotyčný vlastní klíč, se dveře otevřou. Pokud daný klíč nemá, tak pomocí API se v databázi nastaví ID klíče a že je požadován. Po sebrání pomocí mobilní aplikace se požadování nastaví na „0“ a tím pádem má uživatel klíč u sebe v inventáři. Po použití klíče se předmět odstraní z inventáře i z databáze, aby ho nemohl znovu použít.



Obrázek 4: Třída Inventory

Dalším krokem bylo vytvoření objektu zastupujícího hráče. Po několika neúspěšných pokusech o vytvoření funkčního objektu, které zahrnovaly vytvoření herního objektu a nastavení všech vlastností, které se mi nedařily ošetřit, jsem se rozhodl, že využiji knihovnu, která ulehčuje vytvoření objektu zastupujícího uživatele. Knihovna se jmenuje „Characters“ a pochází od vývojářů firmy Unity. Po naimportování této knihovny, se do projektu přidají nové skripty a herní objekty. Základním herním objektem, který jsem z tohoto balíčku využil je „Person Controller“. Tento objekt, je vytvořen pomocí kapsle a má nastaveno základní chování a přiřazenou herní kameru. Po vyzkoušení chování tohoto objektu bylo nutné upravit chování ve skriptu „Character Motor“. Pro potřeby této práce jsem snížil výšku skoku, nepatrně zvýšil rychlost pohybu a upravil gravitaci, která způsobuje rychlost padání po skoku.

Následujícím krokem bylo vytvoření inventáře podle návrhu na obrázku Obrázek 4, který si uchová informace o předmětech, které uživatel již vlastní. To znamenalo vytvoření třídy „Item“, která reprezentuje dané předměty z databáze, v tomto případě klíče. Každý klíč má své identifikační číslo, název a popis. Tyto informace se získávají z databáze pomocí webového API. Poslední vlastností pro všechny klíče je ikona. Tato ikona se načítá ze složky „Resoures“ na základě jména daného klíče. Dalším krokem bylo vytvořit inventář, do kterého se načtou hráčovi předměty z databáze. Při načítání dat z databáze je stav inventáře vrácen v XML formátu, z toho vyplývá, že byl vytvořen „XML Parser“, který se stará o správné načtení do inventáře. Tyto předměty se poté načtou a zobrazí v inventáři, kde po najetí myši se zobrazí informace o daném předmětu. Posledním krokem k dokončení inventáře, bylo nastavení klávesy k zobrazení a schování.

Ve vývojovém prostředí Unity jsem ve vstupech nastavil klávesu „i“ k proměnné „inventory“. Poté stačilo ve skriptu nastavit podmínku a poslední část inventáře je hotova.

Dalším logickým krokem bylo vytvoření systému „scén“. Jednu jsem měl již vytvořenou a to byla ta herní, která obsahovala celý labyrint. Jako další scénu jsem vytvořil úvodní obrazovku, kde se uživatel přihlašuje za pomocí databáze. Přihlašovací jméno, které má uživatel i v databázi, je zároveň jméno jeho herní postavy. Při úspěšném přihlášení se toto jméno nastaví do souboru, pro další používání v ostatních scénách. Po přihlášení se z databáze zjistí pozice „x“, „y“ a „z“, pokud jsou tyto hodnoty nastaveny, uživatel již v této hře byl a objeví se na místě, kde se naposledy odhlásil. Pokud tyto souřadnice nastaveny nejsou, objeví se hráč v předem určené oblasti. Při načítání herní scény, které je automatické pomocí načítacího skriptu, se ze souboru přečte uživatelské jméno a na jeho základě se načte inventář. Poslední scénou k vytvoření, byla ukončovací, které se uživateli zobrazila po dokončení hry, která obsahovala gratulační zprávu s adresou, kam je možné zasílat zpětnou vazbu.

3.3 Implementace pravidel

Po vytvoření základních prostředků v Unity popřípadě v C#, jsem došel k implementaci pravidel pro kombinování virtuálního a reálného prostředí.

Prvním krokem bylo vytvoření „XMLParseru“, který zajistí správné parsování dat z databáze, které vrací na základě žádosti webové API. To probíhá tak, že jsou data načtena do prázdného XML dokumentu a následně jsou data rozparsována a na jejich základě se vytvářejí objekty typu „Item“, který má: „id, název, popis a icon“. Tyto informace jsou předány třídě „Inventory“, která se stará o uchovávání, vykreslování a práci s předměty v rámci inventáře. Posledním krokem k implementování předmětů z reálného prostředí, tedy klíčů, bylo každým dveřím nastavit požadovaný klíč, na základě jeho „id“.

Následujícím krokem bylo vytvoření „rozbitné“ zdi, která blokovala dveře. Tato zeď je v každé místnosti a v každé místnosti se rozbijí jiným způsobem. V první místnosti jde o současnou aktivaci dvou platforem. V následujících místnostech jsou kombinační zámky a aktivace různých mechanizmů, které jsou následně popsány. Každý typ mechanizmu má svůj vlastní skript, který zajišťuje jeho chování a provedení.

Mechanizmus v první místnosti funguje na principu detekce herních objektů. Pokud obě platformy detekují herní objekt s označením hráč, je zeď zničena a zobrazí se dveře, které se dají následně odemknout klíčem, který dotyčný musí mít.

Dalším mechanizmem, který se v labyrintu nachází, jsou kombinační zámky, které jsou ukázány na obrázku Obrázek 5. Tyto kombinační zámky fungují na principu tlačítek. Pokud se stiskne správné tlačítko, změní se jeho barva na zelenou a zamkne se jeho ovládání. Po stisku špatného tlačítka se předchozí postup resetuje a je nutné zkusit jinou kombinaci. Po zadání úspěšné kombinace se nad tlačítky změní barva objektu, který slouží pro detekci postupu, na zelenou, a s tlačítky nejde nadále manipulovat. V druhé místnosti je mechanizmus tlačítek hned několikrát, kde je potřeba všechny aktivovat, k dosažení zničení zdi a postupu do další místnosti.

```
bool swap = true;
foreach (GameObject go in buttons){
    if (swap){
            if (!go.GetComponent<ActivateButton>().getActive()){
                swap = false;
            }
        } else {
            if (go.GetComponent<ActivateButton>().getActive()){
                Reset();
            }
        }
}
if(swap){
    isActive = true;
    rend.material = active;
}
```

Obrázek 5: Ukázka zdrojového kódu kombinačního zámku

Ve třetí místnosti je poprvé nutné spolupracovat s ostatními uživateli, alespoň s jedním. Zde se jedná o podobný systém jako v první místnosti, který je založen na aktivaci platforem. Detekce uživatele na jedné platformě aktivuje platformu tomu druhému a ten se může posunout, následně zase on aktivuje platformu tomu druhému. Společným úsilím se musejí dostat nakonec, kde za pomoci společného kombinačního zámku dokáží zničit zeď, která blokuje dveře. Platformy, které nejsou trvalé, to znamená, že se musejí aktivovat, jsou označeny modrou barvou. Rozdíl toho kombinačního zámku, oproti těm v předchozí místnosti je ten, že zde se jedná o to, že každý má přístup ke svým tlačítkům, ale kombinace se vytváří ze všech, ke kterým mají přístup. Zde je opět indikátor toho, že je kombinační zámek úspěšně prolomen a lze postupovat dále v labyrintu.

V následující místnosti je další skákání, ale s odlišnými překážkami. Nyní je cesta blokována vertikálně orientovanými deskami, které je potřeba aktivovat. Rozdíl od minulého mechanizmu je v tom, že zde je zábrana odstraněna jednou platformou ze dvou. To znamená, že je potřeba spolupracovat, alespoň dva hráči se sejdou na začátku a postupují spolu. Pokud by chtěli skočit oba najednou, tak se objeví stěna, která je zastaví a musejí znovu, ale když skočí pouze jeden, dostane se na další platformu, díky které deaktivuje následující zábranu a k tomu i tu předešlou. Druhá část je rozdělena na dvě cesty a je zde nutná křížová aktivace platforem, jako v předchozí místnosti. Ke zničení zdi a odblokování dveří se zde stačí dostat na konec skákání a chvíli zůstat stát na poslední platformě. Následně po odemčení dveří se ocitnete v poslední místnosti labyrintu.

Poslední místnost labyrintu je rozdělena na dvě části. V první části jde o to, projít menším bludištěm za pomoci spolupráce s dalším hráčem. Zde jsem napsal skript, který hráče vrátí zpět na začátek, pokud se dotyčný dotkne podlahy, mimo vymezenou cestu. Skript funguje na principu kolize s objektem. Pokud dojde ke kolizi, je ověřeno o jaký typ objektu se jedná. Následně pokud se jedná o objekt s označením hráč, je tento objekt přesunut na předem určené souřadnice. Cesty, po kterých se jde, je nutné aktivovat tím, že na aktivační platformě stojí jiný uživatel. V druhé části této místnosti je konečná platforma, na kterou je potřeba se dostat. Tato platforma je postavena vysoko ve vzduchu a je nutné aktivovat kombinační zámky, které odemykaní zobrazení dalších pomocných platforem, díky kterým se lze dostat na konečnou platformu.

Po dosažení konečné platformy, se automaticky po 3 vteřinách přepne na konečnou scénu. Tohoto je dosaženo díky skriptu, který je na konečné platformě. Při detekci uživatele spustí časovač, který po dosažení 3 vteřin, přepne na ukončovací scénu, která obsahuje gratulaci a e-mail pro případné nápady a zpětnou vazbu.

3.4 Interakce uživatelů

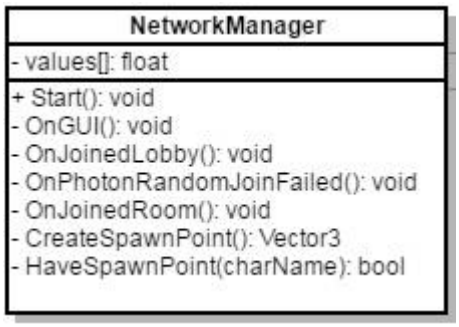
V další části mé práce šlo o dosažení vzájemné interakce uživatelů v mnou navrženém prostředí. Nejdříve jsem tento krok zkoušel vyřešit sám. Jednalo se o skript, který sledoval jednotlivé pozice herních objektů zastupující hráče a při změně některého z parametrů, se poslaly informace na server, který je poslal ostatním uživatelů. Problém nastal v tom případě, že se začalo pohybovat více uživatelů a můj notebook, na kterém běžel i server, nedokázal toto množství dat zpracovávat. Kvůli tomuto důvodu došlo k sekání se hráčů a případným pádům serveru. Po několika dalších neúspěšných pokusech v omezení intervalů posílání zpráv o změnách a omezení informací, které se posílají, jsem se rozhodl, že se porozhlédnu po řešení jinde. Našel jsem balíček, přímo pro Unity, který řeší vzájemnou interakci a synchronizaci hráčů v rámci herního prostředí. Tento balíček se nazývá Photon Unity Networking.

V první části jsem tento balíček nainstaloval a následně se mi do projektu přidaly další funkce, které tento balíček poskytuje a nabízí. Následně jsem vytvořil skript, který propojí mé virtuální prostředí s následnými servery od Photon unity netowrking. V tomto skriptu je metoda, která obsahuje základní informace k navázání spojení.

```
void Start () {
    PhotonNetwork.ConnectUsingSettings (version);
}
```

Tato metoda pouze nastaví verzi pro připojení k serveru. Toto je vše, co je potřeba k navázání spojení se serverem a vytvoření místnosti pro mé virtuální prostředí. Jelikož herní místnosti jsou vytvořeny a identifikovány za pomocí verze aplikace, kterou zde nastavuji, a názvu projektu.

Tato samotná metoda pouze nastaví informace nutné pro připojení se, ale následně bylo nezbytné vytvořit další metody, které určují, co se stane dále. Tyto metody jsou ukázány na následujícím obrázku a dále popíši, co která metoda realizuje.

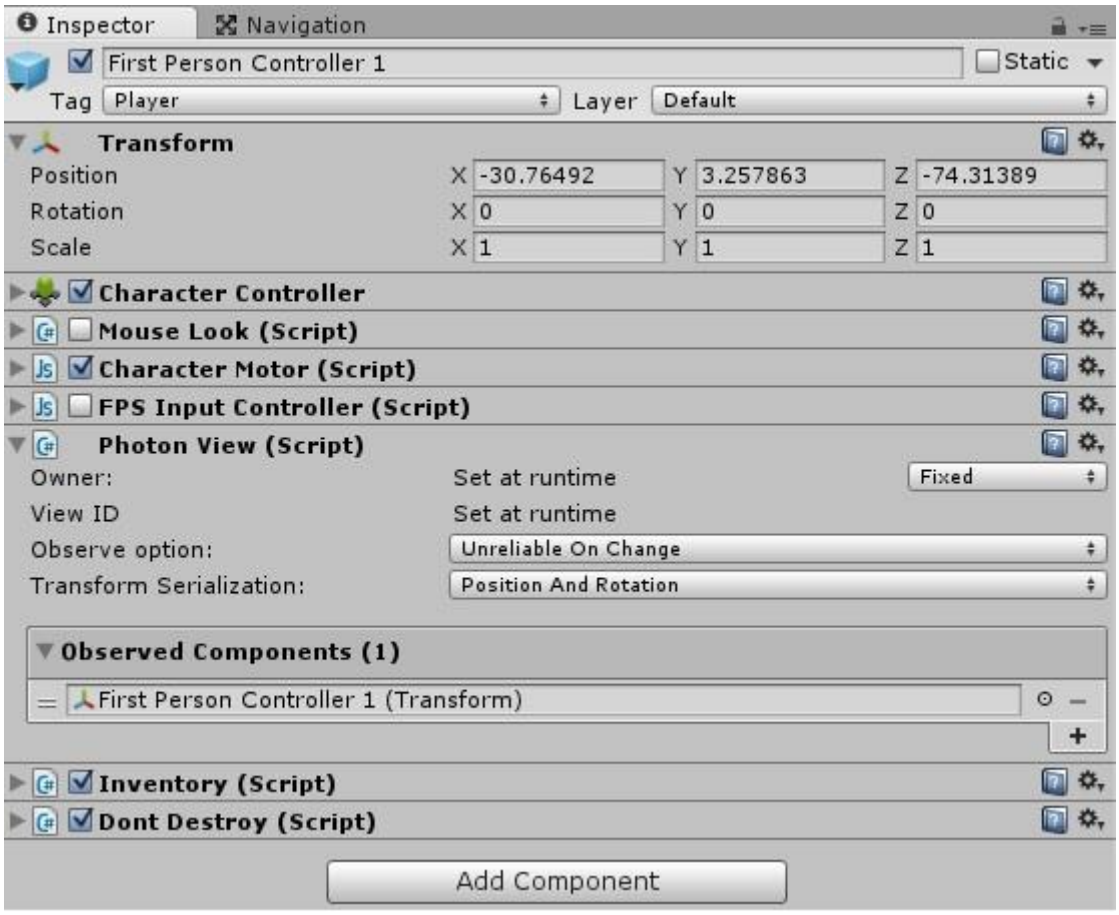


Obrázek 6: Třída NetworkManager

Metoda „Start“ je již popsána a slouží k navázání spojení se serverem. Následující metoda „OnGUI“ slouží k vypsání informací uživateli, zda je připojen, zda je v lobby, nebo že se nešlo připojit. Následující metoda „OnJoinedLobby“ identifikuje, co se má stát, pokud se uživatel připojí do hry. Po zapnutí aplikace je uživatel automaticky přesměrován do místnosti, která slouží k čekání. Zde je možné vytvořit seznam herních místností a dát uživateli možnost, aby si vybral, ve které místnosti chce hrát. Pro mou práci jsem si zvolil, že se žádný seznam místností nezobrazí, ale uživatele to zkusí automaticky připojit do některé z místností. Pokud připojení do náhodné místnosti selže, což se u prvního uživatele stane téměř vždy, jelikož zde není vytvořena žádná místnost, do které by to uživatele připojilo, zavolá se metoda „OnPhotonRandomJoinFailed“. Tato metoda má za úkol vytvořit místnost a následně tam připojit uživatele, který tuto metodu zavolal. V těle metody „OnJoinedRoom“ jsou pokyny pro vytvoření herního objektu ovládaného hráčem. V těle této metody se ověří, zda již byl uživatel přihlášen a má

nějakou uloženou pozici. Ověření proběhne na základě metody „HaveSpawnPoint“, která komunikuje s webovým API, které vrátí uloženou pozici v databázi. Pokud uživatel takovouto pozici má, nastaví se do vektoru, který se následně použije k vytvoření samotného herního objektu. Není-li žádný záznam o uložené pozici v databázi, vytvoří se náhodná pozice na základě začáteční oblasti, která je umístěna v první místnosti.

Photon Unity netoworking dále poskytuje skript, který zařizuje sledování různých objektů v rámci herního prostředí, tento skript se jmenuje „Photon View“. Tento skript se přiřadí objektu, který má sledovat a synchronizovat s ostatními hráči v labyrintu. V mém případě se tedy jedná o „First Person Controller“. Dále je nutné tomuto skriptu přiřadit, co konkrétně má sledovat a synchronizovat. Jelikož potřebuji, aby se hráči viděli, jak se pohybují, tak jsem skriptu „Photon View“ přiřadil objekt „Transform“, herního objektu zastupujícího hráče. „Transform“ je objekt, který má každý herní objekt v prostředí Unity a který si uchovává informace o pozici, rotaci a velikost daného objektu. Díky tomuto skriptu je tedy zajištěné, že se hráči navzájem uvidí, jak se v rámci virtuálního prostředí pohybují.



Obrázek 7: Finální verze controlleru pro uživatele

„Character Motor“ obsahuje nastavení pro ovládání, jako je rychlost chůze, výška skoku, rychlost stoupání atd. Dále je zde „Photon View“, který zajišťuje komunikaci a synchronizaci s ostatními objekty s tímto skriptem. Posledním skriptem, který vznikl na základě zpětné vazby a je tam i popsán je „Dont Destroy“.

4 Testování

Aplikaci bylo nutné testovat po celou dobu jejího vývoje. Nabízejí se k testování dvě alternativy. První je testování přímo v rámci vývojového prostředí Unity. Tato možnost se zezačátku zdála dostačující, ale kvůli některým modulům a scénám, bylo nutné aplikaci vygenerovat do spustitelné aplikace a otestovat mimo prostředí Unity. Hlavní výhodou možnosti testování v samotném vývojovém prostředí Unity, je možnost vidět jakékoliv změny, které proběhly, v jakém stavu jsou proměnné a jak se která věc chová v závislosti na akci.

Při vytváření spustitelné aplikace jsou zde možnosti, pro jaký operační systém má aplikace být, její případná verze a samotné pořadí scén. K testování beta verzí jsem využíval vlastní notebook, takže jsem všechny potřebná nastavení nastavil tak, aby vyhovovala mému počítači.

První větším problémem při testování aplikace bylo špatné zobrazování textur herních objektů labyrintu. Došlo k tomu při implementaci manažeru scén. Poprvé když jsem vytvořil scénu s přihlašování a následné načtení samotného labyrintu došlo k tomu, že bylo vše černé, uživatel se mohl pohybovat a používat všechny dostupné prostředky, ale problém byl v tom, že nic neviděl. Po pátrání po příčině tohoto problému jsem zjistil, že to je problém vývojového prostředí Unity. Po otestování mimo prostředí Unity, vše fungovalo přesně, jak mělo, a uživatel vše viděl.

Dalším problémem, se kterým jsem se setkal, byl problém se vzájemnou interakcí uživatelů. Tento problém jsem se snažil otestovat sám a to tím způsobem, že jsem si danou aplikaci spustil alespoň 2krát. Nastal ale problém a to ten, že se nezablokovalo ovládání neaktivního okna. Tímto jsem se dostal také k problému, že jeden hráč může mít více účtů a hrát sám se s sebou. Po zkoumání a zjišťování čím je tento problém způsoben, jsem zjistil to, že je to „problém“ v balíčku „Character“, který používám. Konkrétně je to problém u dvou skriptů, které má k sobě přiřazeny herní objekt, který zastupuje hráče. Problémové skripty jsou „Mouse Look“ a „FPS Input Controller“. První ze skriptů ovládání uživatelův pohled a druhý ovládání pohybů. Řešení tohoto problému bylo relativně jednoduché a to tak, že jako základní možnost byly tyto dva skripty vypnuty.

Následně se skripty povolí pouze, když je okno aktivní a to za pomoc kódu, který je ukázán na následujícím obrázku.

```
((MonoBehaviour)myCharacter.GetComponent ("FPSInputController")).enabled = true;
((MonoBehaviour)myCharacter.GetComponent ("MouseLook")).enabled = true;
myCharacter.transform.FindChild("Main Camera").gameObject.SetActive (true);
```

Obrázek 8: Ukázka zdrojového kódu pro povolení ovládání

Další fáze mého testování neodhalila žádné závažnější problémy, proto jsem se rozhodl, že aplikaci poskytnu několika vybraným lidem, aby pro mě aplikaci otestovali a sdělili mi své poznatky, názory a chyby, na které narazili. K tomuto testování jsem v databázi vytvořil několik účtů. Každý z těchto účtu dostal některé klíče, ale ne všechny, abych otestoval i žádosti, které se posílají, pokud daný uživatel klíč nevlastní.

Prvních několik zpráv, které mi byly zaslány, obsahovaly převážně totéž, a to zprávy ohledně grafické stránky aplikace. Jelikož nejsem přeborník na grafiku a ani to nebylo předmětem mé práce, tak jsem použil pouze základní grafické styly. Další a již závažnější chybou byly „točící se dveře“. Tento problém se dveřmi spočíval v tom, že občas při otevření dveří se jednoduše neotevřely, jak měly, ale došlo k tomu, že se začaly neustále točit. Po dlouhém pátrání a zjišťování v čem by mohl být problém, jsem nedokázal zjistit žádnou závadu, ať už to byla závada ve skriptu nebo v herním objektu. Po dalším pátrání jsem narazil na chybu, která vznikla mou příčinou. Tato chyba byla ve skriptu, který jsem pro dveře napsal. Chyba spočívala v metodě „Update“, tato metoda se volá v závislosti na FPS. Jelikož se volá neustále, tak by se dalo říci, že se jedná o cyklus a já jsem zde zapomněl dát podmínku, která určovala chování „opačných“ dveří. V labyrintu jsou dveře a všechny vypadají stejně, ale v závislosti na tom, jak jsou umístěny, se otevírají a pokud jsou někde otočeny opačným směrem, stalo se, že se začaly neustále točit.

Další chyba, která se v aplikace vyskytla, bylo občasné zničení „First person controlleru“, který hráč ovládal. Ke zničení docházelo při načítání scén, v mém případě se jednalo o načtení menu a ukončení hry. V normálním případě by skript „Dont destroy“ nebyl potřeba, ale pokud se stalo, že uživatel otevřel menu a vrátil se zpátky do hry, neměl, za koho by hrál, protože při přechodu byl tento objekt zničen.

5 Závěr

V této práci se povedlo vytvořit multimediální aplikaci za pomoci vývojového prostředí Unity. Výsledná aplikace kombinuje reálné a virtuální prostředí za pomoci navržených pravidel a umožňuje interakci více uživatelů v obou prostředích. Vzájemné interakce je dosaženo za pomoci Unity, C# skriptů, webových služeb a SQL databáze. V poslední fázi byla multimediální aplikace otestována vybranými uživateli. Při vývoji aplikace došlo k problému při implementaci více uživatelů, ale problém se podařilo vyřešit za pomoci knihovny "Photon Unity Networking".

Dále je možné aplikaci rozšířit o další pravidla pro kombinování reálného a virtuálního prostředí, rozšíření labyrintu o další místností, přidání dalších typů předmětů nebo vytvoření a implementování soubojového systému.

6 Seznam použité literatury

[1] Will Goldstone: Unity 3.x Game Development Essentials, Packt Publishng; 2 edition (December 20, 2011), ISBN-10: 1849691444

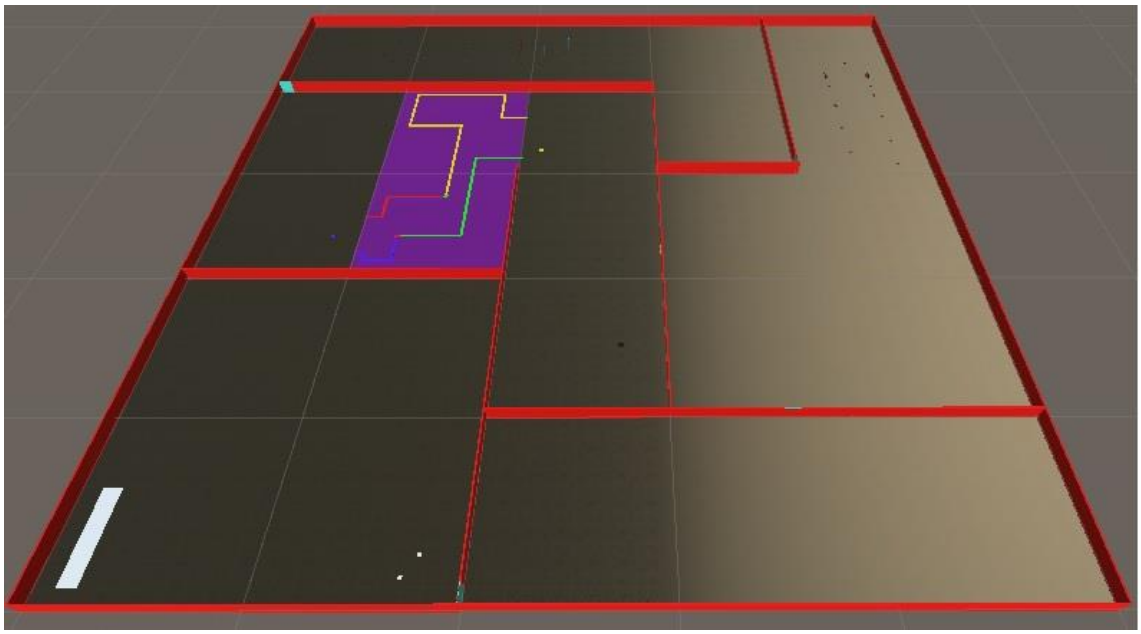
[2] Joe Hocking: Unity in Action: Multiplatform Game Development in C# with Unity 5, Publisher: Manning Publications; 1 edition (June 29, 2015), ISBN-10:161729232X, ISBN-13: 978-1617292323

[3] Unity Manual [online]. Unity Technologies, © 2017 [publikováno 5.5.2017]. Dostupné z: http://docs.unity3d.com/Manual/index.html

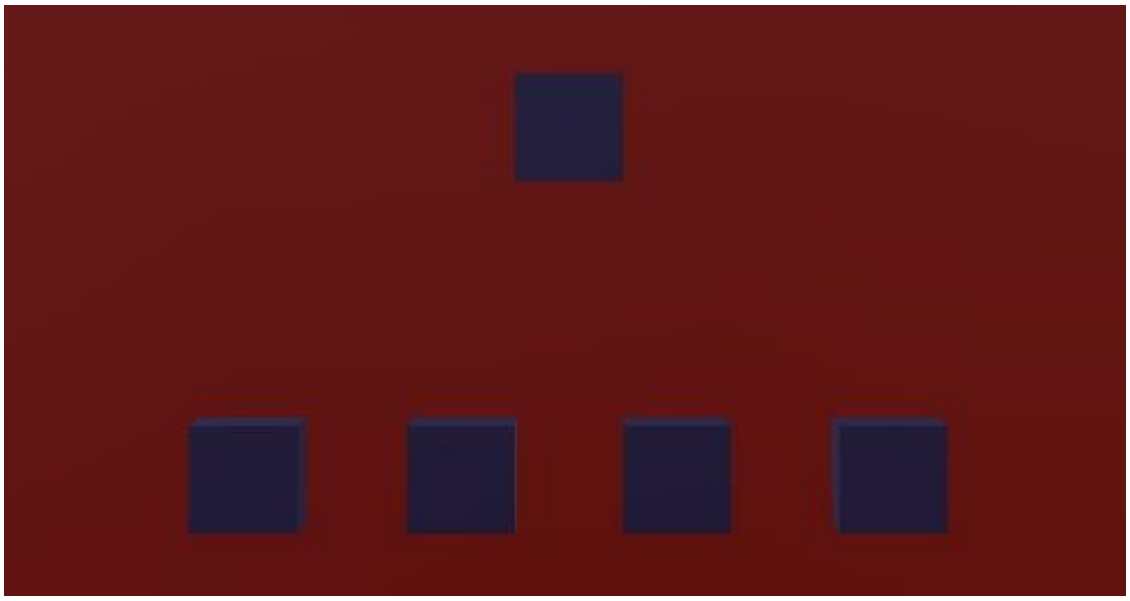
[4] ULink Manual [online]. MuchDifferent, © 2013. Dostupné z: http://developer.muchdifferent.com/unitypark/uLink/uLink

[5] Photon Unity networking dokumentace [online]. Exit Games®, ©2003-2017. Dostupné z: https://doc.photonengine.com/zh-cn/pun/current/getting-started/pun-intro

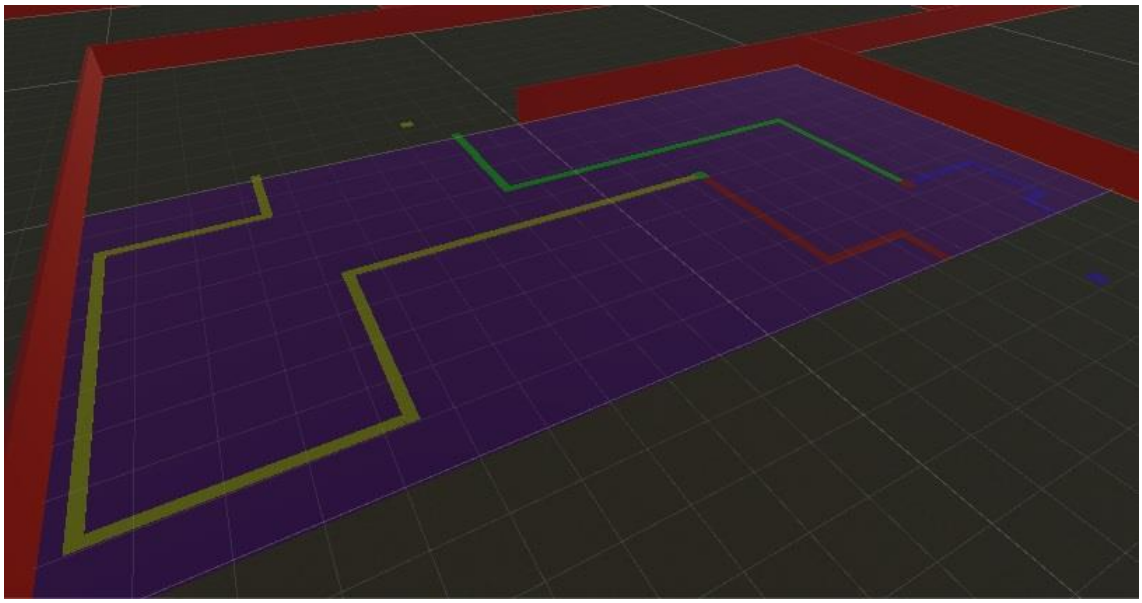
Přílohy



Obrázek 9: Ukázka celého labyrintu



Obrázek 10: Základní controller pro kombinační zámek



Obrázek 11: Labyrint v poslední místnosti