Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Herní Engine

Jiří Kotlán

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jiří Kotlán

Informatika

Název práce

Herní Engine

Název anglicky

Game Engine

Cíle práce

Diplomová práce je tematicky zaměřena na problematiku tvorby 3D prostředí v herních enginech. Hlavním cílem práce je tvorba 3D prostředí v Source engine, včetně tvorby vlastních textur s normálními mapami pro vytvoření iluze 3D povrchů, implementace vlastních modelů, generování a optimalizace reflektivních povrchů a optimalizace vykreslování vytvořeného prostředí.

Dílčí cíle práce jsou:
- Charakteristika schopností a limitací Source engine.
- Porovnání funkčností Source engine s novějšími herními enginy (Unreal, Unity, Source 2) a jeho předchůdcem (GoldSrc).
- Rešerše na způsoby generování odrazů a osvětlení těchto herních enginů.

Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Vlastní řešení je realizováno pomocí programu Hammer++ na tvorbu samotného prostředí, dále jsou využity programy Gimp a VTFedit na tvorbu vlastních textur a programy kompilátor Propper a Crowbar na tvorbu vlastních modelů včetně jejich implementace do Source engine. Na základě výsledků práce budou formulovány závěry DP.

Doporučený rozsah práce

70 – 90 stran

Klíčová slova

herní engine, model, textura, herní průmysl, cloud gaming

Doporučené zdroje informací

Bernier, B.: Source SDK Game Development Essentials, Packt Publishing Ltd., 2014, ISBN 978-1-84969-592-3

Gregory, J.: Game Engine Architecture, CRC Press, 2018, ISBN 978-1138035454

Menard, M., Wagstaff, B.: Game Development with Unity Second Edition, Cengage Learning PTR., 2014, ISBN 978-1-305-11054-0

Satheesh, P.: Unreal Engine 4 Game Development Essentials, Packt Publishing Ltd., 2016, ISBN 978-1-78439-196-6

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

Ing. Eva Kánská, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 4. 7. 2023

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 3. 11. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 18. 03. 2024

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Herní Engine" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30. 3. 2024 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Evě Kánské, Ph.D. za rady a podporu při konstruování diplomové práce.

Herní Engine

Abstrakt

Diplomová práce se zabývá herními enginy a konkrétně je práce zaměřena na problematiku tvorby assetů a herních světů. V rámci teoretické části je prováděna charakteristika schopností a limitací Source Engine, včetně způsobů vykreslování ploch, optimalizace, osvětlení a simulace odrazů.

Vlastní část práce obsahuje porovnání Source Engine s modernějšími herními enginy (Unreal, Unity, Source 2) a jeho předchůdcem (GoldSrc). Hlavní náplní vlastní práce je tvorba 3D prostředí v Source Engine, včetně tvorby vlastních textur, implementace vlastních modelů, generování reflektivních povrchů a optimalizace vykreslovaného prostředí.

Na základě zveřejnění výsledného produktu je vedena diskuze v rámci statistik hodnocení herní komunity a ekonomického odhadu tvorby.

Klíčová slova: herní engine, model, textura, herní průmysl, cloud gaming

Game Engine

Abstract

The diploma thesis is focused on game engines and specifically on the topic of asset creation and level design. The theoretical section consists of Source Engine features and limitations, including surface rendering, optimization, lighting and reflection simulation.

The main section contains comparisons of Source Engine with more modern game engines (Unreal, Unity, Source 2) and its predecessor (GoldSrc). The primary focus of the main section is the development of a 3D environment in Source Engine, including the creation of custom textures, implementation of custom models, reflective surface generation and rendering optimizations.

Based on the publication of the resulting product a discussion is led in terms of community ratings and economical estimation of the design process.

Keywords: game engine, model, texture, gaming industry, cloud gaming

Obsah

Úvod

Problematika herních enginů je velmi široké téma. Samotný herní engine se skládá z mnoha různých systémů a komponentů, které spolu synergicky spolupracují za cílem umožnění vývoje a funkcionality videoher. Komerční vývoj her je nejčastěji skupinovou záležitostí, jelikož každá z částí herních enginů vyžaduje určitou úroveň specializace.

V této práci je převážně brán zřetel na vysokoúrovňovou část problematiky zabývající se tvorbou assetů a herních světů. Tato problematika je velmi důležitou součástí herního vývoje, a to převážně ve hrách trojrozměrné perspektivy, jelikož bez herních světů zkrátka není možné hrát. Navzdory tomu, že se jedná pouze o podkategorii celkového herního enginu, je toto téma stále velmi rozsáhlé a vyžaduje znalosti a dovednosti v různých oblastech celkové problematiky herních enginů.

Pro účely této práce byla tvorba projektu popisovaného v praktické části prováděna individuálně s minimální externí pomocí, čehož bylo možno dosáhnout především díky dlouholeté praxi autora s vývojem a publikací projektů podobného charakteru.

Cíl práce a metodika

1.1 Cíl práce

Diplomová práce je tematicky zaměřena na problematiku tvorby 3D prostředí v herních enginech. Hlavním cílem práce je tvorba 3D prostředí v Source Engine, včetně tvorby vlastních textur s normálními mapami pro vytvoření iluze 3D povrchů, implementace vlastních modelů, generování a optimalizace reflektivních povrchů a optimalizace vykreslování vytvořeného prostředí.

Dílčí cíle práce jsou:

- Charakteristika schopností a limitací Source Engine.
- Porovnání funkčností Source Engine s novějšími herními enginy (Unreal, Unity, Source 2) a jeho předchůdcem (GoldSrc).
- Rešerše na způsoby generování odrazů a osvětlení těchto herních enginů.

1.2 Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Vlastní řešení je realizováno pomocí programu Hammer++ na tvorbu samotného prostředí, dále jsou využity programy Gimp a VTFedit na tvorbu vlastních textur a programy kompilátor Propper a Crowbar na tvorbu vlastních modelů, včetně jejich implementace do Source Engine. Na základě výsledků práce budou formulovány závěry DP.

Teoretická východiska

1.3 Herní engine obecně

Herní enginy jsou velmi komplexní softwarový framework umožňující tvorbu a následně i funkčnost videoher. Dříve byly herní enginy velmi specializované na konkrétní hru a hardware, ovšem v dnešní době jsou enginy plně vybaveny a umožňují vývoj širokého spektra her. Téměř všechny enginy obsahují podobnou množinu komponentů. Jedná se například o komponenty vykreslování, kolizí, fyzikální simulace, animací, zvuků, umělé inteligence a mnoho dalších. (Gregory, 2018)

1.3.1 Vykreslovací frekvence

Tvořené videohry (dále jen „hry“) jsou interaktivní a dynamické – stav herního světa se mění reakcí na chování hráče. Jedná se vlastně o interaktivní simulaci v reálném čase. Engine musí dokázat vykreslovat snímky alespoň 24 krát za sekundu pro vytvoření iluze responsivního pohybu, přičemž většina her se snaží dosáhnout alespoň 30 nebo až 60 snímků za sekundu (tyto frekvence odpovídají násobkům obnovovacích frekvencí monitorů).

Různé komponenty herních enginů mají různé nároky na frekvenci „přemýšlení“, například fyzická simulace může vyžadovat až 120 aktualizací za sekundu z důvodu stability. Komponent umělé inteligence by měl „přemýšlet“ alespoň jednou za sekundu a zvukový komponent by měl být schopen aktualizace alespoň 60 krát za sekundu, aby se předešlo zasekávání zvuku či dalším zvukových problémů. (Gregory, 2018)

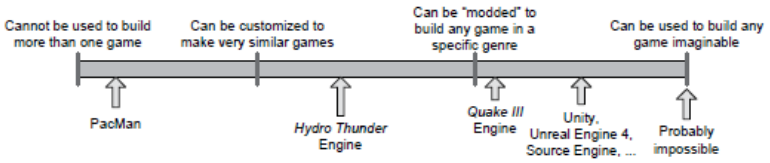
1.3.2 Měkké a tvrdé systémy reálného času

Herní enginy mohou být takzvané měkké systémy reálného času. Když v těchto enginech není dosaženo přesně určené snimkové frekvence, nemají katastrofické následky, jelikož v nich čas není přímo spojen s vykreslovací frekvencí. Na druhou stranu také existují tvrdé systémy reálného času, ve kterých buď nedosažení cílové snímkové frekvence nebo jejího přesažení může vést k vážným problémům, protože je vše vázáno a přesně načasováno na tuto konkrétní vykreslovací frekvenci. (Gregory, 2018)

1.3.3 Recyklovatelnost

Rozdělení herních enginů na jednotlivé komponenty se ukázalo býti velmi užitečným pro použití jednoho enginu na široké množství rozdílných her bez nutnosti velkých změn celého enginu. Díky této modularitě byla zrozena takzvaná moderská komunita. V rámci těchto komunit dokáží jednotlivci a malá vývojová studia vytvořit nové hry (nebo modifikace již existujících her) použitím existujících nástrojů pro práci s enginem poskytnutých původními vývojáři.

Koncem devadesátých let se začaly objevovat enginy vytvořené za cílem snadné modifikace pomocí skriptovacích jazyků a enginové licencování se stalo realistickým zdrojem příjmů pro vývojáře těchto enginů. (Gregory, 2018)



Obrázek 1 – Míra „recyklovatelnosti“ herních enginů (Gregory, 2018)

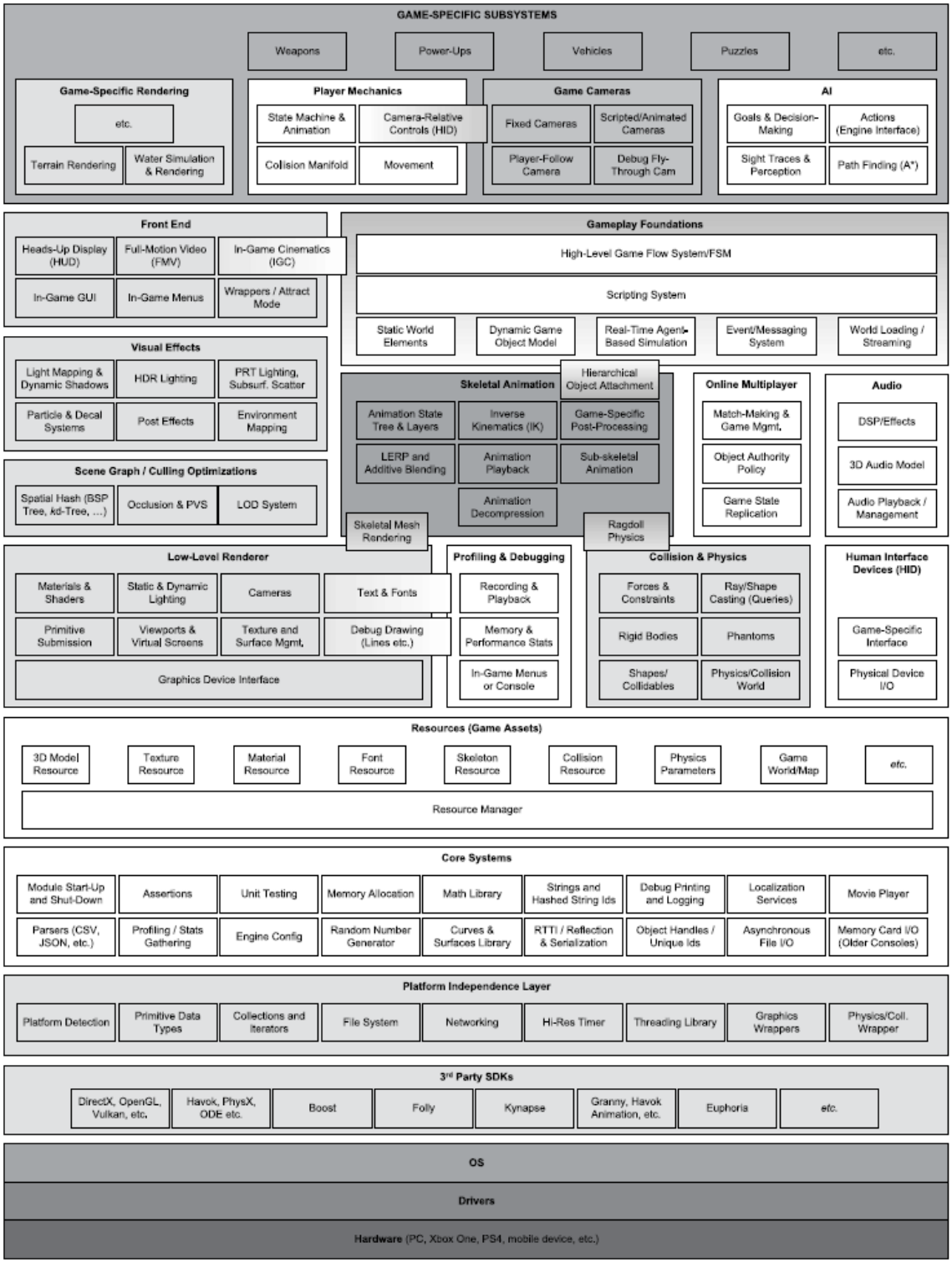
Žádný herní engine není perfektně oddělen od hry. Existují sice enginy použitelné na širší škálu her, ovšem většina enginů jsou alespoň z části vyvinuty na určitý typ hry běžící na specifickém hardwaru. Obecně platí, že čím obecněji je engine zaměřen, tím méně optimální může engine být pro konkrétní druh hry či herní platformy.

Jedním z důvodů může být například zaměřenost na určitý typ prostředí. Například v rámci enginů vyvinutých pro převážně interiérové světy je třeba zajistit, aby nebyly vykreslovány místnosti, do kterých hráč nevidí, převážně pomocí portálových systémů, které rozdělí prostředí na jednotlivé sekce.

Engine zaměřený na vykreslování venkovních prostředí může tento portálový způsob používat velmi zřídka, nebo třeba vůbec. Tyto enginy mohou naopak více používat takzvaných level-of-detail (dále jen „LOD“) technik, kdy objekty dál od hráče jsou vykreslovány v menším detailu a technik zakrývání, přičemž engine neustále vypočítává, zda pohled hráče není něčím zakryt a jaká část světa za vizuální překážkou má být vykreslena.

Neznamená to však, že nelze použít engine více zaměřený na interiéry (například Source Engine) pro tvorbu otevřených venkovních prostorů. Výsledný výkon pro koncového hráče ovšem nebude optimální.

Herní engine obvykle obsahuje runtime komponent a sadu nástrojů. Bývá rozdělen do vrstev, kdy vrchní vrstvy závisí na spodních vrstvách, ale ne naopak. Herní engine je velmi komplexní software systém. Následující diagram představuje obecnou architekturu typického 3D herního enginu. (Gregory, 2018)



Obrázek 2 – Diagram architektury 3D herního enginu (Gregory, 2018)

1.4 Herní světy

1.4.1 Elementy

Většina her se odehrává ve dvou či tří dimenzionáních virtuálních herních světech. Tyto světy se obvykle skládají z mnoha diskrétních elementů, které lze obecně kategorizovat na statické a dynamické.

Statické elementy zahrnují terén, budovy, silnice, mosty a jakékoliv objekty, které se nepohybují. Mezi dynamické elementy patří například postavy, dopravní prostředky, zbraně a vybavení, částicové efekty, dynamická světla, neviditelné regiony, které spouští nějaké relé, a podobně. (Gregory, 2018)

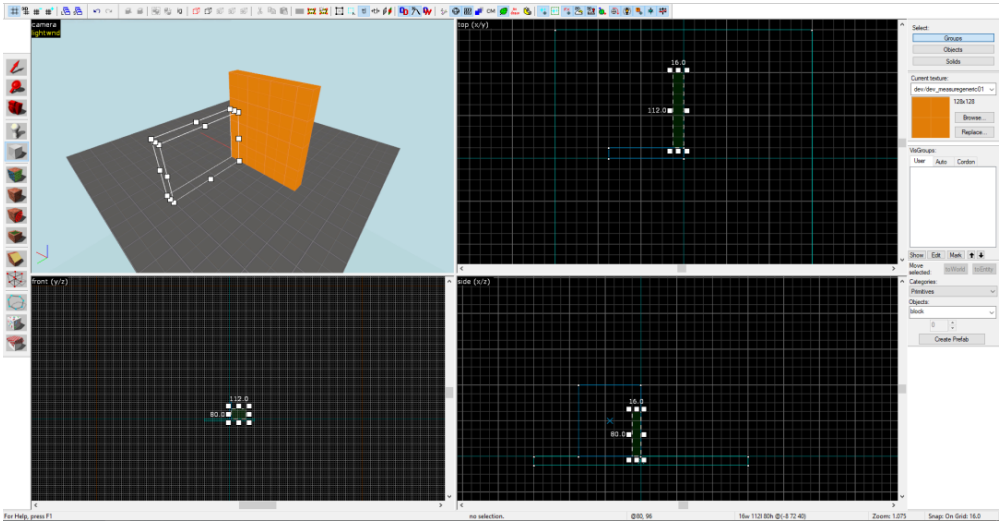
Obrázek 3 – Zobrazení statických a dynamických elementů herního světa (Gregory, 2018)

Samotná herní logika je realizována v doméně dynamických elementů, ovšem statické elementy také hrají svoji roli v samotném hraní – například definují půdorys. Poměr statických a dynamických elementů se liší v každé individuální hře. Většina her ovšem používá malé množství dynamických elementů v relativně rozsáhlém statickém světě, jelikož dynamické elementy mají mnohem výraznější dopad na výkon a často je také jejich počet enginem limitován. (Gregory, 2018)

1.4.2 Statická geometrie

Způsob vykresování statické geometrie se liší napříč různými enginy. Ta může být řešena jako jedna velká síť spojených trojúhelníků nebo může být rozdělena do samostatných částí. Statické části scény mohou být stavěny z takzvaných „instancovaných geometrických tvarů“. V tomto způsobu stavění a vykreslování světa se statická část skládá z předem definovaných tvarů, které jsou ve světě vykreslovány v různých lokacích a pozicích rotace (což způsobuje iluzi variace). Tato technika šetří výpočetní paměť enginu.

Statické elementy také mohou být stavěny z „brush“ geometrie. Ty reprezentují tvar a kolekci konvexních prostorů ohraničených zdmi. Tento způsob tvorby stěn je relativně rychlý a jednoduchý, zároveň je dobře integrovatelný do vykreslovacích enginů založených na způsobu vykreslování „BSP-tree“. (Gregory, 2018)

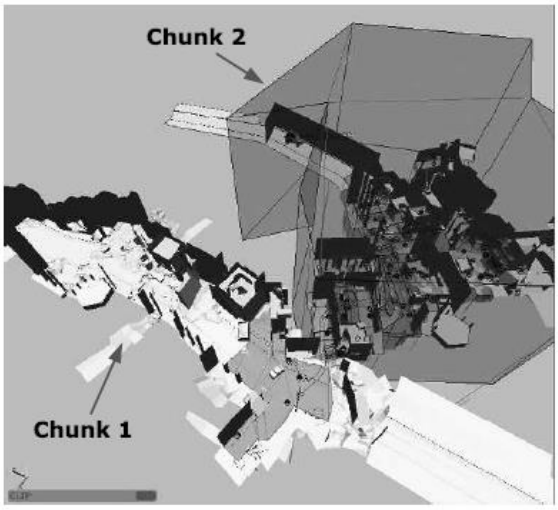


Obrázek 4 – Brush geometrie (Source Engine, vlastní zpracování)

1.4.3 Světové „chunky“

Typicky jsou z důvodu šetření výpočetním výkonem virtuální světy rozděleny do menších částí – hráči se vykreslují jen ty části, které vidí. V minulosti, kdy byl hardware nejvýraznější limitací herních enginů, byl přechod mezi těmito částmi velmi zjevný (například načítací obrazovky při přechodu do další části), ovšem novější enginy tento přechod zvládají plynuleji.

Herní světy mohou mít různé topologie. Existují světy s hvězdicovou topologií, kde hráč začíná hru v centrální lokaci, která má na sebe napojeno několik postranních lokací. Na druhou stranu mohou být herní světy více lineární, kde hráč prochází předem definovanou cestou s malými výlukami a alternativními trasami, které se později opět spojují do lineární cesty. V obou případech (a i v případech zdánlivě velmi malého herního světa) se provádí tato segmentace světa na menší izolované části. Nejen, že je tak ušetřen výkon, ale tento způsob segmentace zároveň zjednodušuje rozdělení práce pro vývojářské týmy. (Gregory, 2018)



Obrázek 5 – Příklad rozdělení herního světa do izolovaných částí (Gregory, 2018)

1.4.4 Implementace dynamických elementů

Herní dynamické elementy bývají tvořeny objektově orientovaným způsobem. Tyto elementy jsou předem definovány, jsou jim přiřazeny různé atributy i chování a designer herního světa je pak v tomto světě rozmístí a nastaví jim potřebné vlastnosti. Často je zde také používáno dědičnosti při tvorbě těchto dynamických elementů, například všechny fyzikální objekty mohou zdědit atribut váhy pro výpočet gravitační síly. (Gregory, 2018)

1.5 Funkcionality mapový editorů

Mapové editory (dále jen „editory“) umožňují vývojářům využít všechny vytvořené assety pro vyplnění herního světa a také umožňují specifikaci vlastností a chování konkrétních dynamických elementů. Mapové editory a jejich funkcionality se opět liší napříč různými herními enginy, ovšem obecně mají několik sdílených funkcionalit přítomných v téměř všech aktuálních editorech. (Gregory, 2018)

1.5.1 Tvorba světových „chunků“

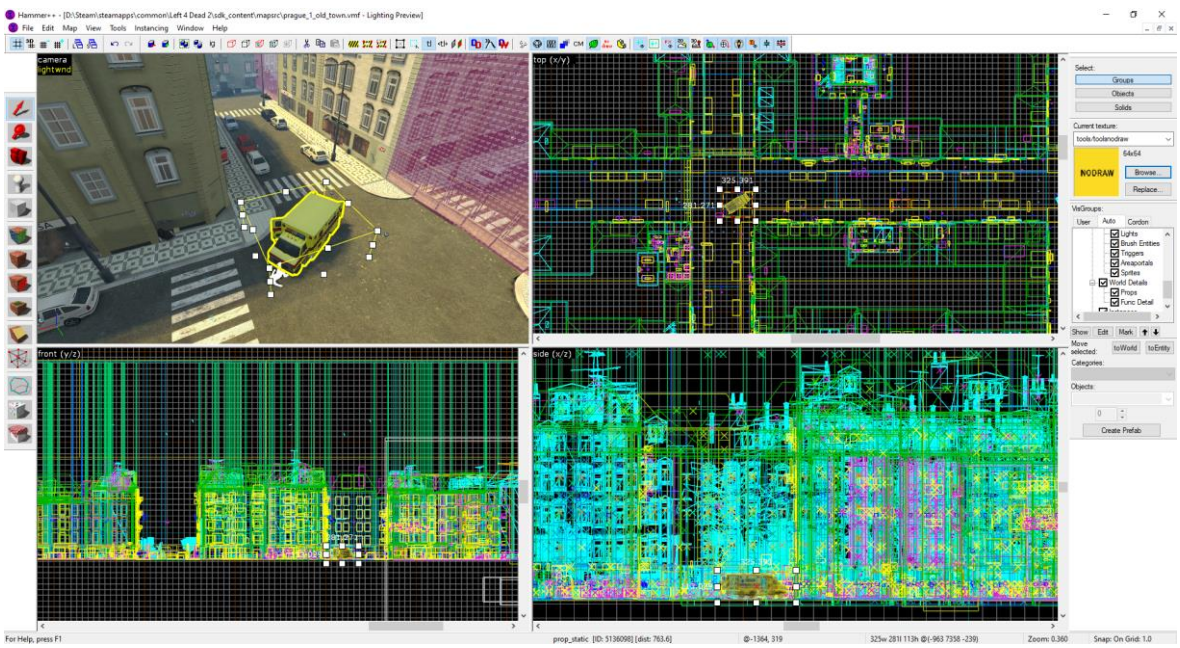
Jednotka tvorby světa často bývá „chunk“, neboli část světa (může také znamenat samostatnou mapu/level). Editor obvykle umožňuje tvorbu těchto chunků, jejich kombinaci či segmentaci. Chunk může být v enginu definován jednou společnou sítí pozadí, nebo může naopak existovat nezávisle (často definován hranicemi).

Editory mohou obsahovat možnosti přímého tvarování terénu, definice vody a dalších specializovaných statických elementů. Jiné editory mohou umožňovat odkázání na určitý typ elementu (například aplikováním určitého materiálu, engine toto chápe jako odkaz na potřebnou funkcionalitu elementu). (Gregory, 2018)

1.5.2 Vizualizace a navigace

Aby byl vývoj herních světů co nejvíce intuitivní a přístupný, je potřeba aby měl designer herního světa možnost různých pohledů na tvořený svět. Často editory poskytují 3D pohled reprezentující to, jak svět vypadá ve hře (s různou mírou přesnosti). Dále často poskytují 2D pohled ze shora a dva pohledy z různých stran.

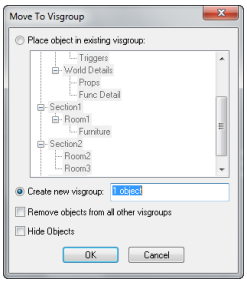
Zároveň je také potřeba, aby byl vývojář schopen se po tomto tvořeném světě pohybovat, bývá tedy standardem možnost „prolétávání“ světem, vyhledávání konkrétních elementů a automatické koncentrování pohledů na tyto elementy. (Gregory, 2018)



Obrázek 6 – Vizualizace v editoru Hammer (Source Engine, vlastní zpracování)

1.5.3 Vrstvení

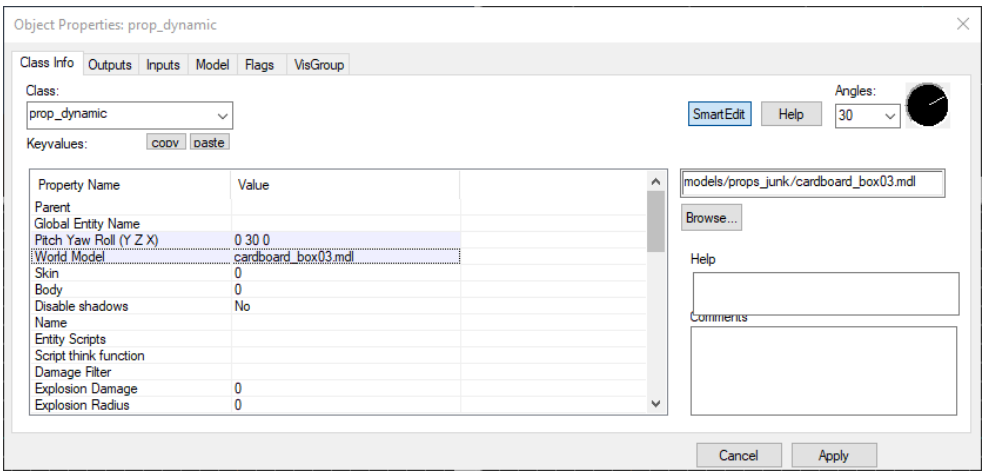
Při tvorbě světa bývají pohledy z editoru na něj velmi nepřehledné. Nejen, že je svět vyplněn potřebnými viditelnými elementy, jsou zde také různé „triggery“ herní logiky či další diskrétní objekty. Editory tedy velmi často umožňují vrstvení herního světa do samostatně ovladatelných skupin, které lze skrýt pro lepší přehlednost při tvorbě. Zároveň toto vrstvení může umožnit více lidem pracovat na jednom světě bez konfliktů v paměti. (Gregory, 2018)



Obrázek 7 – Vrstvení pomocí „Visgroups“ v Source Enginu (Valve, 2005)

1.5.4 Nastavení vlastností objektů

Statické a dynamické elementy často mívají různé vlastnosti (atributy), které lze velmi jednoduše nastavit vývojářem. Může se jednat o názvy objektů (pro využití v herní logice), číselné hodnoty (například vzdálenost, za kterou se objekt přestane vykreslovat) či více komplexní hodnoty jako RGB (red, green, blue) hodnota barvy a odkazy na externí assety (zvuky, animace, modely...). (Gregory, 2018)



Obrázek 8 – Nastavení vlastností dynamického modelu v Source Engine (vlastní zpracování)

1.5.5 Rapidní iterace

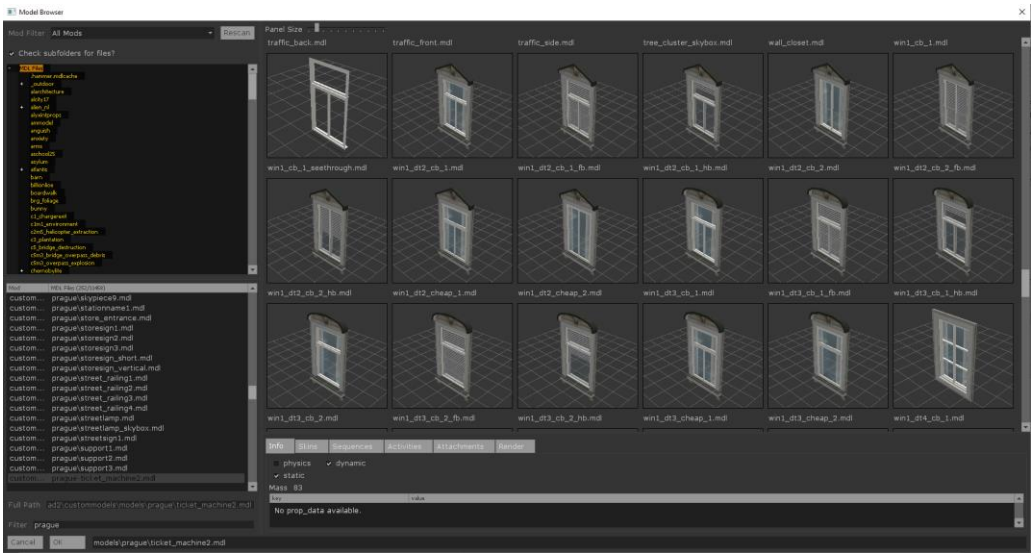
Modernější editory umožňují vytvořené části světa rychle sestavit a testovat. Tyto editory běží v samotné hře a vývojář vidí výsledky své práce v reálné čase. Převážně starší editory naopak vyžadují zdlouhavý proces kompilace světa (zápekání světel, segmentování chunků) před jeho testováním.

Existují ovšem také editory, které mají čas iterace variabilní, například změna nějaké číselné hodnoty elementů se projeví velmi rychle, naopak změna osvětlení bude trvat déle, jelikož je potřeba znovu vypočítat osvětlení pro celý svět. (Gregory, 2018)

1.5.6 Správce assetů

Mnoho editorů obsahuje integrovaného správce assetů, který umožňuje vývojáři vybírat mezi potřebnými texturami, modely, zvuky a podobně. Dobrým příkladem je UnrealEd, tento editor je určený pro tvorbu obsahu her běžících na Unreal Engine. Umožňuje procházet všechny dostupné assety v reálném čase přesně tak, jak doopravdy vypadají, a změny v herním světě jsou ihned testovatelné díky přímému propojení s herním enginem.

Zároveň ovšem také existují méně intuitivní editory, které buď prohlížeč assetů vůbec nemají a vývojář musí definovat cestu k vyžadovaným assetům, nebo je prohlížeč nepřehledný a omezující (například lze vždy prohlížet jen jeden asset, procházení je velmi pomalé). (Gregory, 2018)



Obrázek 9 – Prohlížeč modelů v editoru Hammer++
(Source Engine, vlastní zpracování)

1.6 Charakteristika schopností a limitací Source Engine

Source Engine je 3D herní engine vyvinut společností Valve™ v roce 2004. Tento engine je převážně určen na hry z pohledu z první osoby a herní světy tohoto enginu jsou převážně malé uzavřené detailní prostory. (Bernier, 2014)

1.6.1 Verze Source Engine

Předchůdcem tohoto enginu je GoldSrc Engine převážně známý díky hře Half-Life, která na tomto enginu funguje. Od roku 2015 společnost Valve pomalu začala Source Engine nahrazovat enginem Source 2, který je snahou o modernizaci tohoto velmi dlouho používaného enginu. Technologie využívané enginem Source byly při jeho prvním užití ve hře Half-Life 2 revoluční, ovšem dnes už jsou tyto technologie převážně neaktuální.

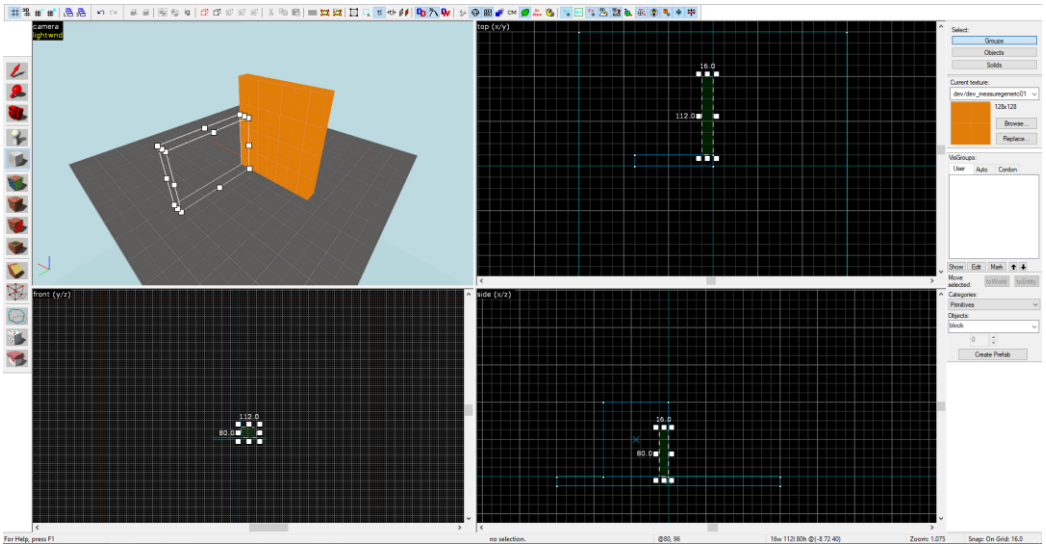
Existuje mnoho verzí Source Enginu, společnost Valve postupem času engine modifikovala podle potřeb vyvíjených her a nejnovější verze Source Enginu – „Counter-Strike: Global Offensive branch“, podporuje i moderní funkcionality jako například dynamické stíny v reálném čase či zastínění okolí (ambient occlusion). (Valve, 2005)

Obrázek 10 – Přelomní verze Source Engine (vlastní zpracování)

1.6.2 Vykreslování ploch

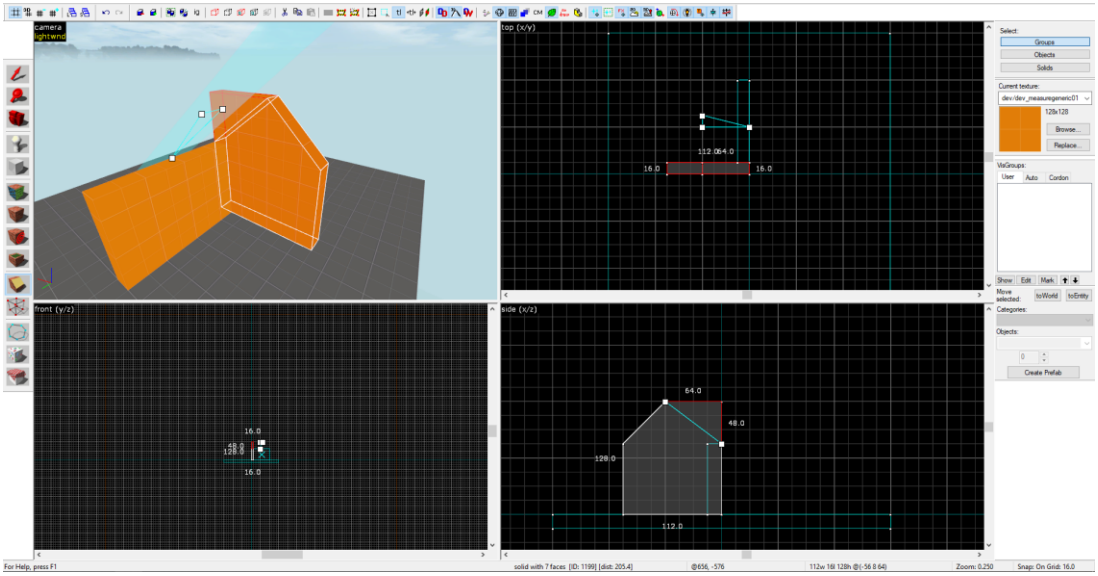
Plochy herních světů Source Engine se skládají z geometrických tvarů (brushes) a terénu (displacements). Na tyto plochy je vykreslována definovaná textura. Plochy mohou simulovat hloubku pomocí „normálních map“ a reflektivitu pomocí „cubemap“ za pomoci reflektivní masky. Většina ploch také odráží světlo, což do určité míry simuluje osvětlení v reálném světě.

V rámci tvorby geometrických ploch (dále jen „brush“) Source podporuje základní tvary jako kvádr, kužel, válec a složitější tvary, jako například oblouk a koule. Tyto tvary musí být v souladu s mřížkou (grid) editoru, kde nejmenší jednotka je zhruba 2,5 cm. (Bernier, 2014)



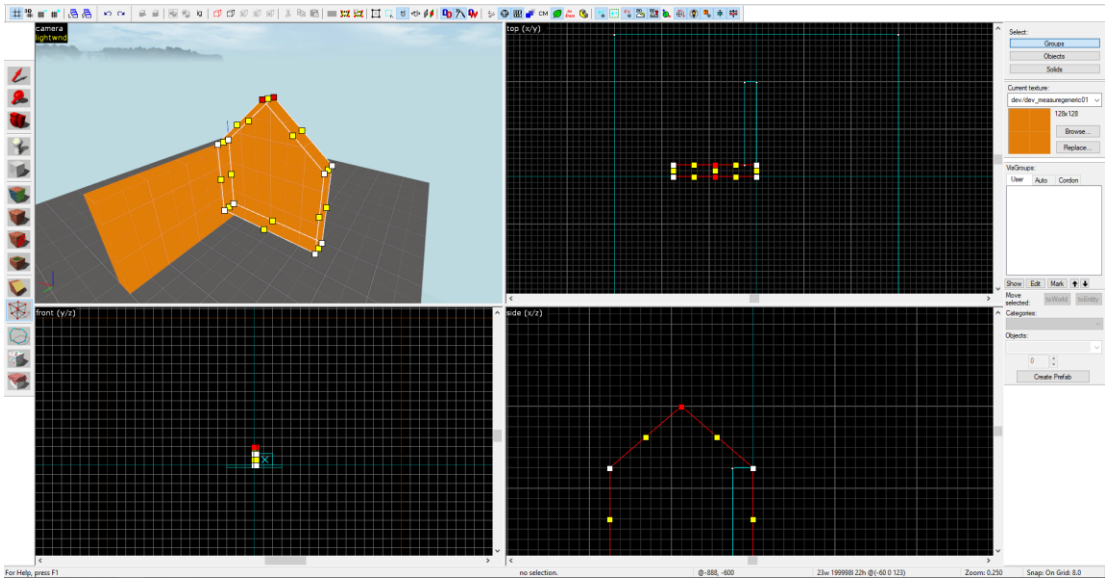
Obrázek 11 – Tvorba plochy (Hammer, vlastní zpracování)

S výslednými plochami lze dále manipulovat, nejčastěji pomocí nástroje sekání, pomocí kterého lze z jednoduchých geometrických tvarů vytvořit tvary komplexnější. Je ovšem dobrým pravidlem se při úhlovém sekáním držet násobků 45°. (Bernier, 2014)



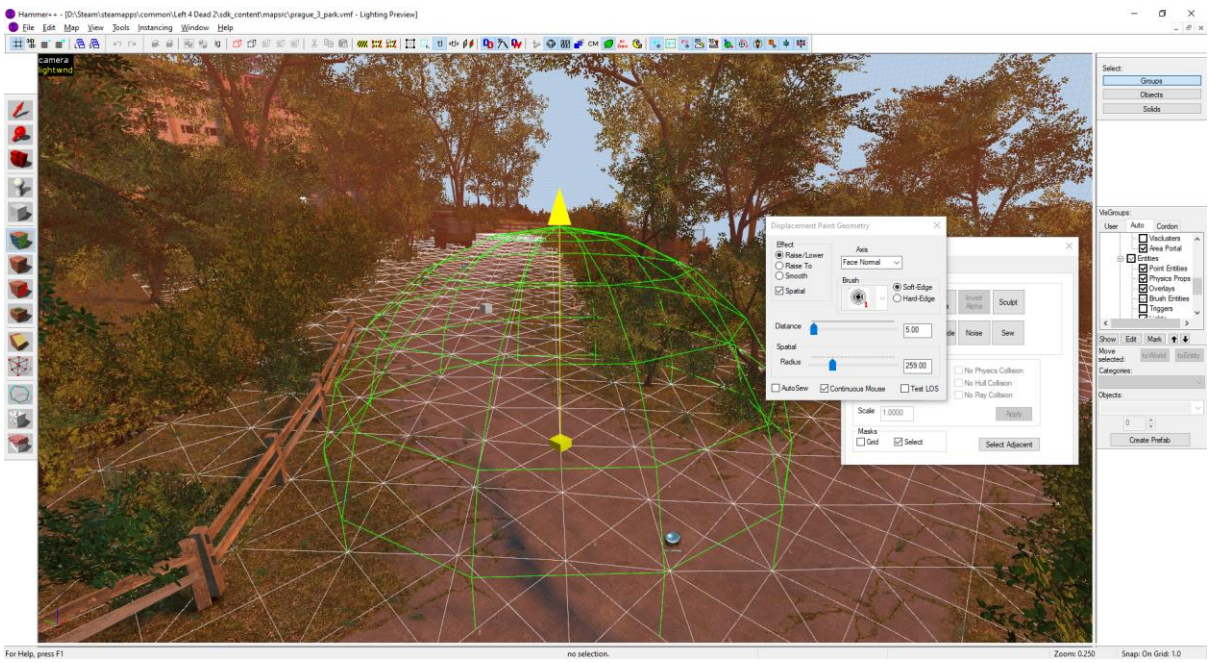
Obrázek 12 – Sekání ploch (Hammer, vlastní zpracování)

Velmi specifickým a méně často používaným nástrojem pro manipulaci ploch je vertex nástroj. Ten umožňuje výběr jednoho či více rohových bodů a následovnou manipulaci. Nevhodné zacházení s tímto nástrojem ovšem může způsobit korupci plochy a následně její automatické smazání. (Bernier, 2014)



Obrázek 13 – Vertexní manipulace (Hammer, vlastní zpracování)

Pro nerovné plochy je vhodné využít terén, neboli „displacement“ v Source Engine. Je tvořen stejně jako standardní plocha, následně je vybrán nástrojem pro manipulaci textur a v okně „Displacement“ je zněj generován terén určitého rozlišení (od 2 do 4, 4 je nejdetailnější a nejnáročnější na výkon). Dále lze pak terén zvýšit či snížit všemi možnými směry a lze jej také vyhladit. (Bernier, 2014)

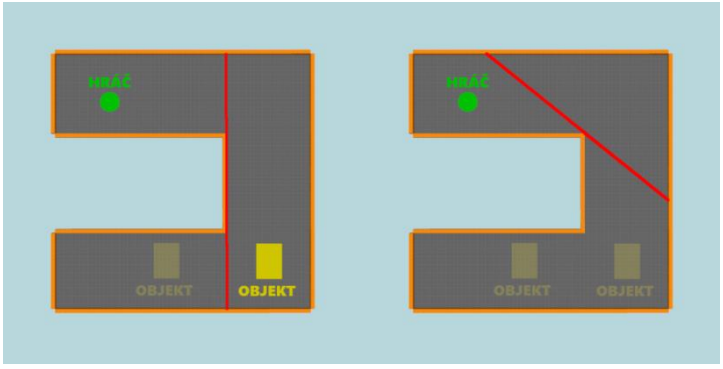


Obrázek 14 – Tvarování terénu v Hammer++ (vlastní zpracování)

1.6.3 Optimalizace

Optimalizace herních světů v Source Engine funguje na bázi rozdělení prostředí pomocí „visleafs“ na jednoduché geometrické tvary (většinou krychle a kvádry). Každý tvar pak v sobě uchovává informaci o tom, jaké ostatní tvary tento tvar vidí. Vykreslovač poté hráči zobrazuje části světa podle toho, v jakém geometrickém tvaru zrovna je, a vykresluje objekty ve všech tvarech, které lze potenciálně z aktuálního tvaru vidět.

Tento proces je kompilerem prováděn automaticky, avšak výsledky tohoto procesu obvykle nejsou postačující (obrázek vlevo). Je tedy často potřeba manuálně definovat, jak má engine visleafs rozdělit, a to tak, aby neměly přímý pohled na visleafs, které vykreslovat nechceme. (Bernier, 2014) (MangyCarface, 2009)



Obrázek 15 – Automatické rozdělení visleafs vs. manuální rozdělení pod 45° (vlastní zpracování)

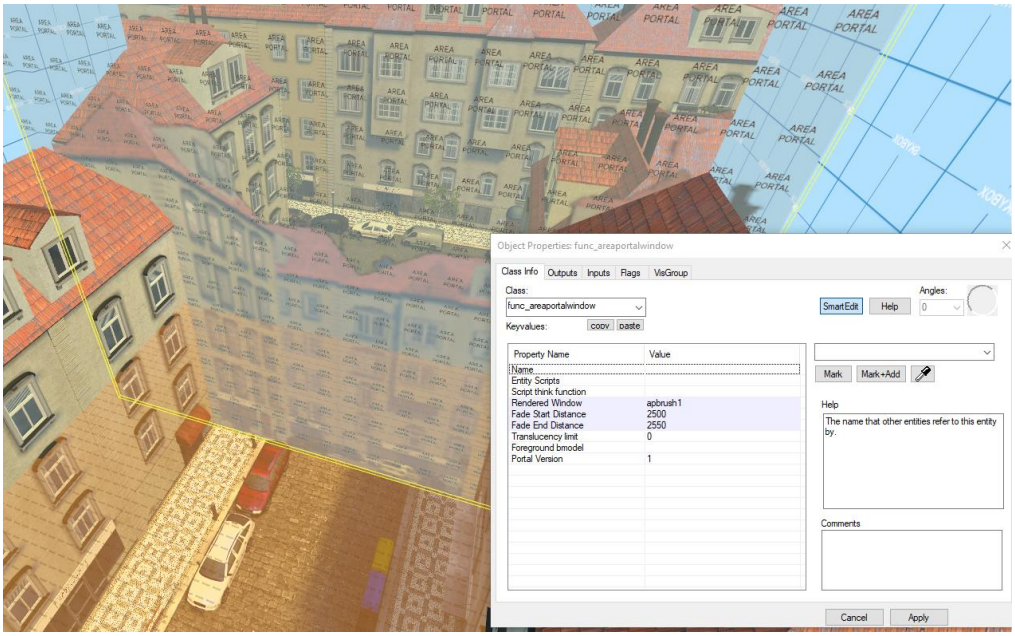
V procesu optimalizace tohoto typu je také třeba dbát na různé perspektivy, jelikož visleafs jsou trojrozměrné prostory. Pokud například mezi hráčem a objektem stojí zeď, kompilátor většinou vytvoří vysoké visleafs, které na sebe přes zeď stále vidí. Hráči jsou pak vykreslovány objekty, které není potřeba vykreslovat.

Často je tedy potřeba manuálně rozdělit visleafs, a to jak pohledem ze shora, tak pohledem ze strany na úrovni vrchu stěn. (Bernier, 2014) (MangyCarface, 2009)

Obrázek 16 – Výchozí rozdělení visleafs ze strany vs. manuální rozdělení (vlastní zpracování)

Další téměř vždy používanou technikou je nastavení maximální vykreslovací vzdálenosti (dále jen „far-z clip“). Tuto hranici obvykle vývojáři schovávají pomocí mlhy. Source Engine také umožňuje využití LOD modelů, tedy jednodušších verzí modelů, které jsou vykreslovány po přesažení definované vzdálenosti od hráče. Tyto LOD modely ovšem nejsou automaticky vytvářené enginem. Vývojáři musí manuálně vytvořit LOD modely pro všechny potřebné objekty.

Poslední často používanou technologií pro optimalizaci jsou areaportály. Ty jsou využívané převážně pro zakrytí interiérů viditelných z venkovních prostorů, ovšem lze je také občas využít při rozdělování sousedních venkovních lokací. Areaportály mohou být vázány buď na dveře (při zavření dveří není vykreslován interiér), nebo na okno, na kterém je potřeba definovat texturu, která bude vykreslována v okně a zakryje tak nevykreslovaný interiér za portálem. Když jsou portály otevřené (buď jsou otevřené definované dveře, nebo je hráč dostatečně blízko okna), engine vypočítává, co přesně hráč skrz portál vidí. To je však náročné na výpočetní zdroje a je tedy potřeba tyto portály využívat zřídka, aby naopak nezpůsobily horší výkon. (Bernier, 2014) (MangyCarface, 2009)

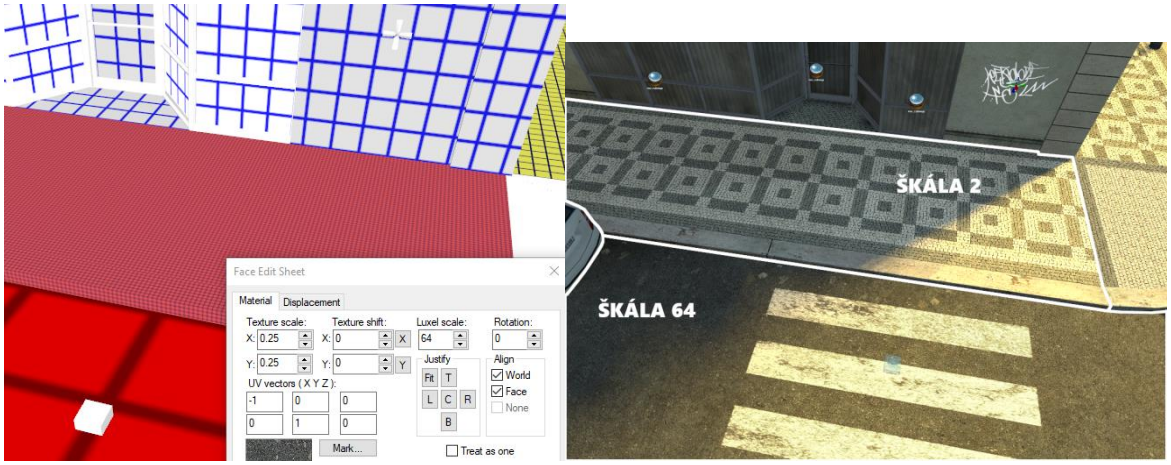


Obrázek 17 – Areaportál typu okno – odděluje 2 části města při přesažení vzdálenosti 2550 jednotek od areaportálu (vlastní zpracování)

Obrázek 18 – Zavřený areaportál, nic za portálem není vykreslováno (vlastní zpracování)

1.6.4 Osvětlení

Osvětlení v Source Engine je zapékáno do ploch pomocí světelných map. Každý povrch v herním světě má definovanou škálu světelných map (výchozí hodnota je 16). Světlo a stíny jsou pak do ploch zapékány podle požadované přesnosti, přičemž menší škály světelných map znamenají detailnější osvětlení a stíny. (Bernier, 2014) (Valve, 2012)



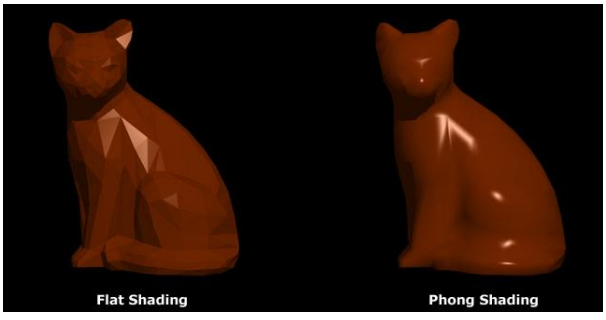
Obrázek 19 – Rozdíly v nastavení světelné škály různé přenosti (vlastní zpracování)

Source Engine také podporuje dynamické osvětlení a stíny, ovšem s omezením jednoho aktivního dynamického světla ve vykreslované části světa hráče. Často bývá toto jedno povolené dynamické světlo využito na simulaci baterky hráče, přičemž každý hráč vidí pouze svou baterku, zatímco baterky ostatních hráčů jsou vykreslovány pouze jako iluze světla. (Bernier, 2014) (Valve, 2012)

Obrázek 20 – Příklad dynamického světla pomocí entity env_projectedtexture (Valve, 2012)

Navzdory tomu, že jsou světla permanentně zapékána do ploch, mohou vývojáři simulovat světlo, které lze vypnout a zapnout. Source Engine poté ovšem musí zapékat světlo dvakrát, jednou jako vypnuté a jednou jako zapnuté. Pokud je v jedné oblasti mnoho světel, které lze nezávisle vypnout a zapnout, počet potřebných zápeků světel exponenicálně roste, a proto tuto funkcionalitu nenajdeme v téměř žádné oficiální hře běžící na tomto enginu.

Verze Source Engine od roku 2006 a dále také podporují takzvaný „Phong shading“, který dokáže aproximativně osvětlit textury převážně na modelech charakterů. Obecně se jedná se o zjednodušenou metodu zjišťování osvětlení na specifických bodech 3D povrchu, která počítá s měnící se výškou povrchu a vyhlazuje přechody interpolací. (Bernier, 2014) (Valve, 2012)



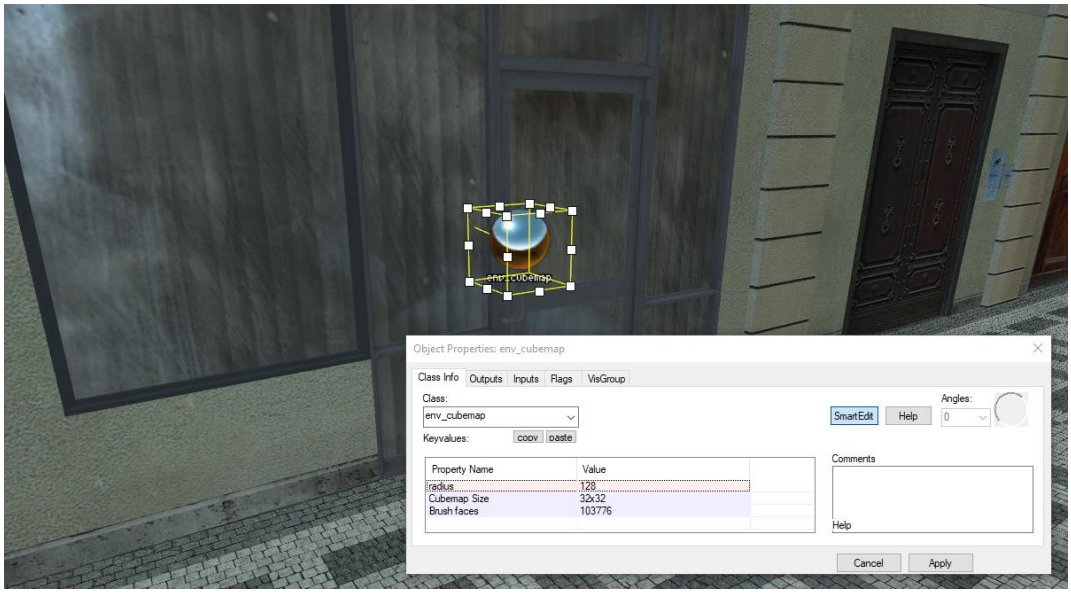
Obrázek 21 – Phong shading (the_best_flash, 2022)

Source Engine používá metody kombinace lokálního a enironmentálního osvětlení pro výpočet osvětlení modelů a dále pak aplikuje vrstvu spekulárních zvýraznění pro výsledný efekt. (Bernier, 2014) (Valve, 2012)

Obrázek 22 – Kombinace technik osvětlení pro dynamické objekty (the_best_flash, 2022)

1.6.5 Simulace odrazů

Source Engine převážně používá předvykreslené environmentální mapy ve formě cubemap pro simulaci odrazů na modelech a plochách. Cubemapa je obvykle malá textura reprezentující 360° pohled v definované lokaci. Na plochách a modelech, které mají simulovat odraz v dané lokaci, je pak promítán korespondující pohled takovým způsobem, aby byla vytvořena iluze odrazu prostředí. (Bernier, 2014) (Valve, 2023)



Obrázek 23 – Příklad využití env_cubemap (vlastní zpracování)

Dále Source Engine také podporuje planární odrazy, které jsou převážně používány na vodních texturách. Tento efekt je ovšem v Source Engine velmi výpočetně náročný a je omezen na jednu skupinu planárních ploch stejné souřadnice výšky ve vykreslované části hráče. V případě potřeby dvou vodních ploch v různě vyvýšených pozicích je třeba využít vodních textur bez planárních odrazů. (Bernier, 2014) (Mariteaux, 2017)

Obrázek 24 – Příklad planárních odrazů v Source Engine (Mariteaux, 2017)

1.6.6 Limitace

Jelikož se jedná o relativně starý herní engine, obsahuje spoustu redundantních omezení, které v modernějších herních enginech převážně nebývají. Mezi tyto omezení patří například limit toho, jak detailní mohou být celkově škály světelných map celého světa, a vývojáři tedy často musí oblastem mimo hrací plochu nastavit větší škálu světelných map, aby mohli nastavit oblasti v hratelné části světa na detailnější škálu.

Dále má většina verzí Source zdánlivě arbitrární limit modelů v herním světě. Pro statické modely je limit 4096 a dynamické modely spadají do limitu dynamických entit, kterých může být dohromady také jen 4096. Ovšem dynamické entity jsou dále rozděleny do dvou kategorií s limitem 2048 pro každou z nich. Maximální potenciální počet dynamických modelů ve světě je tedy 2048 (ovšem pravděpodobně méně, jelikož například částicové efekty, fyzikální modely, zbraně, sprity a monoho dalších často užívaných entity do tohoto limitu spadají také.

V rámci Source Engine je také omezen počet světel různé barvy pro každou plochu. Vývojáři mohou aplikovat maximálně 4 světla různé barvy pro každou plochu herního světa. (Bernier, 2014) (Valve, 2009)

Vlastní práce

1.7 Porovnání Source Engine s novějšími enginy a jeho předchůdcem

1.7.1 Porovnání Source s Unreal Engine 5

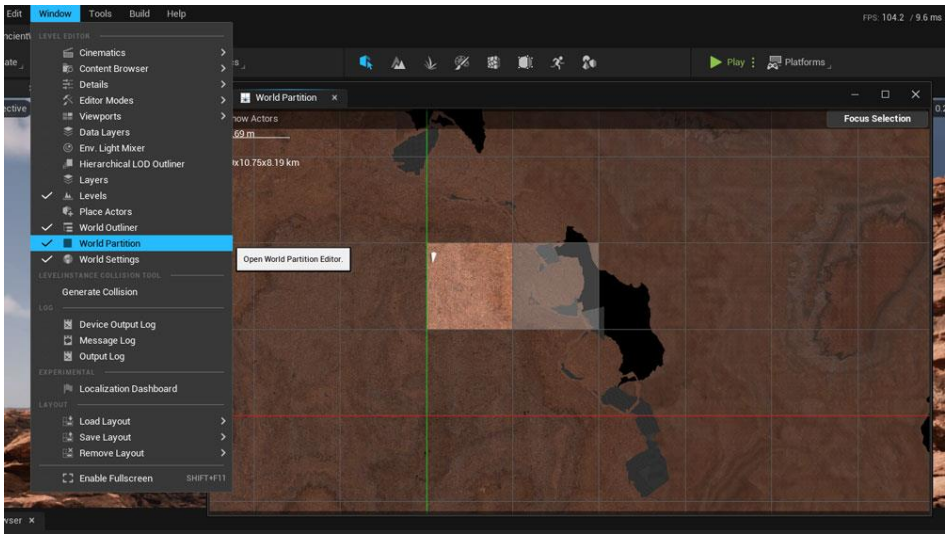
Unreal Engine 5 přináší revoluční technologii „Nanite“, která umožňuje vykreslování mnoha komplexních modelů zároveň díky automatickému generování LOD modelů. V ostatních enginech včetně Source Engine je třeba tyto LOD modely tvořit manuálně. Nanite umožňuje importovat jakýkoliv 3D model (třeba i filmové kvality). Tvůrci herního světa pak model jednoduše vloží do světa a nemusí se starat o zátěž na systém.

Další novou technologií Unreal Enginu 5 je „Lumen“, který je blíže definován jako plně dynamický systém globální iluminace. Oproti Source Engine, kde je osvětlení převážně kalkukolváno podle světelných map a zapékáno do prostředí, mohou být světla a stíny v Unreal Engine 5 kalkulovány za běhu enginu. Tato funkcionalita umožňuje simulaci světla v reálném čase a jeho reakci na změny prostředí. (Epic Games, Inc., 2022)

Obrázek 25 – Nanite: Unreal Engine 5 (Epic Games, Inc., 2023)

Unreal Engine také obsahuje komponent zvaný „virtuální stínové mapy“, který umožňuje generování konstantně ostrých a detailních stínů ve velkém otevřeném herním světě. Od standardních implementací stínů v jiných enginech se liší vysokým rozlišením těchto virtuálních stínových map a jelikož je tato úroveň detailu konstantní napříč celým herním světem, hráč nezpozoruje změnu detailnosti stínů v závislosti na vzdálenost. Pro zpracování částicových efektů a tekutin Unreal Engine 5 používá systém „Niagra“ a v rámci simulace fyziky systém zvaný „Chaos“.

Unreal Engine 5.3 umožňuje vývojářům tvořit velmi rozsáhlé světy. V Source Engine je limit jednoho herního světa 0,83 km^2, Unreal Engine 5 používá „word partition“ systém, který svět rozdělí na menší části a načítá jen ty potřebné. Potenciální maximální velikost světa v Unreal Engine je tedy o mnoho větší (21+ km^2). (Epic Games, Inc., 2023)



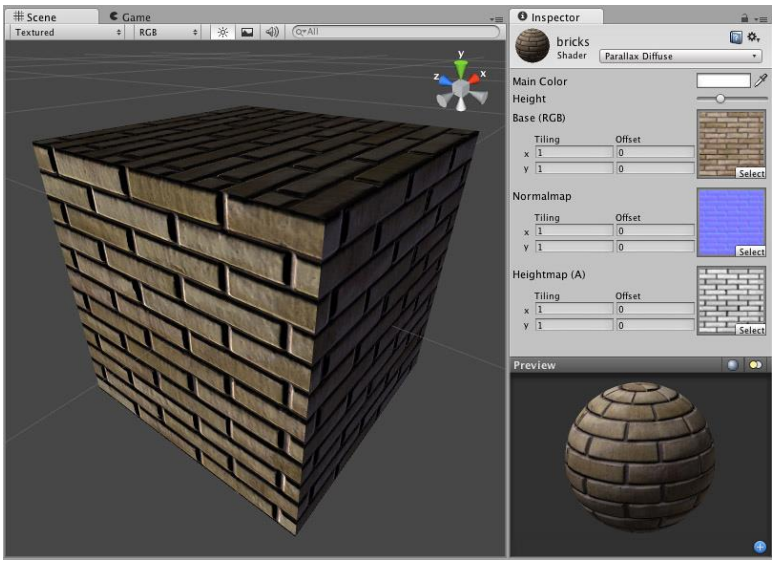
Obrázek 26 – Unreal Engine 5: World Partition systém (Epic Games, Inc., 2023)

Při vývoji může více vývojářů pracovat na stejné části světa díky systému „one file per actor“, zároveň tento engine poskytuje funkcionalitu datových vrstev, která umožňuje tvorbu více variací jednoho světa.

Také poskytuje daleko intuitivnější systém pro tvorbu zvuků a jejich nastavení. V Source Engine musí vývojář manuálně vytvořit soubor skriptů, ve kterém definuje dosah, hlasitost, výšku/hloubku a zvukový kanál pro konkrétní zvukové soubory. Ovšem v Unreal Engine 5 je tento proces daleko uživatelsky přívětivější díky grafickému rozhraní umožňující nastavení těchto atributů i mnoho dalších nedostupných v Source.
(Epic Games, Inc., 2022)

1.7.2 Porovnání Source s Unity

Unity je herní engine často používaný malými nezávislými vývojářskými studii, který umožňuje tvorbu her jak ve 2D, tak i ve 3D perspektivě. Tento engine poskytuje podobné funkcionality jako Source Engine, konkrétně kompresi textur, „mip mapy“, „bump mapy“, reflekční mapy, efekty na celou obrazovku a také další funkcionality, které Source nepodporuje, jako například „paralaxní mapování“, „screen space ambient occlusion“ a dynamické stíny pomocí stínových map. (Menard, 2015) (Satheesh, 2016)



Obrázek 27 – Unity: parallaxní mapování (Unity Technologies, 2023)

Unity využívá takzvaný „render pipeline“, což je systém, který provádí operace analýzy scény a následného vykreslování na obrazovku. Na vysoké úrovni to jsou operace „zakrývání“, „vykreslování“ a „post-process“. Na rozdíl od Source Engine Unity vývojářům poskytuje několik „render pipelines“ s různými schopnostmi a odlišným výkonem.

Konkrétně si mohou vývojáři vybrat mezi základní „built-in“ verzí, univerzální verzí a „high-definition“ verzí. Zároveň umožňuje vývojářům vytvořit si vlastní. Obvykle je obtížné hru převést z jedné render pipeline na jinou, jelikož často využívají různé výstupy shaderů.

Unity využívá .NET platformu, díky které tento engine dokáže fungovat na různých hardware konfiguracích. Platforma .NET podporuje široké rozmezí jazyku a API knihoven. Source Engine je ve srovnání omezen na operační systémy Windows, MacOS a herní konzole Xbox, Xbox 360 a PlayStation 3. (Menard, 2015) (Unity Technologies, 2023)

1.7.3 Porovnání s předchůdcem – GoldSrc

GoldSrc Engine je předchůdcem Source Enginu. Byl opět vyvinut společností Valve v roce 1996 a poprvé byl prakticky využit v roce 1998 ve hře Half-Life. Jedná se o těžce modifikovanou verzi Quake Enginu vyvinutou společností Id software, která běží na programovacím jazyce C++ (podobně jako Source, který ovšem využívá kombinaci C a C++).

Funkcionality, jako například umělá inteligence, byly vytvořeny kompletně nezávisle na existující verzi Quake Enginu. Valve v rámci GoldSrc také vyvinulo podporu pro skeletální animace umožňující reálnější pohyb a animace obličeje, které jsou na tuto dobu velmi revoluční. Dále GoldSrc podporuje barevná světla (světla v Quake jsou monotónní), transparentní textury a více polygonů pro modely. GoldSrc nepodporuje tradiční způsoby generování odrazů, pouze umožňuje aplikovat sekundární „Chrome“ texturu na jednotlivé modely, čímž může být dosaženo iluze reflektivních materiálů. (Valve, 2005) (Zorn, 2019)

Obrázek 28 – GoldSrc: snímek ze hry Half-Life (Valve, 2005)

GoldSrc a Source nejsou vzájemě kompatibilní, GoldSrc například využívá jiný způsob ukládání informací o plochách a geometrických tvarech než Source, který originální konfiguraci těchto tvarů ukládá do samotného mapového souboru .bsp. GoldSrc také používá formát .WAD (zkratka pro „Where’s All the Data“) na ukládání textur do jedné palety. Source Engine naopak používá jednotlivé .vtf (Valve Texture Format) a .vmt (Valve Material) soubory pro konkrétní textury a jejich atributy.

Data hry vybudované na GoldSrc engine jsou ukládány do .pak souborů, zatímco Source Engine používá vlastní .vpk formát (zkratka pro Valve Pak). (Valve, 2005)

1.7.4 Porovnání s následovníkem – Source 2

Nástupcem Source Engine je engine Source 2 opět vyvinutý v programovacím jazyce C++. Na rozdíl od jeho předchůdců podporuje 64 bitovou architekturu a technologii Vulkan, která umožňuje lepší vícejádrové vykreslování a efektivnější 3D vykreslování. Source 2 má nativní podporu jak pro DirectX 11, tak pro Vulkan, kategoricky tedy Source 2 podporuje Shader Model 5.0 a dále. Díky tomu tento engine podporuje rozsáhlejší a detailnější herní světy se stabilnějším výkonem než v Source a GoldSrc. (Valve, 2017)

Source 2 zavedl integrovaného správce assetů v podobě „asset systému“. Vývojáři také modernizovali nástroje pro tvorbu herních světů, a to například přidáním podpory pro moderní nástroje úpravy sítě polygonů. Source 2 také umožňuje vývojářům využít buď dopřednou, nebo odloženou vykreslovací pipeline.

Valve vyvinulo vlastní engine fyziky zvaný Rubikon, který podporuje technologie jako například simulace fyziky oblečení a tekutin. Tento fyzický engine nahradil dříve využívaný fyzický engine třetí strany – Havok. (Valve, 2017)

Obrázek 29 – Half-Life: Alyx: simulace tekutin (Lang, 2020)

Zároveň také Source 2 nativně podporuje virtuální realitu, prakticky využitou ve hře Half-Life: Alyx vydanou v roce 2020. Ve srovnání Source nepodporuje virtuální realitu nativně, existují ovšem modifikace třetích stran umožňující tuto funkcionalitu v určitých hrách Source Engine. Source 2 již nepodporuje 32 bitové systémy a není přímo kompatibilní s herními světy jeho předchůdce Source. (Valve, 2017)

1.7.5 Souhrnná tabulka porovnání funkčností zvolených herních enginů

	Source	Unreal 5	Unity	Source 2	GoldSrc
Bitová architektura	32 bit	64 bit	32 bit, 64 bit	64 bit	32 bit
Generování osvětlení	Statické, 1 dynamické	Statické, dynamické	Statické, dynamické	Statické, dynamické	Statické
Generování odrazů	Cubemap, 1 planární	Reflection capture actors, screen space, planární	Cubemap, screen space, planar probe	Parallax corrected cubemap, planární	Chrome textura
Maximální velikost světů	0,83 km^2	21 km^2 (teoreticky i více)	10 km^2	0,83 km^2 (teoreticky i více)	0,2 km^2
Vykreslovací API	DirectX9/10/11	DirectX11/12, Metal 2.0, OpenGL, Vulkan	DirectX11/12, Metal, OpenGL, Vulkan,	DirectX 11, Vulkan	OpenGL
Fyzický engine	Havok	Chaos	Nvidia PhysX	Rubikon	QPhysics
Programovací jazyk	C, C++	C++	C#	C++	C, C++

Tabulka 1 – Porovnání funkčností zvolených herních enginů

1.8 Plánování a příprava herního světa

Cílem praktické části je vytvořit herní kampaň skládající se ze 4 navazujících herních světů (dále: mapa) pro hru Left 4 Dead 2 odehrávající se v Praze. V této hře mají hráči za úkol dostat se z bodu A do bodu B, mapy tedy mívají podobu navazujících lokací, které hráče provedou specifickým prostředím stejné tématiky. Plánem je hráče provést určitými částmi Prahy, a to směrem z centra až k hranicím města.

Jednotlivé mapy kampaně budou vytvářeny prostřednictvím modifikovaného editoru Hammer++, který mimo jiné výhody ve srovnání s výchozím editorem Hammer umožňuje zobrazení map s přibližným odhadem stínů a osvětlení. Z herních a technických důvodů není cílem vytvořit přesnou reprezentaci 1:1 Prahy, ale spíše množinu navazujících lokací reprezentujících specifické části Prahy. Jsou zde k dispozici modely a textury poskytované samotnou hrou, avšak oficiální kampaně hry se odehrávají jen ve Spojených státech. Je tedy potřeba vytvořit textury a modely aplikovatelné v evropském městě.

Jelikož Source Engine nepodporuje dynamicky se měnící osvětlení podle denní doby, bude posun času reprezentován formou postupně se měnícího osvětlení v každé mapě. První mapa bude obsahovat osvětlení brzkého odpoledne s odpovídajícím nebem a poslední mapa bude mít noční osvětlení a oblohu. V rámci druhé a třetí mapy bude tento časový posun vyhlazován.

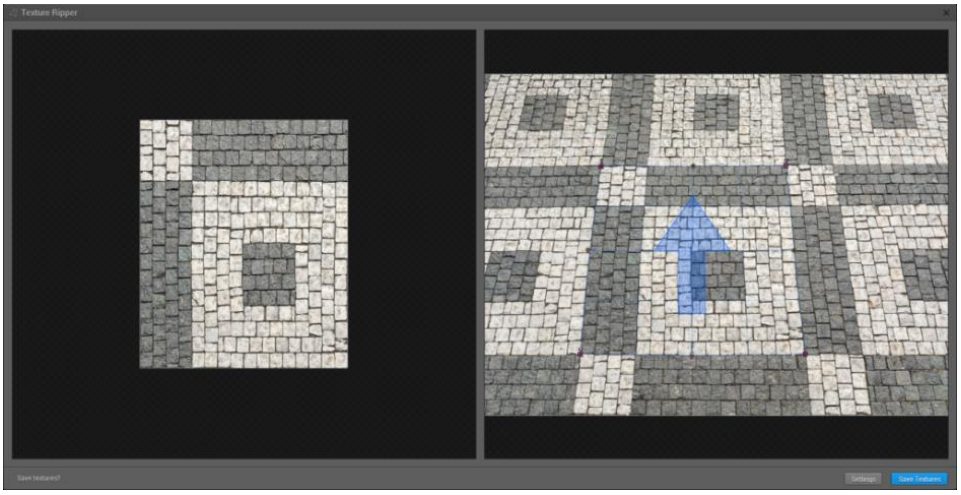
1.8.1 Plánování lokací

Kampaň byla rozdělena do 4 navazujících map, přičemž každá mapa reprezentuje specifickou část Prahy. Samotné mapy se dále skládají z několika navazujících lokací, které reprezentují daný distrikt. Konkrétně je plán map a lokací následující:

1. Staré město
    - Bytový dům
    - Ulice a obchody
    - Staroměstské náměstí
    - Úzké ulice Starého Města
    - Silnice u řeky
    - Stanice metra
2. Okolí centra
    - Pokračování stanice metra
    - Ulice a obchody
    - Vnitroblok
    - Průchod dvěma bytovými domy
    - Park
    - Podchod pod silnicí
3. Sídliště
    - Stánek s občerstvením
    - Les s troskami helikoptéry
    - Sídliště
    - Panelový dům
    - Vnitroblok sídliště
    - Prádelna panelového domu
4. Hranice města
    - Pokračování panelového domu
    - Sídliště
    - Parkoviště
    - Autobusové depo

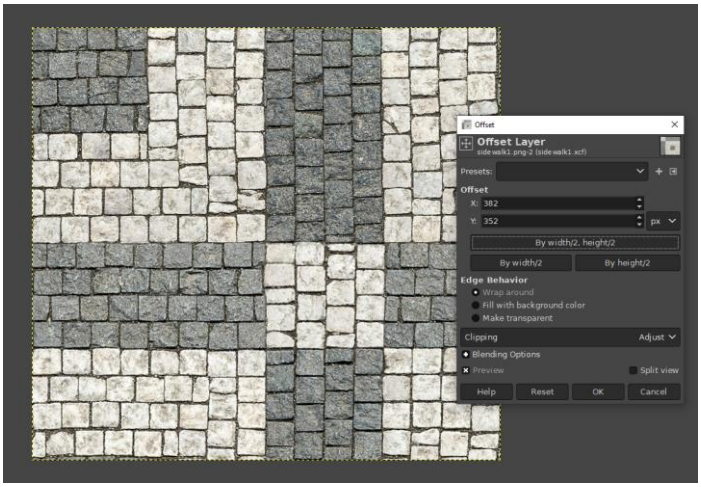
1.8.2 Obecný princip tvorby textur z fotografií

Prvním krokem tvorby textur z fotografií je vyhledání či nafocení fotografie, která je co nejméně nakloněna od fotoaparátu. Dále je fotografie importována do programu „Shoebox“, který umožňuje definování okrajů požadované textury a následně výslednou texturu „narovná“ na plochou perspektivu. Program Shoebox byl vybrán díky svému intuitivnímu ovládání a možnosti bezplatného využití.



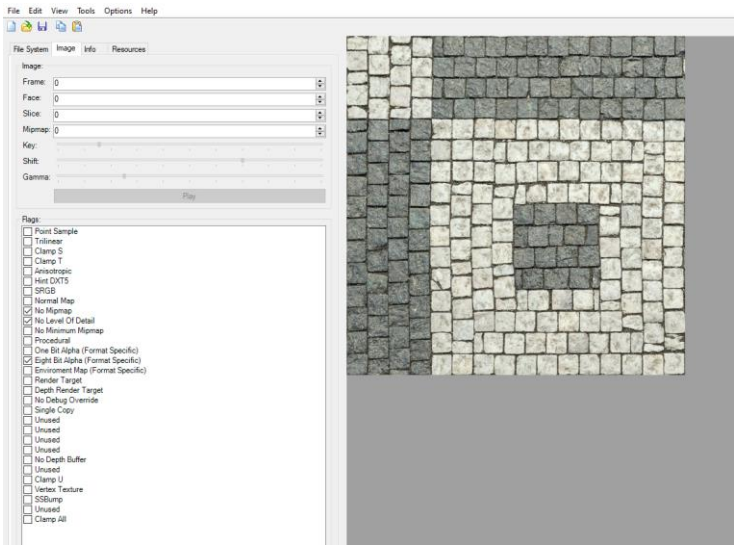
Obrázek 30 – Narovnání perspektivy v programu Shoebox (vlastní zpracování)

Dalším krokem je vyhlazení přechodů při dlaždicovém opakování textury. V následujícím obrázku je zobrazen proces vyhlazování. Nejdříve je obrázek vertikálně a horizontálně posunut o polovinu, čímž jsou zviditelněny přechody na krajích textury. Následně je možné nástroji duplikace a vyhlazování přechod vyhladit. Toto bylo prováděno v programu Gimp, jelikož je přístupný zdarma.



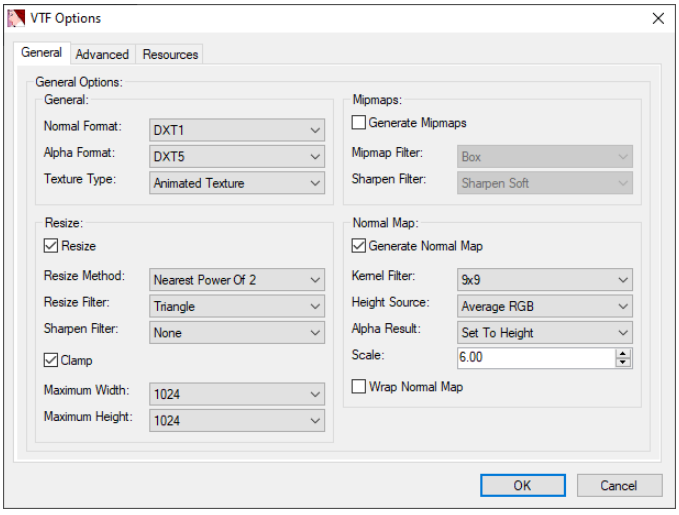
Obrázek 31 – Poloviční vertikální a horizontální posunutí obrázku v programu Gimp (vlastní zpracování)

Dále je potřeba výsledný obrázek převést na formát využitelný Source Enginem – soubor VTF s metadaty VMT. Textura exportovaná z programu Gimp (například .png, .tga, .jpg) je importována do programu VTFedit. Zde lze nastavit rozměry a atributy výsledné textury a uložit ji ve formátu VTF.

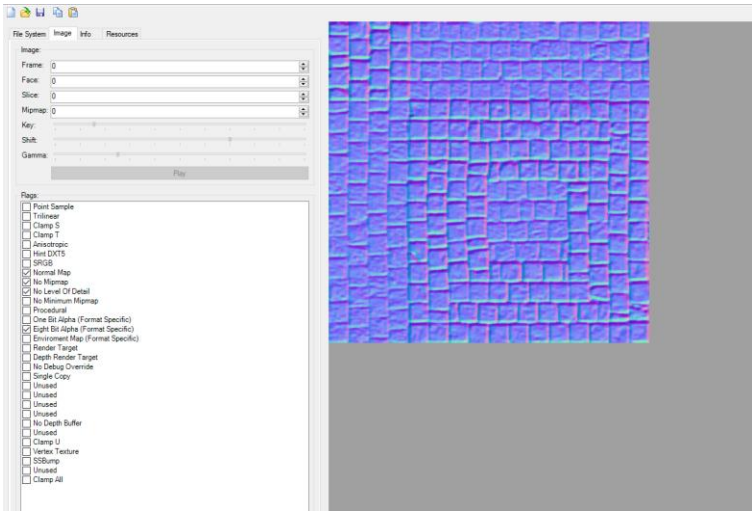


Obrázek 32 – Importovaná textura formátu .png (VTFedit) (vlastní zpracování)

Tento program zároveň umožňuje velmi rychle vytvořit „normální mapy“ pro iluzi 3D povrchů, při importování je vybráno „Generate Normal Map“, je nastavena škála a zbytek program zařídí sám. Výslednou normální mapu lze pak samostatně uložit a odkázat na ni v metadatech textury.

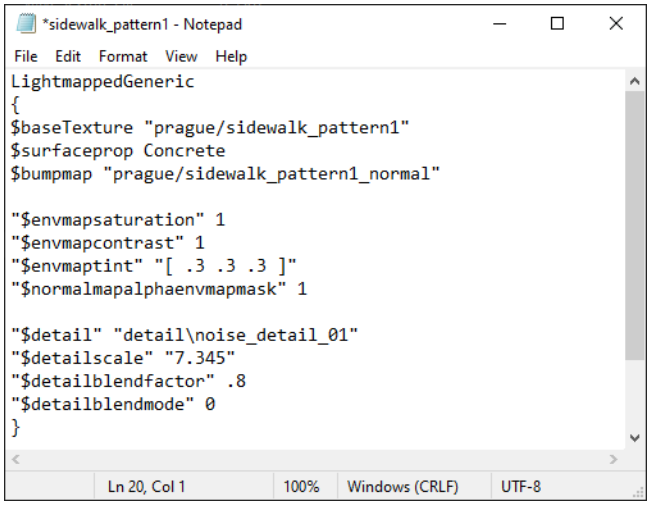


Obrázek 33 – Konverze původního obrázku na normální mapu (VTFedit) (vlastní zpracování)



Obrázek 34 – Výsledná normální mapa při využití škály 6 (VTFedit) (vlastní zpracování)

Nakonec je třeba pro texturu sestavit metadata v podobě VMT souboru. Zde je třeba definovat cestu k VTF souboru, ve vztahu ke kořenové složce „materials“ v adresáři herních souborů. Dále lze nastavit informaci o typu povrchu ($surfaceprop), připojit normální mapu ($bumpmap), povolit a upravit reflektivnost ($envmap...), aplikovat vrstvu opakující se detailní textury ($detail) a další.



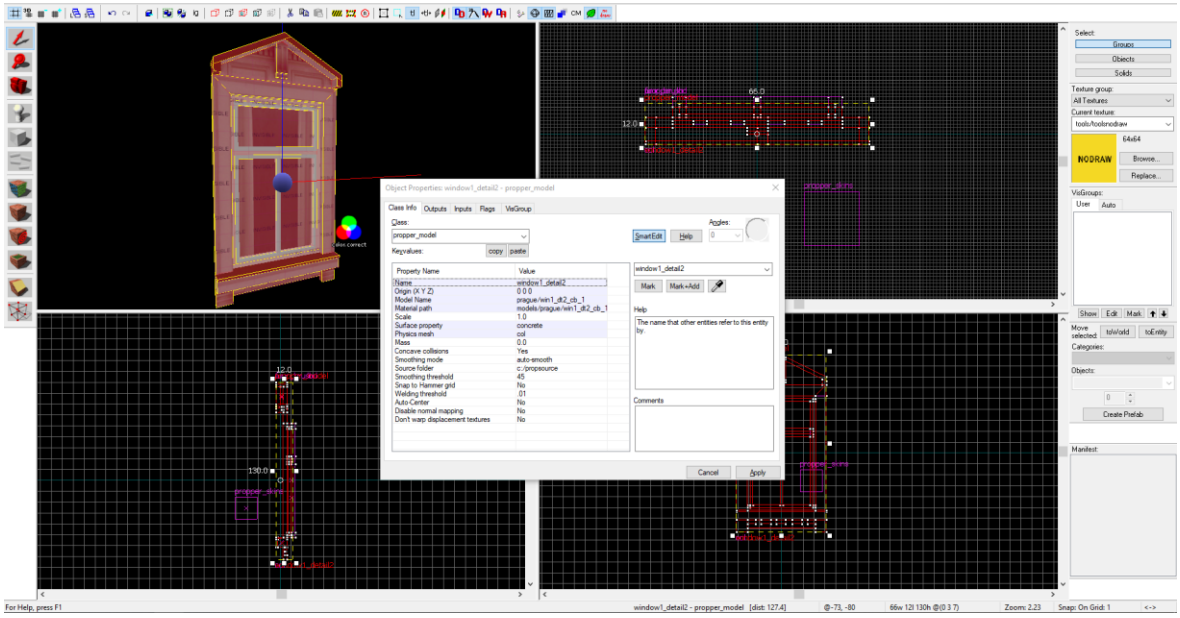
Obrázek 35 – VMT soubor pro texturu chodníku (vlastní zpracování)

1.8.3 Obecný princip tvorby modelů pomocí kompilátoru „Propper“

Jedním ze způsobů tvorby modelů pro Source Engine je modifikovaný kompilátor „Propper“. Díky němu je možné vytvořit model v samotném editoru ze stěn a následně jej zkompilovat na model. Výhodou je, že nevyžadule další specifické schopnosti ovládání externích modelovacích programů. Ovšem hlavní nevýhodou tohoto kompilátoru je velikostní omezení tvorby stěn – stěny jsou tvořeny na mřížce o minimální velikosti 1 Hammer jednotky (2,5 cm).

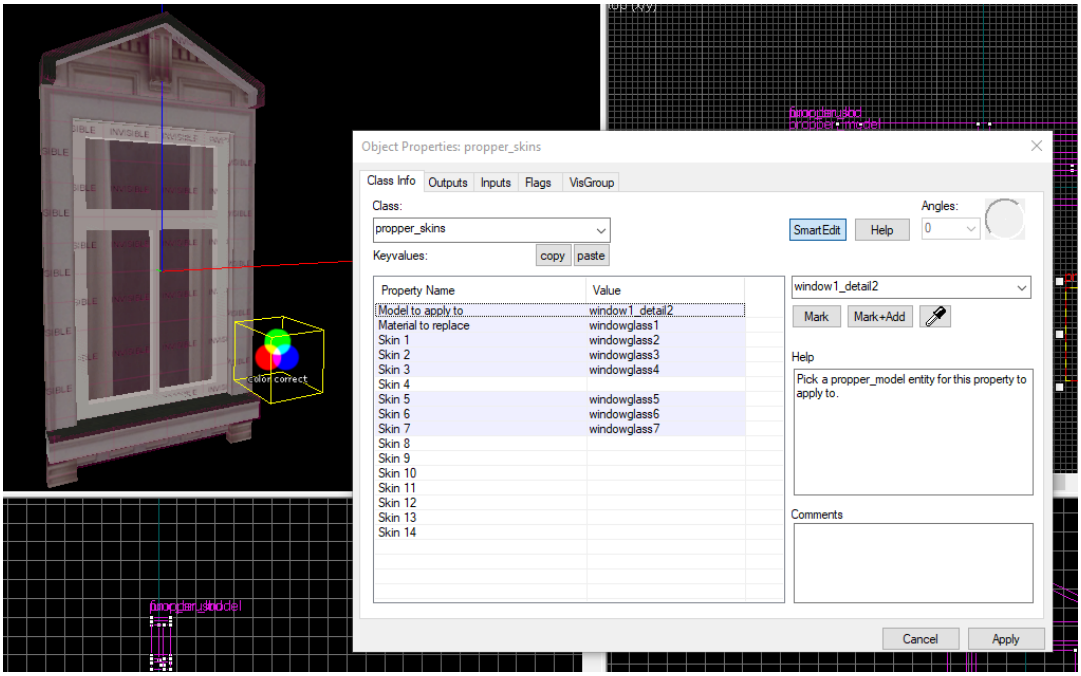
Tuto limitaci lze redukovat tvorbou modelu na vyšší škále a nastavením nižší škály ve vlastnostech entity tvořeného modelu (tedy například model je tvořen 4x větší, škála ve vlastnostech bude 0,25). Zároveň tento kompilátor neumožňuje využití příliš komplexních geometrických tvarů, protože je objekt tvořen v rámci Hammer editoru. Tuto limitaci lze opět zčásti eliminovat zvýšením škály modelu, ovšem z důvodu nepřesnosti jednotlivých bodů komplexních tvarů (například koule) bude po opětovaném načtení uloženého souboru rozpracovaného modelu tvar zkreslen.

Navzdory těmto limitacím ovšem kompilátor Propper poskytuje všechny potřebné funkcionality pro tvorbu jednodušších modelů v Source Engine. Ve vlastnostech je například možné definovat entity reprezentující kolize modelu (kolizní model bývá o mnoho jednodušší), typ povrchu, hmotnost, maximální stupeň, do kterého bude vyhlazováno světlo na záhybech, a další.

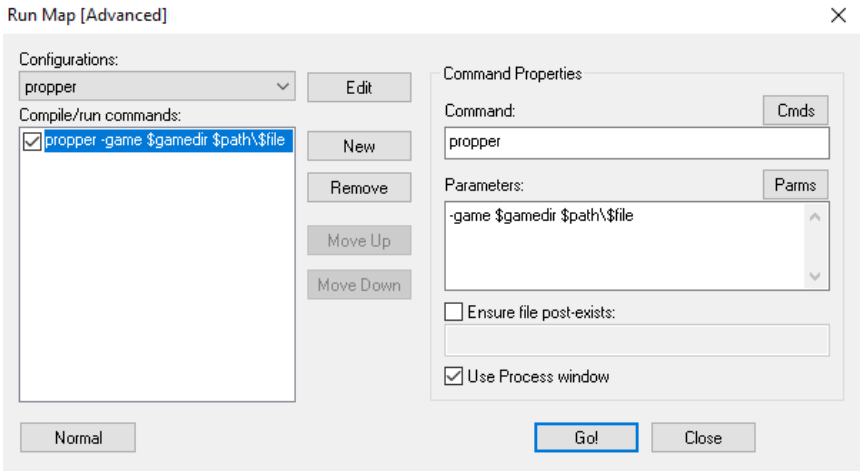


Obrázek 36 – Náhled do vlastností modelu okna před kompilací (vlastní zpracování)

Propper také umožňuje přidělit modelu různé variace textur (dále jen „skin“), v následujícím případě je nahrazována textura se závěsy různými variantami interiérových textur (plně zahrnuté závěsy, žaluzie, ...).



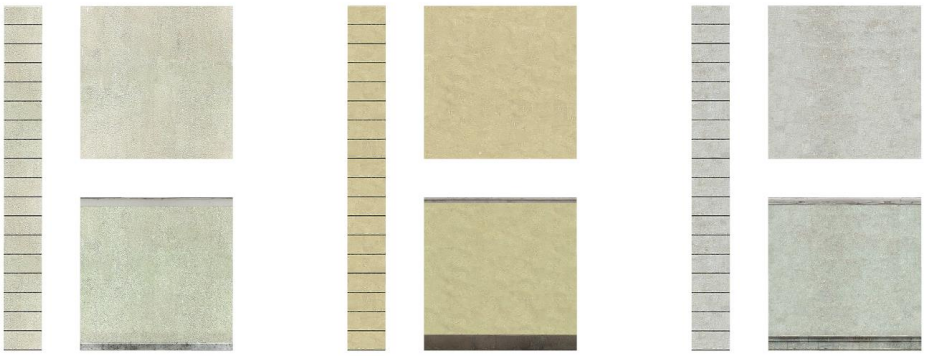
Obrázek 37 – Definování variant textur pro model okna (vlastní zpracování)



Obrázek 38 – Okno zahájení kompilace modelu (vlastní zpracování)

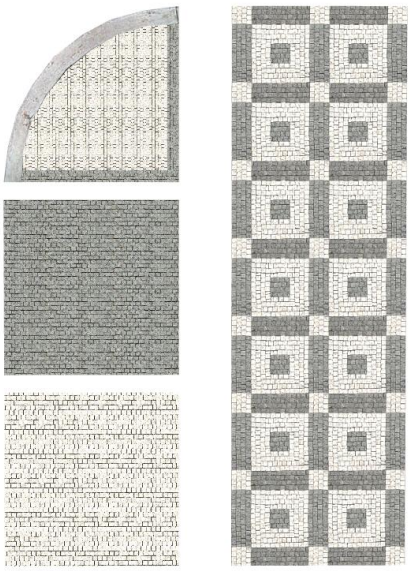
1.8.4 Příprava assetů

Bylo zřejmé, že bude potřeba velké množství vlastních modelů a textur na tvorbu map evropské tématiky. Prvním krokem tak byla tvorba základních stavebních assetů. Na základě již existujících textur poskytovaných hrou byly vytvořeny 3 variace textur venkovních zdí domů, včetně spodní varianty pro každou texturu, která viditelně odděluje zeď od chodníku.



Obrázek 39 – Vytvořené textury stěn (vlastní zpracování)

Dále byly z fotek sestaveny textury dlažeb chodníků spolu s čistě modrou a bílou variantou. Tyto textury byly dále využity na tvorbu modelu záhybu pro rohy chodníků.



Obrázek 40 – Vytvořené textury dlažeb a model záhybu chodníku (vlastní zpracování)

Byly navrženy 4 typy oken a 3 typy okenních ornamentů. Původním plánem bylo tyto varianty kombinovat, čímž by bylo dosaženo velmi mnoho možných kombinací. Jak se ovšem později ukázalo, první mapa kvůli tomuto způsobu kombinace velmi rychle přesáhnula limit 4096 statických modelů – každé okno se skládalo ze 2 modelů (samotné okno a ornament). Bylo tedy třeba vytvořit několik modelů s předem určenými kombinacemi oken a ornamentů, a to spojením těchto variant. Ve výsledku tedy mají okna méně možných variací.

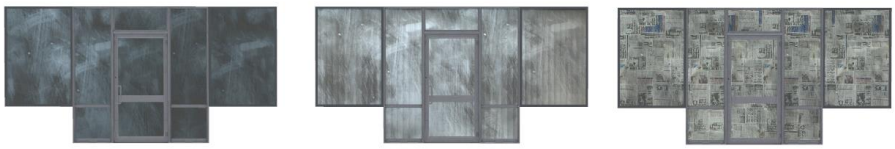
Obrázek 41 – Vytvořené modely oken (vlastní zpracování)

Také byly vytvořeny 3 verze balkónů různé délky, přičemž v každé verzi pak může být využita jedna ze tří textur zdí pro lepší splynutí s budovou (9 možných kombinací). Tyto balkóny byly nakonec použity pouze v první mapě.

Obrázek 42 – Vytvořené modely balkónů (vlastní zpracování)

Pro tvorbu vstupních vchodů do falešných obchodů bylo využito modelů vstupních dveří obchodů již obsažených ve hře, avšak bylo potřeba vytvořit textury reprezentující falešný interiér. Textury reprezentující vnitřek obchodu za sklem byly nejdříve plánovány jako textury vytvořené z fotografií reálných interiérů, nicméně po mnoha testování se ukázalo, že tento typ falešných interiérů není příliš důvěryhodný.

Byly tedy vytvořeny 3 varianty těchto falešných interiérů takovým způsobem, aby hráč nikdy neviděl dovnitř. Těmito variantami jsou zahrnuté závěsy, nalepené noviny a špinavé neprůhledné sklo.



Obrázek 43 – Kombinace modelů vstupních dveří a textur falešných interiérů (vlastní zpracování)

Pro přístupné obchody byly vytvořeny 3 varianty rámu vstupních dveří a 2 typy oken.

Obrázek 44 – Vytvořené modely oken a vstupních rámů obchodů (vlastní zpracování)

Každý obchod také musí mít nějakou formu identifikace – vertikální a horizontální cedule. Bylo vytvořeno 5 typů rámů cedulí a zhruba 16 verzí textur cedulí obchodů pro každou z kategorií: restaurace, večerka, kadeřnictví, tetovací salón, obchod s oblečením, kancelář. Pro restaurace byly vytvořeny modely cedulí jiného rozměru. Jejich názvy byly inspirovanány českými vynálezci, sportovci a literárními autory. Jelikož má každá restaurace také ceduli typu točeného piva, bylo potřeba vytvořit alespoň dvě varianty těchto cedulí pro dvě nejrozšířenější značky piva konzumované v České republice.

Obrázek 45 – Vytvořené identifikační modely obchodů (vlastní zpracování)

Nakonec byla vytvořena množina modelů pro „detailování“ ulic. Konkrétně se jedná o modely, jako pouliční zábradlí, pouliční lampy s variantami pro zapnutou a vypnutou lampu, odpadkové koše a popelnice.

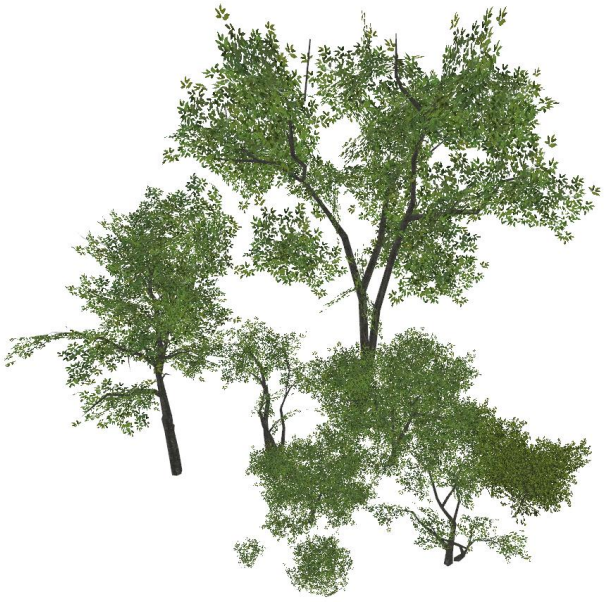
Obrázek 46 – Vytvořené pouliční modely (vlastní zpracování)

1.8.5 Vyhledání komunitních assetů

Velmi důležitou součástí tvořeného světa bylo osazení ulic dopravními prostředky. Komunitní přispěvatel pod přezdívkou „Dinus Saurus“ na stránce https://gamebanana.com/mods/2019 zveřejnil volně použitelné modely aut. Tyto modely byly využity na všech mapách tvořené kampaně.

Obrázek 47 – Komunitní modely aut
(komunitní přispěvatel „Dinus Saurus“ – GameBanana.com)

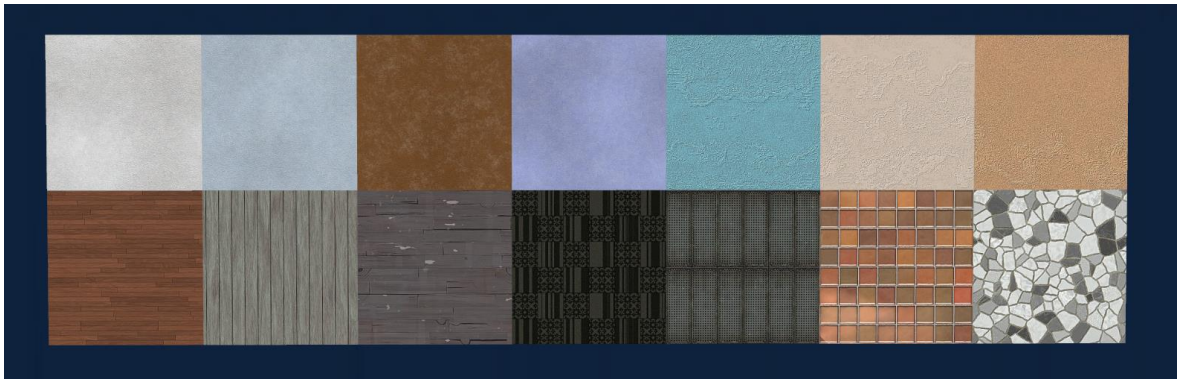
Důležitou částí především pro třetí mapu byly modely zeleně. Na stránce komunitních modelů Gamer-lab.com (https://gamer-lab.com/eng/source/Models) publikoval uživatel pod přezdívkou „Cep}|{„ sadu modelů stromů a keřů, které dobře vizuálně splývaly s prostředím třetí mapy.



Obrázek 48 – Komunitní modely zeleně
(komunitní přispěvatel „Cep}|{“ – Gamer-lab.com)

Zároveň byl nalezen relativně jednoduchý model tramvaje na Steam workshopu hry Garry's Mod (hra stejného enginu), který byl využit. Tento model byl využit jen ve vzdálených nedosažitelných prostorech. Také bylo použito 14ti textur z texturového balíku „Real World Textures 2" vydaného komunitním přispěvatelem pod přezdívkou „TopHattWaffle" na stránkách www.tophattwaffle.com.

Obrázek 49 – Komunitní model tramvaje
(komunitní přispěvatel „terran462" – Steam Workshop hry Garry's Mod")



Obrázek 50 – Použité textury z balíku „Real World Textures 2"
(komunitní přispěvatel „TopHattWaffle" -
https://www.tophattwaffle.com/downloads/realworldtextures-2/)

Dále byly vyhledány příslušné komunitní textury obloh. Konkrétně bylo využito oranžové večerní oblohy pro třetí mapu (https://gamebanana.com/mods/6836) a tmavě modré noční oblohy pro čtvrtou mapu (https://gamebanana.com/mods/366901).

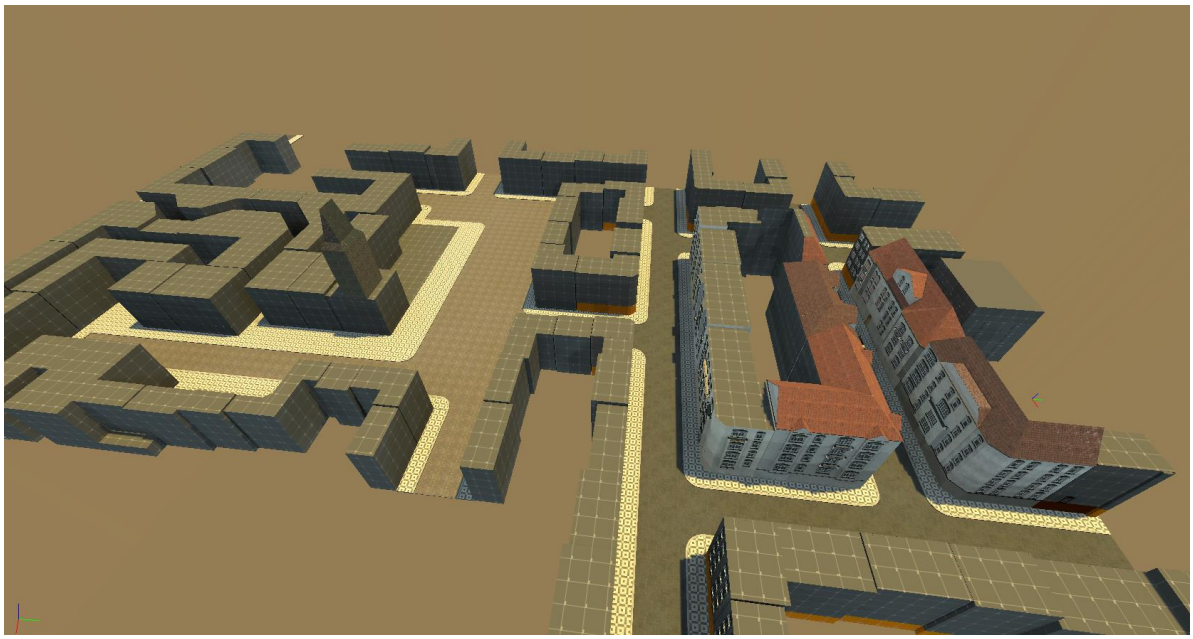
Komunitních assetů bylo využito převážně z důvodu nedostatečné zkušenosti s pokročilejšími modelovacími programy a potřebě urychlení vývoje této velmi časově náročné kampaně. Výsledná kampaň není nijak zpoplatněná a negeneruje žádný peněžní příjem, tvorba této kampaně slouží především ke vzdělávacím účelům.

1.9 Mapa 1 – centrum

1.9.1 Tvorba hrubého obrysu světa

Počáteční aktivitou při vytváření této mapy byla tvorba startovního bytu, jelikož nebylo jisté, kolik místa bude nutné dedikovat a kolik prostoru bude potřeba rezervovat na interiér bytového domu. Po vytvoření hrubého interiéru bytu a domu byla tato část obalena stěnami.

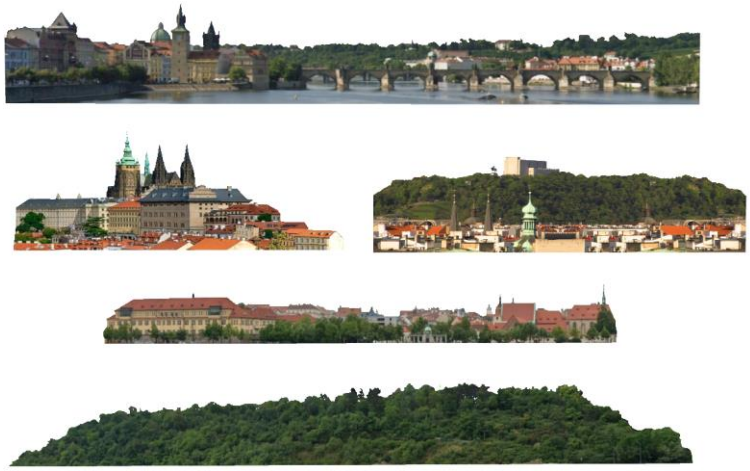
Po zjištění přibližné velikosti jednoho bytového domu, bylo možné relativně rychle poskládat ulice a vyhradit prostory pro ostatní bytové domy. Problémem ovšem bylo, že každý z těchto vyhrazených prostorů pro bytové domy měl různou výšku, šířku a délku. Nebyla tedy jiná možnost než všechny budovy skládat manuálně bez kopírování – tento proces byl velmi časově náročný.



Obrázek 51 – Brzká fáze tvorby hrubého obrysu světa – mapa 1 (vlastní zpracování)

1.9.2 3D skybox

Pro tvorbu 3D skyboxu bylo potřeba textur panoramatických obrázků Prahy s transparentním nebem a 16x zmenšené modely budov. Panoramatické snímky byly získány pomocí Google Street View, snímky pak byly následně narovnány a zpracovány do použitelných textur.

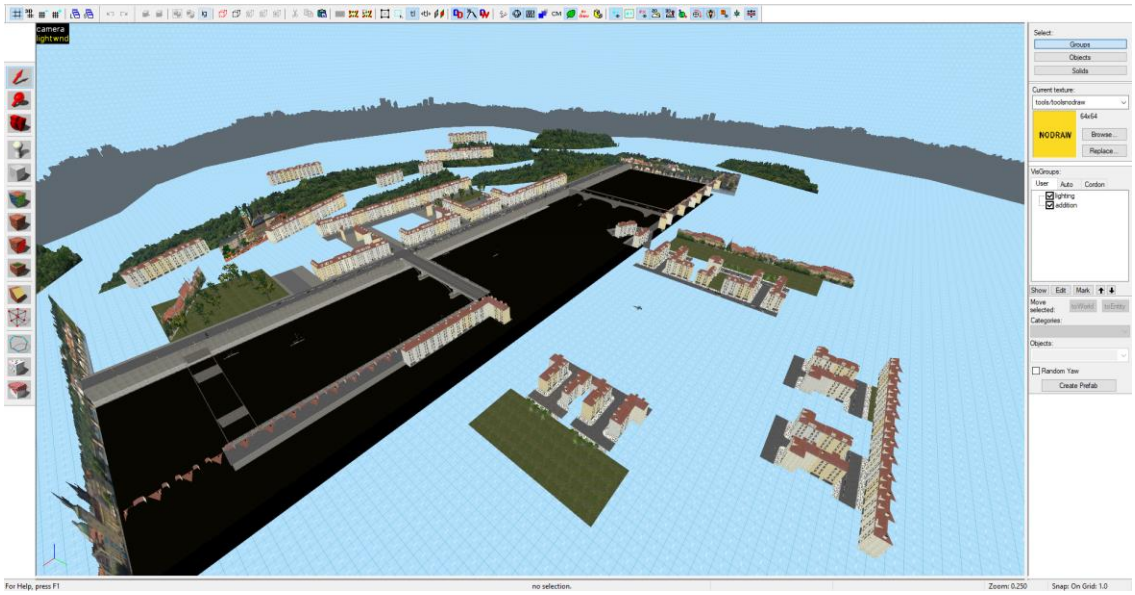


Obrázek 52 – Výsledné panoramatické textury pro 3D skybox – mapa 1 (vlastní zpracování)

Dále byly vytvořeny velmi zjednodušené modely budov na základě již vytvořených budov v této mapě.

Obrázek 53 – Vytvořené modely budov pro 3D skybox – mapa 1 (vlastní zpracování)

3D skybox byl nejdříve tvořen jako součást samotného hratelného prostoru. Poté byly části které patří do 3D skyboxu odříznuty a spolu s entitou „sky_camera“ byl výběr 16x zmenšen pomocí transformačního nástroje. Tento výběr byl pak obalen nebem a panoramatickými texturami na pozadí.



Obrázek 54 – Výsledný 3D skybox v Hammer++ (mapa 1) (vlastní zpracování)

V hratelné části je tento prostor vykreslován ve standardní velikosti s podstatně nižším nárokem na výkon, přičemž hráč má iluzi, že svět pokračuje mimo hratelnou zónu.

Obrázek 55 – Iluze okolního města projekcí 3D skyboxu (vlastní zpracování)

1.9.3 Tvorba potřebných assetů

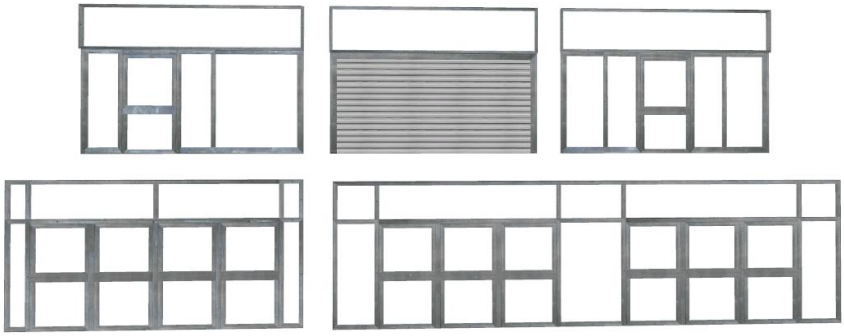
Konkrétně pro tuto mapu bylo potřeba vytvořit relativně velké množství vlastního obsahu. Zejména pouliční cedule, modely na sestavení věže Pražského orloje a textury/modely pro „detailování“ stanice metra. Potřebné fotky, na základě kterých byl tento obsah tvořen, byly zprostředkovány převážně pomocí Google Street View.

Obrázek 56 – Vytvořené varianty pouličních cedulí (vlastní zpracování)

Obrázek 57 – Vytvořené modely využité pro konstrukci Orlojské věže (vlastní zpracování)

Obrázek 58 – Kombinace vytvořených modelů pro vstup do stanice metra (vlastní zpracování)

Obrázek 59 – Vytvořené textury podlahy a zdi stanice metra (vlastní zpracování)



Obrázek 60 – Vytvořené modely vstupních dveří do falešných obchodů stanice metra (vlastní zpracování)

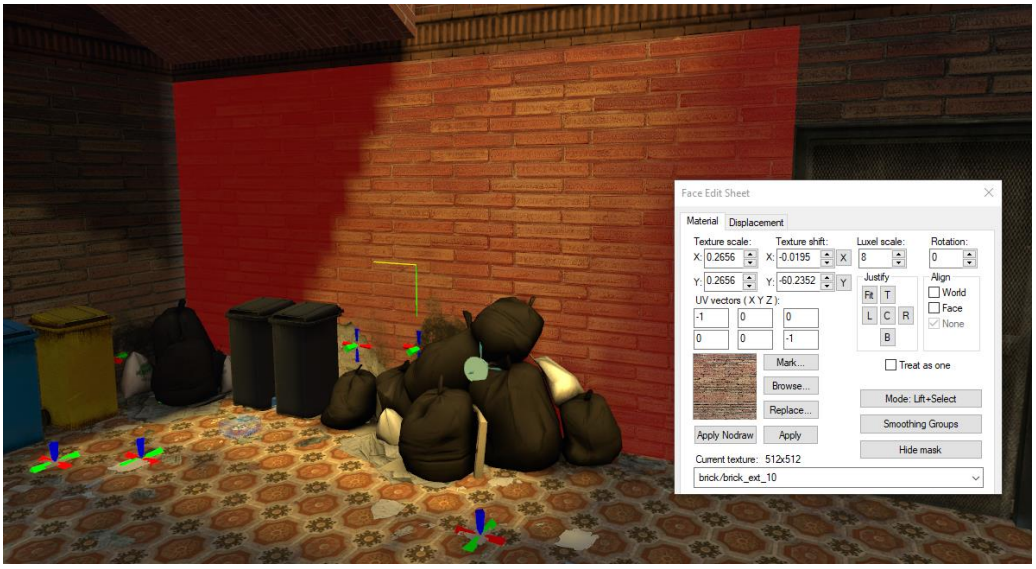
Obrázek 61 – Vytvořené modely pro detailování stanice metra (vlastní zpracování)

1.9.4 Obecný princip detailování

Základním krokem, jak lze oblasti obohatit větším detailem, je volba příslušných textur. Výběr vhodných textur pro určitou lokaci ovlivňuje jak světelné odrazy, tak i barevný styl. Tento krok velmi zásadně určuje, jak detailovaná oblast vypadá.

Obrázek 62 – Prohlížeč textur v Hammer++ (vlastní zpracování)

Nástroj pro manipulaci textur umožňuje nastavení posunu a škály textury, škály světelných map, rotace a další.

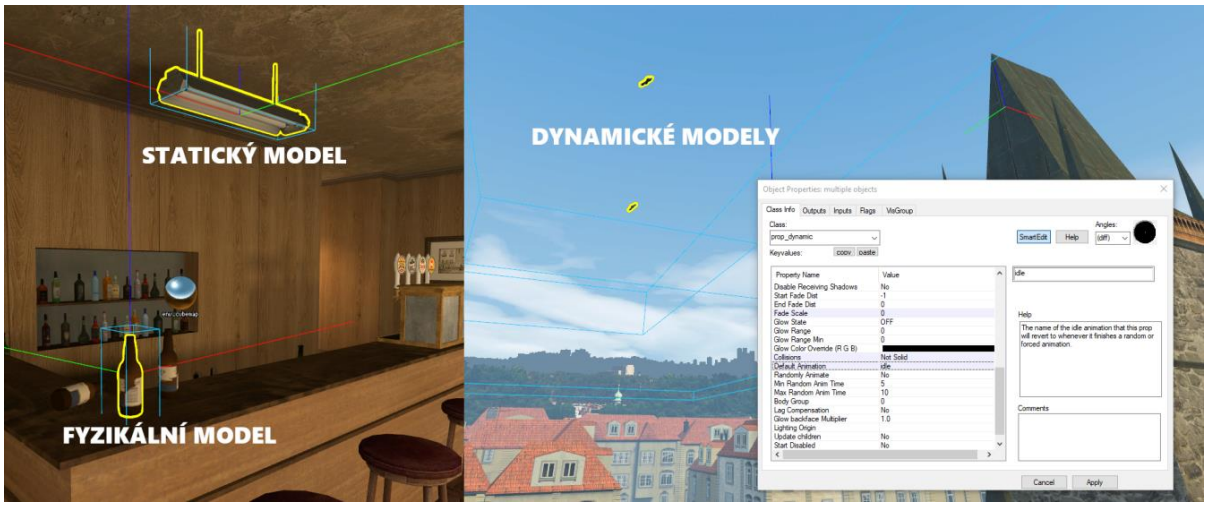


Obrázek 63 – Aplikace textur v Hammer++ (vlastní zpracování)

Hlavní náplní fáze detailování je rozmístění různých typů modelů. Většina map se převážně skládá ze statických modelů – v jedné mapě jich totiž může být až 4096. Jelikož se statické modely ve hře nepohybují, je do nich možné „zapéct“ lepší osvětlení a také tyto modely produkují kvalitnější stíny.

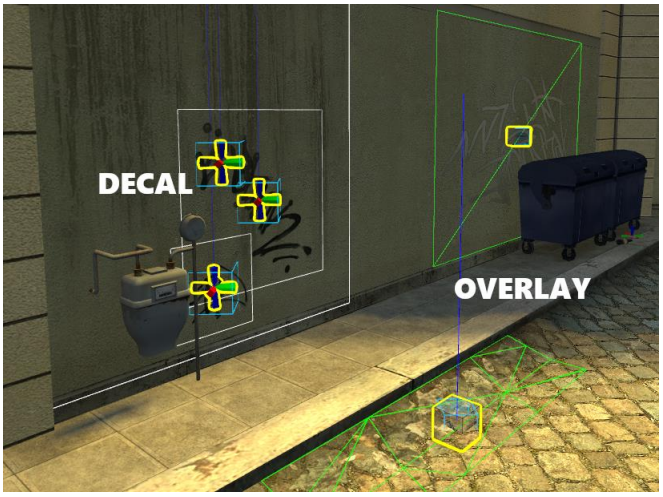
Specifické modely ovšem také reagují na chování hráče a je pro ně simulována fyzika. Zároveň je také často možné tyto modely ve hře rozbít na měnší části. Pro tento účel slouží fyzikální modely. Jelikož se jedná o dynamické entity, jejich počet se započítává do dynamického limitu Source Engine (2048) a je tedy potřeba použití těchto modelů omezit na minimum.

Posledním typem jsou modely dynamické, které nejčastěji slouží pro vykreslení modelu s animací. Tyto modely lze manipulovat pomocí herní logiky a hráči s nimi opět mohou různě interagovat. Stejně jako ty fyzikální, i dynamické modely se započítávají do limitu dynamických entity a jejich osvětlení a stíny jsou méně přesné než u statických modelů.



Obrázek 64 – Typy modelů a jejich příklady v Hammer++ (vlastní zpracování)

Pro přerušení monotónosti opakujících se textur slouží výhradně „nálepky“, neboli v případě Source Enginu jsou to „overlays“ a „decals“. Overlays lze různě tvarovat, ovšem nelze jimi manipulovat ve hře pomocí herní logiky. Decals mají naopak konstantní předem definovanou velikost, ale lze jimi manipulovat při samotném hraní. Zároveň je limit overlays mnohem nižší než u decals, je tedy vhodné overlays využívat jen tam, kde je potřeba specifické natvarování (rotace, roztažení).



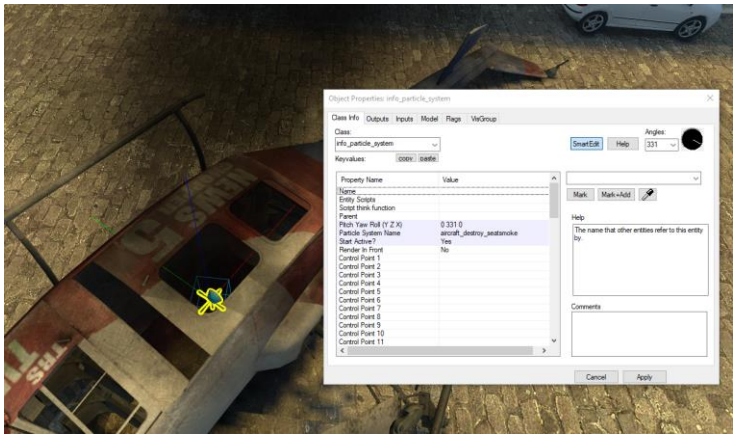
Obrázek 65 – Příklady infodecal a info_overlay v Hammer++ (vlastní zpracování)

V rámci vertikálního detailu je často využíváno provazů. Source Engine podporuje téměř plně dynamické provazy se simulovanou fyzikou, ovšem každý záchytný bod provazu je započítáván do limitu dynamických entit a je tedy třeba jimi šetřit.

Entity provazu (keyframe_rope a move_rope) zahrnují vlastnosti jako počet záhybů, texturu, tloušťku a další.

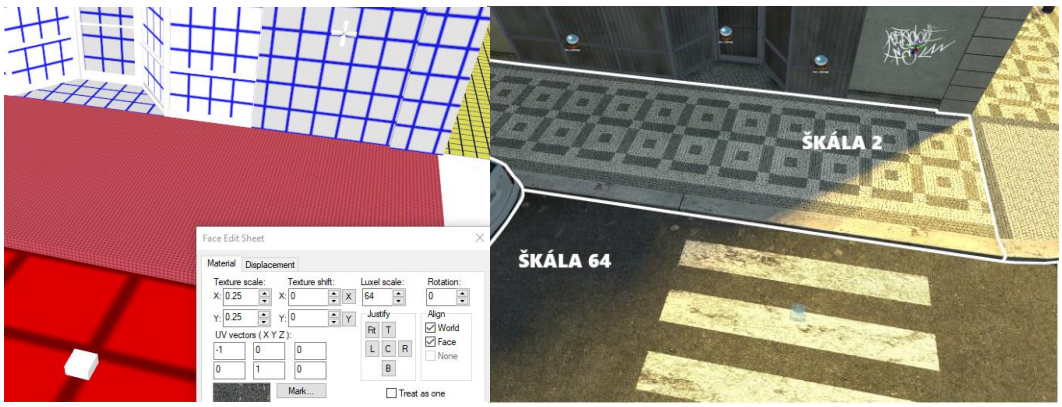
Obrázek 66 – Příklad využití provazů v Hammer++ (vlastní zpracování)

V určitých lokacích je vhodné využít částicových efektů, jako například kouře či poletujícího hmyzu, a pro tento účet slouží entita info_particle_system. V rámci této entity lze definovat, jaký částicový efekt má entita využívat, lze entitě přiřadit jméno a později je s ní možné manipulovat ve hře pomocí herní logiky. Tyto entity ovšem velmi rychle zaplňují limit dynamických entit a je třeba jejich použití omezit.



Obrázek 67 – Příklad využití částicového efektu v Hammer++ (vlastní zpracování)

Jelikož je osvětlení do map předem „zapékáno“ při kompilaci, je vhodné definovat úroveň detailu osvětlení pro různé lokace. Například interiérové lokace velmi benefitují z nízké škály světelných map (detailnější osvětlení a stíny), zatímco venkovní, vzdálené a nepřístupné prostory stačí pokrýt méně detailním osvětlením (vyšší škála světelných map).



Obrázek 68 – Nastavení škály světelných map a rozdíl v rozlišení (vlastní zpracování)

1.9.5 Detailování

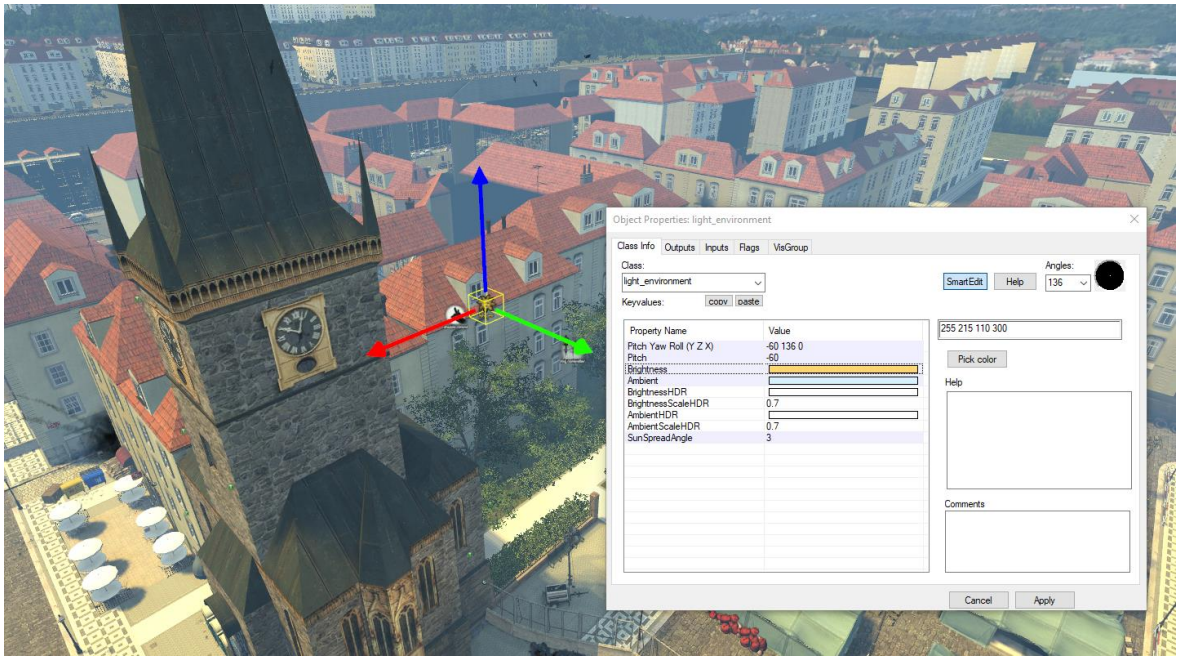
Výsledky detailování jednotlivých lokací viz příloha č. 1 sekce „Mapa 1“.

1.9.6 Osvětlení

Původním plánem bylo, aby se tato mapa odehrávala v pravé poledne. Rychle se ovšem ukázalo, že toto osvětlení není příliš vizuálně zajímavé, jelikož budovy nehází téměř žádné stíny. Sklon slunce byl tedy nastaven na -60° (-90° je pravé poledne).

Barva a jas slunce – neboli „Brightness“ v nastavení light_environment byla nastavena na světle oranžovou s relativně nízkým jasem. V samotné hře bývají světla méně barevná a zhruba 3x světlejší než v náhledu editoru. Je tedy potřeba vybírat výraznější barvy a nastavovat nižší jas.

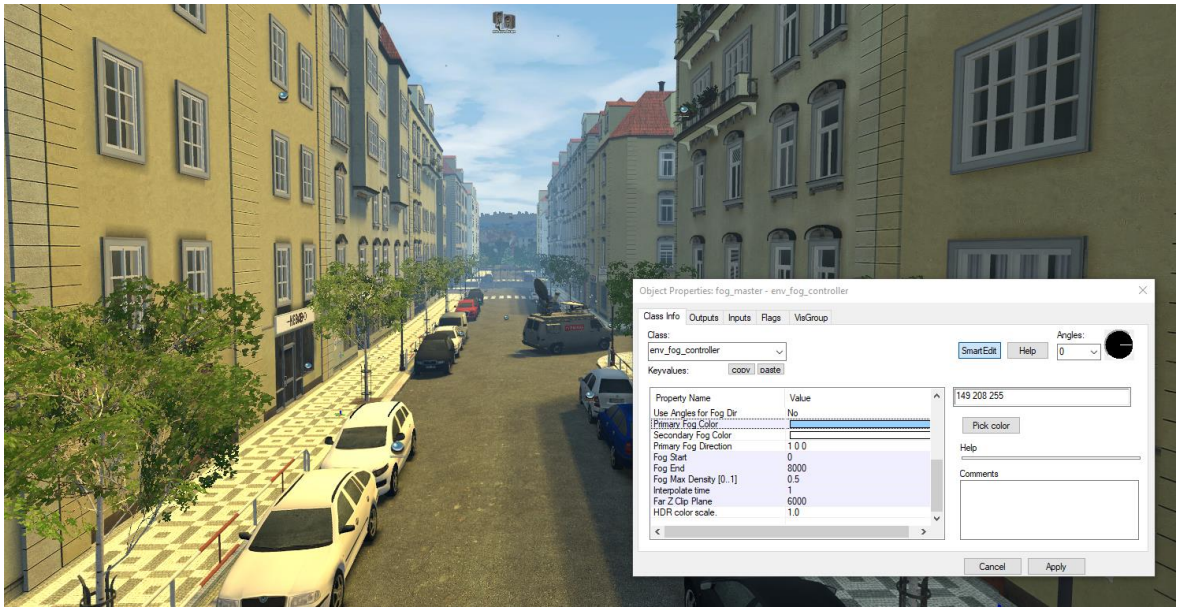
Položka „Ambient“ v nastavení light_environment určuje barvu a jas stínů neboli venkovního světla mimo přímý dopad slunce. Zde byla zvolena světle modrá barva.



Obrázek 69 – Vlastnosti entity light_environment – mapa 1 (vlastní zpracování)

1.9.7 Mlha

Tato mapa obsahuje 2 typy mlhy, venkovní a interiérovou. Venkovní mlha je nastavena na stejný odstín modré barvy, jako je nebe. Díky tomuto nastavení vzdálené objekty obalené mlhou s oblohou lehce splývají. Maximální plnost mlhy je ovšem nastavena jen na 50%, jinak by vzdálené objekty úplně zmizely, což pro aktuální počasí není realistické.



Obrázek 70 – Nastavení venkovní mlhy – mapa 1 (vlastní zpracování)

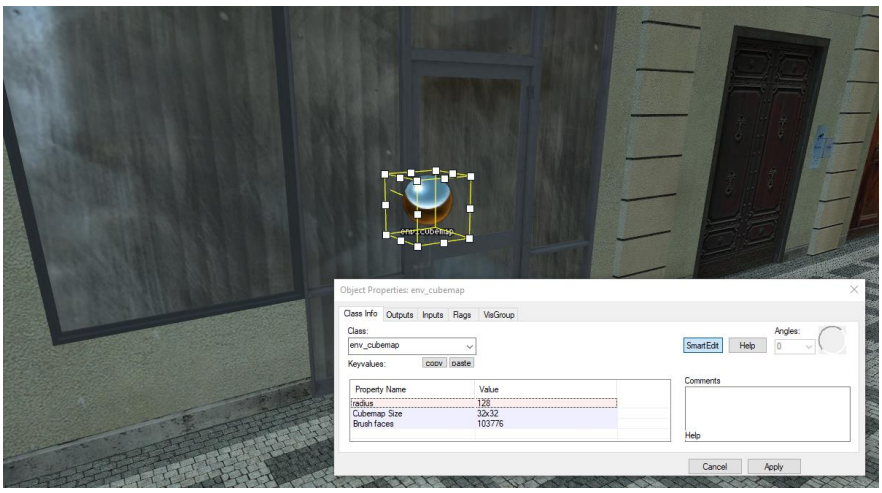
Interiérová mlha je naopak oranžová a vzdálenost, kde nabude svůj limit plnosti, je o polovinu kratší než u venkovní mlhy. Venkovní část je tedy při pohledu ven z interiérů viditelně oddělena a vnitřek interiérů není pokryt modrou mlhou. Tento rozdíl se může zdát jako zanedbatelný a nelze dostatečně posoudit z obrázku, při hraní je však tento efekt mnohem výraznější.

Obrázek 71 – Nastavení interiérové mlhy – mapa 1 (vlastní zpracování)

1.9.8 Odrazy

Pro generování odrazů bylo potřeba rozmístit entity env_cubemap. Zadáním příkazu „buildcubemaps“ do herní konzole je možné vygenerovat environmentální mapy (textury 360° pohledu) na místě každé entity env_cubemap. Tyto environmentální mapy jsou buď automaticky aplikovány na nejbližší reflektivní povrch, nebo pro ně lze zvolený povrch manuálně definovat.

Při tvorbě této mapy bylo potřeba entity rozmístit převážně před okna bytových domů, vstupy obchodů a kolem modelů dopravních prostředků. Testování generovaných reflexí bylo vždy možné provést až ve hře. Po opravení nalezených chyb bylo tedy vždy potřeba mapu znovu kompilovat a opět vygenerovat nové environmentální mapy. Z tohoto důvodu byl tento proces relativně časově náročný.



Obrázek 72 – Příklad využití entity env_cubemap (vlastní zpracování)

1.9.9 Zvuková kulisa

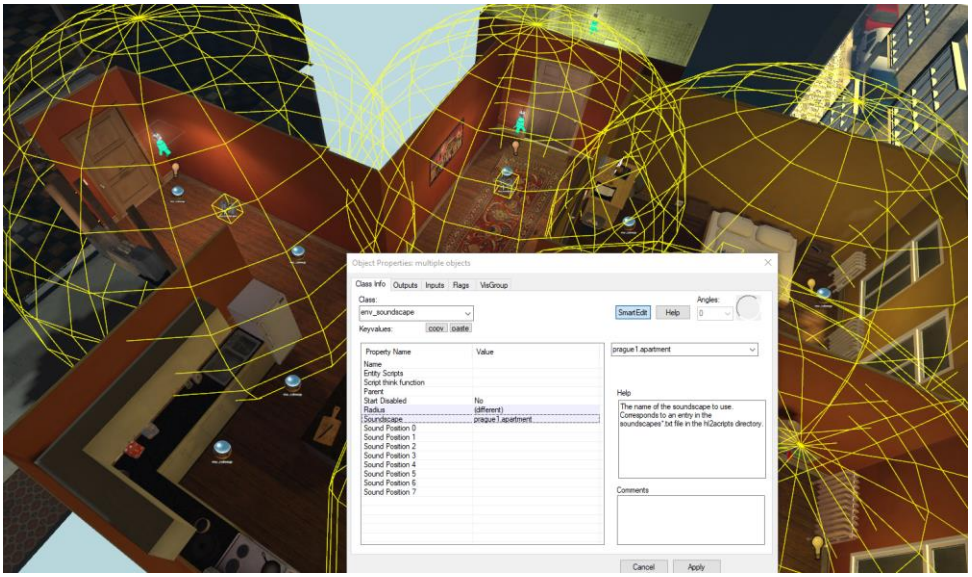
Pro definici zvukové kulisy bylo využito externího textového souboru, ve kterém bylo podle příslušných pravidel definováno několik typů zvukové kulisy, konkrétně: byt, bytový dům, koupelna, ulice, obchod s elektronikou, bar, večerka 1, náměstí, restaurace, úzké ulice, večerka 2, hlavní silnice podél řeky, stanice metra a místnost pro změnu mapy.

Pro každou kulisu byl zvolen opakující se zvuk (vyznačováno jako „playlooping“) a výběr z náhodných zvuků opakující se v definovaném časovém intervalu. Dále byly vytvořeny obecné vzory zvukových kulis a tyto vzory byly později vnořeny do kulis pro konkrétní lokaci. Například do zvukové kulisy pro ulice byla vnořena kulisa obecného chaosu obsahující vzdálené exploze, policejní sirény, přelety helikoptér a další.

```
"prague1.streets"
{
    "dsp" "1"


    "playlooping"
    {
        "volume"    "1"
        "pitch"     "100"
        "wave"      "ambient\ambience\crucial_empty_street_wind_loop.wav"
    }

    "playrandom"
    {
        "time"      "8,14"
        "volume"    "1"
        "pitch"     "90,100"
        "position"  "random"

        "rndwave"
        {
            "wave"  "ambient\random_amb_sfx\randomwindgust01.wav"
            "wave"  "ambient\random_amb_sfx\randomwindgust02.wav"
            "wave"  "ambient\random_amb_sfx\randomwindgust03.wav"
        }
    }

    "playsoundscape"
    {

        "name"          "prague.general_chaos"
        "volume"        "1"
    }

}
```

Obrázek 73 – Část souboru prague_1_old_town_soundscapes.txt (vlastní zpracování)

Dále bylo potřeba rozmístit entity env_soundscape, ve kterých byl zvolen rozsah a definován název příslušné kulisy (dříve definované v externím textovém souboru). Pokrytí nemuselo býti nastaveno perfektně, jelikož zvukové kulisy nepřestávají hrát po opuštění rozsahu entity. Pouze se změní, pokud hráč narazí na rozsah entity s kulisou jiného názvu.

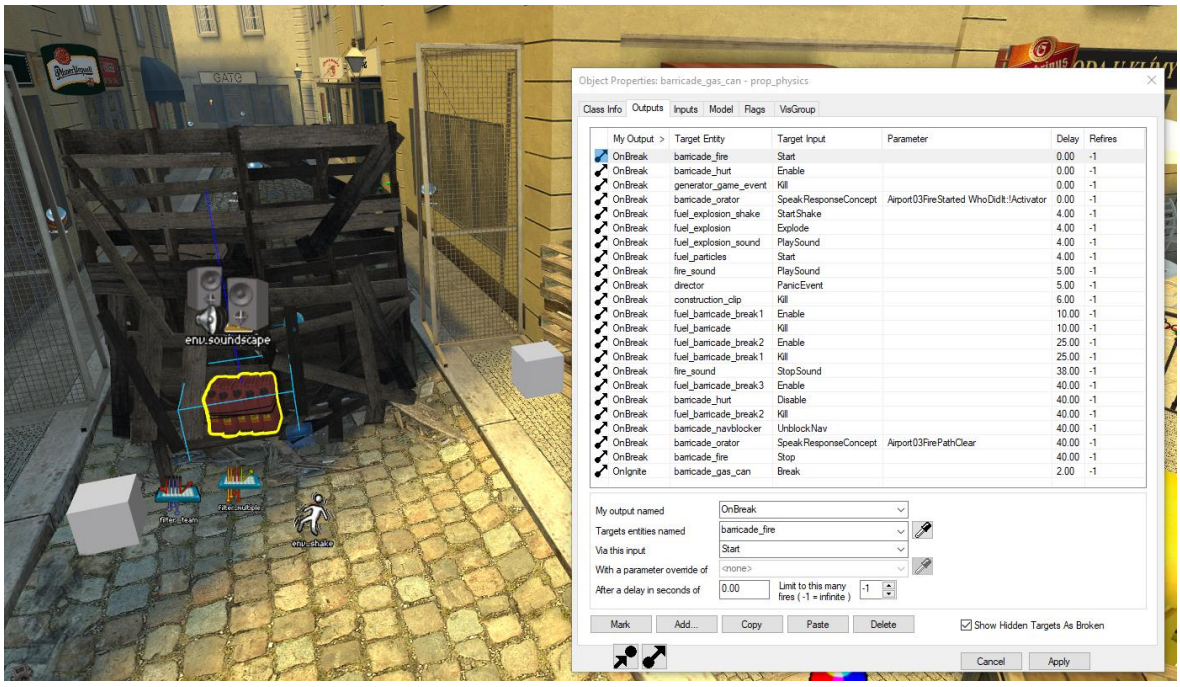


Obrázek 74 – Implementace zvukové kulisy v Hammer++ (vlastní zpracování)

1.9.10 Herní logika

Dynamické měnění herního světa bylo prováděno pomocí vstupů a výstupů relé a potřebných entit. Testování nastavených sekvencí vstupů a výstupů bylo opět možné testovat až po kompilaci ve hře, v této fázi ovšem nezáleželo na vizuální kvalitě kompilovaného světa, jelikož byla testována jen funkcionalita. Bylo tedy možné při testování a opravování chyb kompilovat mapu s horším osvětlením a optimalizací, díky čemuž kompilace nevyžadovala tolik času.

Konkrétně bylo potřeba nastavit: počáteční ukázku, průlet stíhacího letadla kolem orlojské věže po vstupu na náměstí, panický event způsobený hořící dřevěnou barikádou a místnost pro změnu mapy. Tyto sekvence vstupů a výstupů se někdy ukázaly býti složitější, než by se dalo očekávat. Například do výstupů pro pálení barikády bylo potřeba zahrnout zapínání a vypínání modelů tří variant barikády v různém stavu spálení. Také do nich bylo zahrnuto dočasné zablokování navigace umělé inteligence, zvuky a částicové efekty exploze a mnoho dalších výstupů.



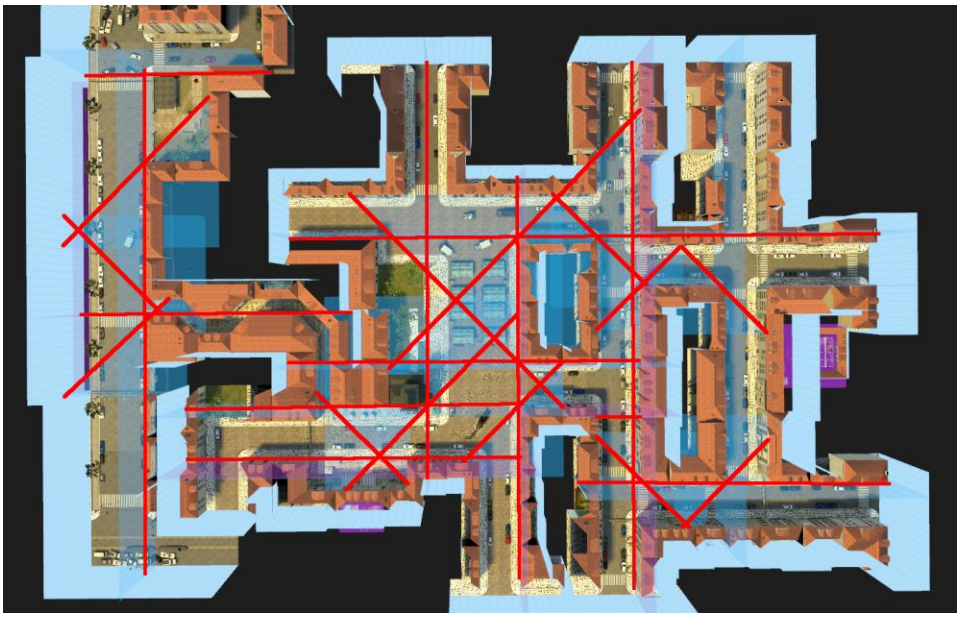
Obrázek 75 – Definovaný výstup na fyzikálním modelu způsobující shoření barikády (vlastní zpracování)

Pro funkcionalitu změny mapy stačilo pouze místnost pokrýt kvádrem, definovat ho jako trigger_changelevel, položit a pojmenovat entitu info_landmark a vložit její název do atributu kvádru. Díky entitě info_landmark bylo nastaveno zanechání pozice hráčů a okolních zbraní/vybavení po změně mapy.

Obrázek 76 – Oblast pro změnu mapy v koncové místnosti (vlastní zpracování)

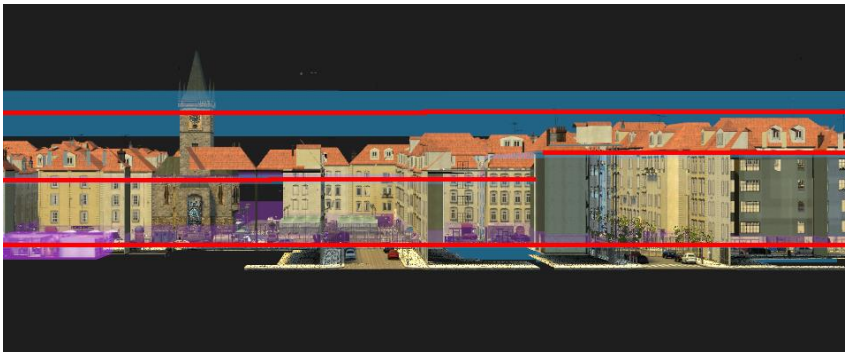
1.9.11 Optimalizace

Tato mapa vyžadovala velký důraz na optimalizaci, jelikož jednotlivé sekce mapy nebyly příliš dobře separovány. Visleafs této mapy bylo tedy potřeba na téměř každém rohu rozseknout pod 45° úhlem pohledem ze shora.



Obrázek 77 – Sekání visleafs pohledem ze shora – mapa 1 (vlastní zpracování)

Zároveň bylo potřeba visleafs celé mapy také trojitě proseknout ze strany, aby rozsekané visleafs nebyly kvůli jejich značné výšce ze shora navzájem viditelné. Z důvodu velkého množství sekání visleafs narostl počet visleafs na zhruba 11 000. Časová náročnost kompilace mapy se tedy exponenciálně zvýšila, protože bylo pro každou visleaf zjišťováno, které další visleafs je z ní možné vidět.



Obrázek 78 – Sekání visleafs pohledem ze strany – mapa 1 (vlastní zpracování)

Dalším pokusem o větší segmentaci mapy bylo definování areaportálů. Převážně jich bylo třeba na oddělení Staroměstského náměstí a bloku ulic se startovním bytovým domem. Nakonec byly areaportály také aplikovány na interiéry, i když tento krok neměl příliš výrazný dopad na výkon.

Obrázek 79 – Hratelná část mapy s vyznačenými lokacemi areaportálů (vlastní zpracování)

1.9.12 Navigace umělé inteligence

Vytvoření navigační sítě pro umělou inteligenci proběhlo kompletně ve hře. Pomocí konzolových příkazů „nav_edit 1“ byl zapnut režim úprav navigační sítě, pomocí příkazů „nav_mark_walkable“ a „nav_generate_incremental“ byly postupně generovány části sítě a pomocí příkazu „nav_analyze“ proběhla herní analýza vygenerované navigace.

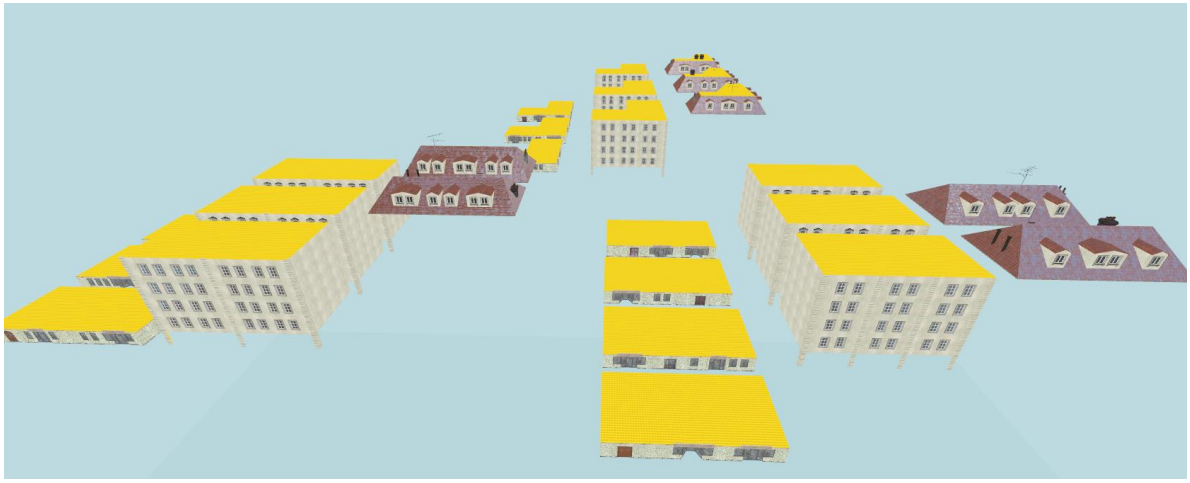
Dále bylo potřeba manuálně dotvořit navigaci v místech, kde automatická generace selhala (převážně na autech, zábradlích a jiných vyvýšených objektech). Tyto manuálně vytvořené navigační plochy bylo později potřeba spojit s existujícími plochami. Zároveň bylo potřeba manuálně označit plochy, na kterých se nemají objevovat nepřátelé (opět na autech, stolech, zábradlích), a také definovat startovní lokaci a místnost změny mapy. Celý tento proces byl velmi časově náročný.

Obrázek 80 – Navigace umělé inteligence na hlavní silnici podél řeky (vlastní zpracování)

1.10 Mapa 2 – město

1.10.1 Hrubý obrys světa

Po velmi náročné tvorbě první mapy bylo potřeba standardizovat skládání bytových domů. Byly navrženy dva podlouhlé domy různé šířky a výšky a jeden rohový dům. Následně bylo pro každý z těchto tří typů vytvořeno několik variací spodní části, prostřední části a střechy. Zároveň bylo možné velmi jednoduše změnit texturu složené budovy na jednu ze tří typů zdí. Tímto kombinováním bylo tedy možné složit velmi mnoho různých kombinací budov s minimálním úsilím.



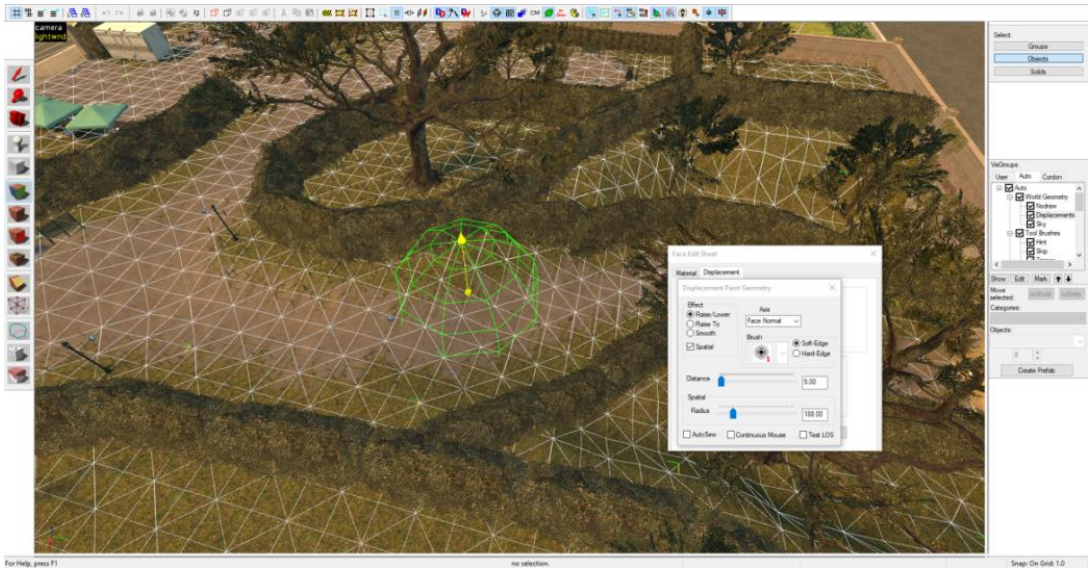
Obrázek 81 – Stavební segmenty 3 typů budov (vlastní zpracování)

Tvorba hrubého obrysu začala pokračováním metra, kde hráči skončili v předešlé mapě. Jelikož byla již část metra sestavená v předešlé mapě, nebylo náročné vytvořit druhou navazující část. Například zablokovaný vstup do spodní části metra byl téměř celý zkopírovaný z předešlé mapy.

Jelikož nyní již bylo zřejmé, kolik místa budou zabírat jednotlivé domy, následovala fáze skládání jednotlivých ulic. Zaplnění ulic bylo nyní mnohem jednodušší, avšak nevýhodou této standardizace bylo, že mezi domy někdy zbyl příliš malý prostor, do kterého by se žádný z typů domů nevešel. Tyto prostory tedy bylo potřeba manuálně vyplnit.

Obrázek 82 – Manuálně vyplněná mezera mezi domy (vlastní zpracování)

Zároveň byla pomocí „displacements“ natvarována travnatá podlaha části parku, která se postupně snižuje a hráče zavede do podchodu pod silnicí, kde se nachází přechod do další mapy.



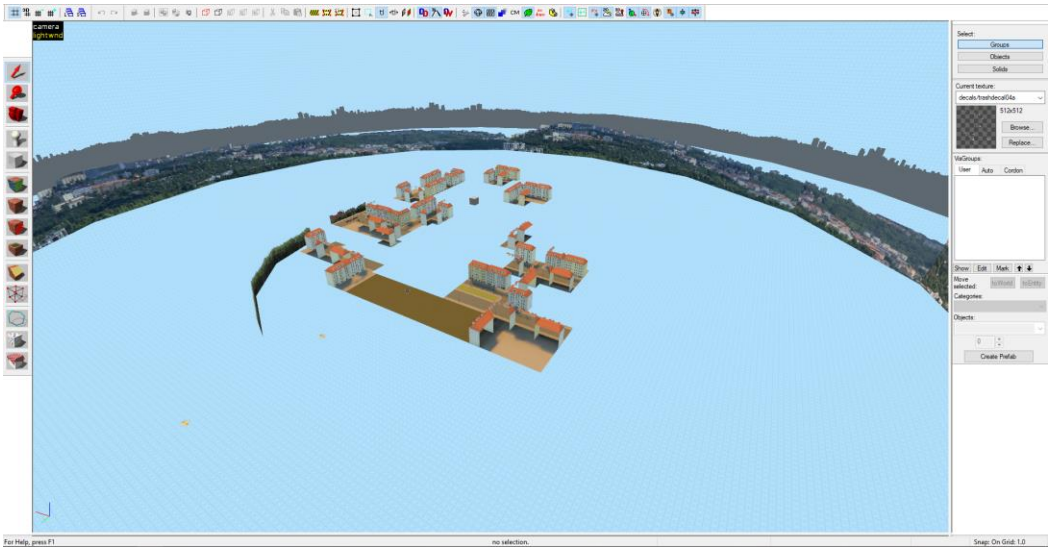
Obrázek 83 – Tvarování terénu části parku (vlastní zpracování)

1.10.2 3D skybox

Jelikož tato mapa obsahuje budovy jiného typu než v předešlé mapě, bylo opět potřeba vytvořit zmenšené modely bytových domů pro 3D skybox.

Obrázek 84 – Vytvořené modely bytových domů pro 3D skybox – mapa 2 (vlastní zpracování)

Po náročné optimalizaci první mapy bylo zřejmé, že je nezbytné, aby půdorys obsahoval co nejméně spojujících se rozdílných cest. Ovšem aby prostředí pro hráče dávalo smysl, bylo nutné, aby měl hráč alespoň iluzi, že vidí do ulic viditelných z různých částí mapy. Tyto spojující ulice byly tedy také přeneseny do 3D skyboxu. Hráči se pak nevykreslují ulice, do kterých nevidí, jelikož byly díky tomuto přístupu části mapy lépe separovány.



Obrázek 85 – Výsledný 3D skybox v Hammer++ (mapa 2) (vlastní zpracování)

Obrázek 86 – Iluze pokračování města za hranicemi mapy projekcí 3D skyboxu (mapa 2) (vlastní zpracování)

1.10.3 Tvorba potřebných assetů

Pro tuto mapu bylo hlavně potřeba vytvořit přechodový materiál trávníku a chodníku s atributem generování trávníkových spritů. Toto bylo provedeno nejprve extrakcí jednotlivých textur trávníku, chodníku a modulační textury přechodu betonu již poskytovaných hrou. Dále byl vytvořen .vmt soubor na definici textury, ve kterém byly definovány textury přechodu a modulační textura, která určuje vzor na přechodu textur.

Poté byl definován typ materiálu na „dirt“ (tato možnost ovlivňuje typ zvuků kroků a chování při úderu) a byl nastaven typ trávnikových spritů na „urban_grass_overgrown“. Využitý typ (položka %detailtype) a textura (definována dříve ve vlastnostech mapy) byly již součástí hry.

Obrázek 87 – Výsledná přechodová textura pro terén parku (vlastní zpracování)

Obrázek 88 – VMT soubor definující přechodovou texturu pro terén parku (vlastní zpracování)

Obrázek 89 – Výsledek aplikované přechodové textury s trávnikovými sprity (vlastní zpracování)

Obrázek 90 – Vytvořené modely pro detailování lékárny (vlastní zpracování)

Obrázek 91 – Vytvořené modely pro konstrukci rozbitých stěn ve vnitrobloku (vlastní zpracování)

1.10.4 Detailování

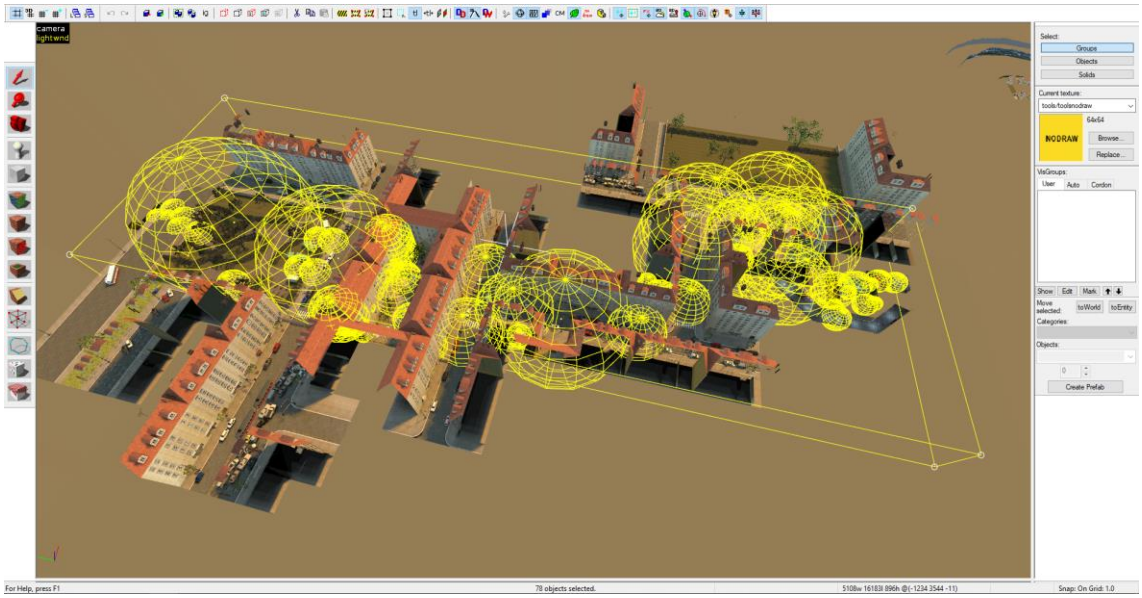
Výsledky detailování jednotlivých lokací viz příloha č. 1 sekce „Mapa 2“.

1.10.5 Osvětlení, mlha a odrazy

Osvětlení a mlha byly nastaveny na měkčí, růžovější barvu pro znázornění časového přechodu do vyrázně oranžové třetí mapy. Oproti předešlé mapě byl ovšem nastaven hraniční limit vykreslování (far-z clip) na nižší hodnotu, jelikož hráči v žádný moment nevidí tak daleko, jako je tomu v určitých lokacích první mapy.

1.10.6 Zvuková kulisa

Při definici zvukových kulis bylo převážně vycházeno z kulis předešlé mapy. Konkrétně byly definovány zvukové kulisy pro: startovní místnost, zaměstnanecké chodby, stanici metra, ulice, lékárnu, chodbu mezi obchody, večerku, bytový dům, vnitroblok, byt, děravou stěnu v bytě, koupelnu, ulici u parku, park, podchod pod silnicí a místnost změny mapy.



Obrázek 92 – Pokrytí entit env_soundscape – mapa 2 (vlastní zpracování)

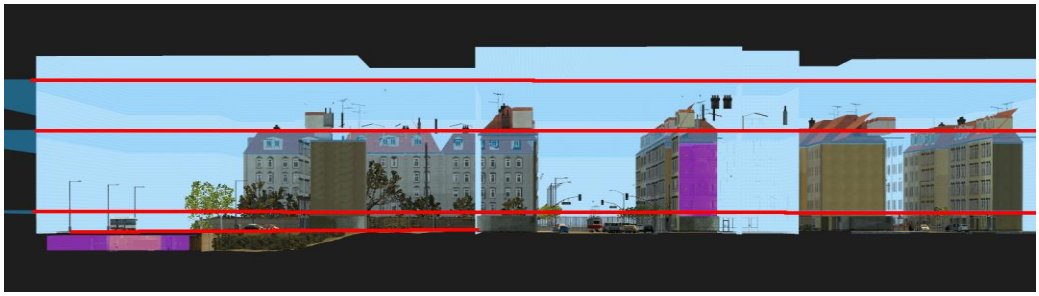
1.10.7 Herní logika

V rámci herní logiky byl nastaven alarm u dveří do lékárny, který se spouští rozbitím okenních nebo dveřních skel. Zároveň byl sestaven mechanismus, který zaručí, že se vždy buď v okolí parku nebo v podchodu pod silnicí objeví nepřítel, který vyžaduje mnoho zásahů, čímž budou hráči donuceni v parku strávit více času. Nakonec byla opět definována místnost změny mapy na toaletách v podchodu pod silnicí.

1.10.8 Optimalizace

Díky přenesení spojujících se ulic do 3D skyboxu byla tato mapa velmi dobře segmentována. Nebylo tedy nutno příliš sekat visleafs pohledem ze shora, stačilo jen opět celou mapu 3-4 krát rozdělit pohledem ze strany (rozdělení vysokých visleafs na menší části). Nakonec nebylo ani potřeba definovat žádné areaportály.

Obrázek 93 – Sekání visleafs pohledem ze shora (mapa 2) (vlastní zpracování)



Obrázek 94 – Sekání visleafs pohledem ze strany (mapa 2) (vlastní zpracování)

1.10.9 Navigace umělé inteligence

Při tvorbě navigační sítě bylo postupováno velmi podobně jako u první mapy. Rozdílem ovšem bylo, že vnitroblok a jeho střechy nejsou odděleny od ostatních částí mapy. Bylo tedy třeba definovat žebříky pro umělou inteligenci na okapech a vygenerovat navigační síť i na střechách. V první mapě měly specifické části střech také definovány malé navigační sítě a spojující žebříky, ovšem nikdy nebyly tyto sítě komplexně spojeny s ostatními lokacemi.

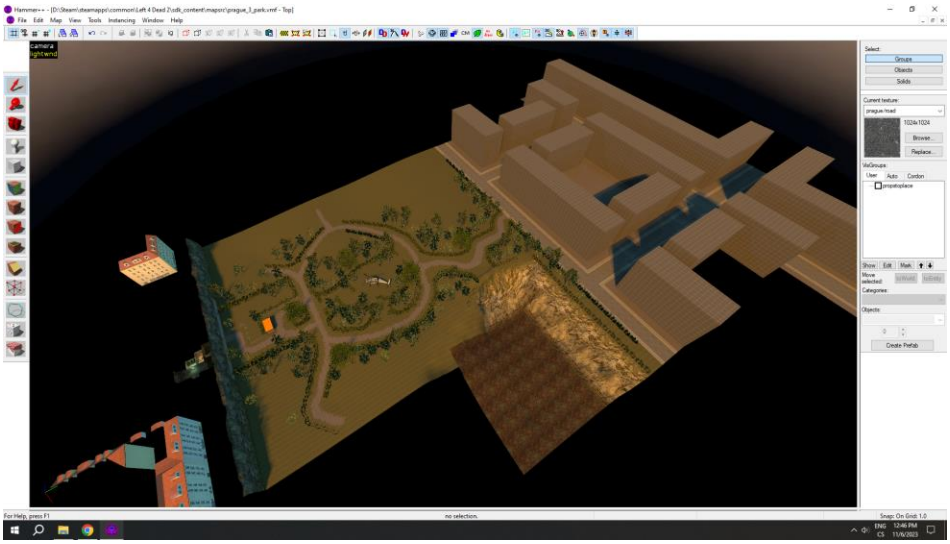
Obrázek 95 – Část navigace umělé inteligence – mapa 2 (vlastní zpracování)

1.11 Mapa 3 – přechod na sídliště

1.11.1 Hrubý obrys světa

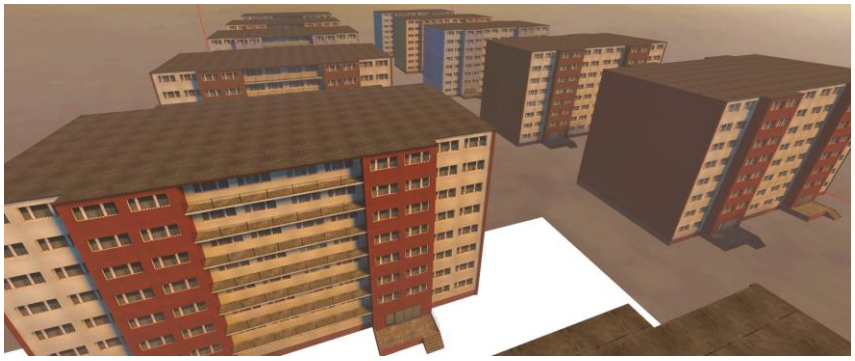
Tvorba této mapy se velmi lišila od předešlých dvou, bylo potřeba vytvořit park a postupný přechod na sídliště. Nejdříve byla zkopírována část podchodu pod silnicí, včetně místnosti změny mapy (nyní startovní místnosti) z mapy předešlé, a poté byl tvarován terén parku.

Terén byl natvarován tak, aby se postupně zvyšoval a hráči šli postupně směrem nahoru. Před výstupem z parku bylo vytvořeno prohloubení, aby zde byl prostor na pozdější implementaci panoramatického výhledu. Samotné sídliště bylo prozatím obsazeno kvádry, jelikož zatím nebyly vytvořeny vzory panelových domů.



Obrázek 96 – Výsledný hrubý obrys – mapa 3 (vlastní zpracování)

Následně bylo vytvořeno pět různobarevných kombinací panelových domů, pro každou kombinaci byla také vytvořena varianta s balkóny a bez balkónů. Dříve položené kvádry na místo panelových domů byly poté nahrazeny vytvořenými vzory.



Obrázek 97 – Vytvořené varianty dvou typů panelových domů (vlastní zpracování)

Obrázek 98 – Obsazení hratelného prostoru panelovými domy (vlastní zpracování)

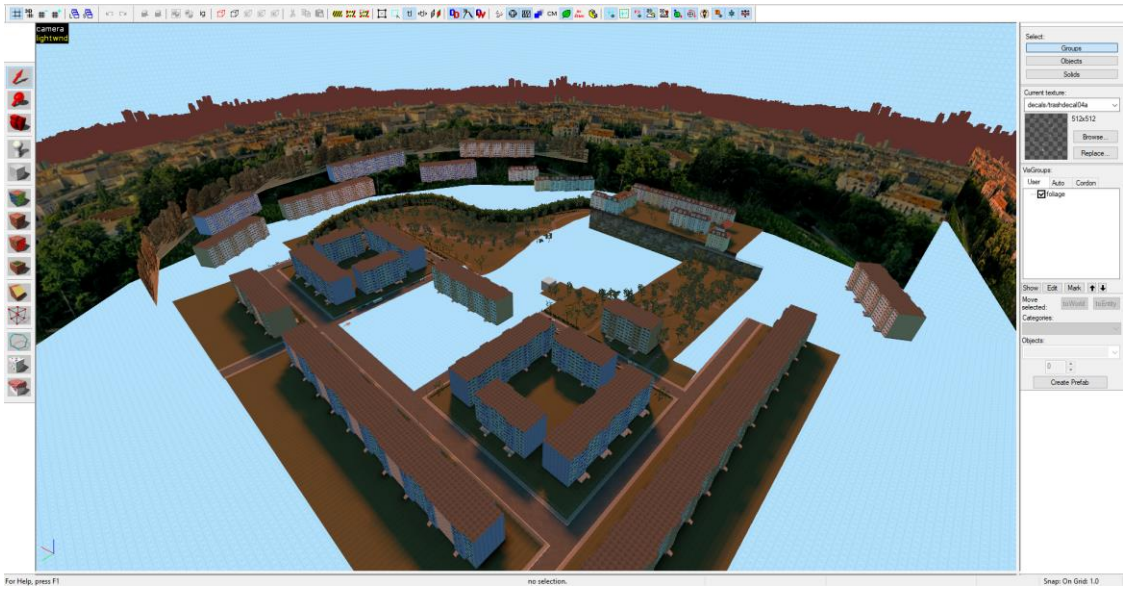
1.11.2 3D skybox

Pro sestavení 3D skyboxu bylo potřeba vytvořit příslušný model skupiny stromů – toho bylo docíleno změnou textury modelu již poskytnutého hrou a rekompilace tohoto upraveného modelu na nový (pomocí programu Crowbar).

Dále byly ze dříve vytvořených vzorů panelových domů zkonstruovány zjednodušené a zmenšené modely pro 3D skybox (pomocí kompilátoru Propper). Nakonec byla z panoramatické fotky pořízené z Google Street View vytvořena panoramatická transparentní textura (prostřednictvím programu Gimp a VTFedit).

Obrázek 99 – Vytvořené modely pro 3D skybox – mapa 3 (vlastní zpracování)

Obrázek 100 – Panoramatická textura města pro 3D skybox – mapa 3 (vlastní zpracování)



Obrázek 101 – Výsledný 3D skybox – mapa 3 (vlastní zpracování)

Obrázek 102 – Iluze výhledu na město projekcí 3D skyboxu – mapa 3 (vlastní zpracování)

1.11.3 Detailování

Výsledky detailování jednotlivých lokací viz příloha č. 1 sekce „Mapa 3“.

1.11.4 Osvětlení, mlha a odrazy

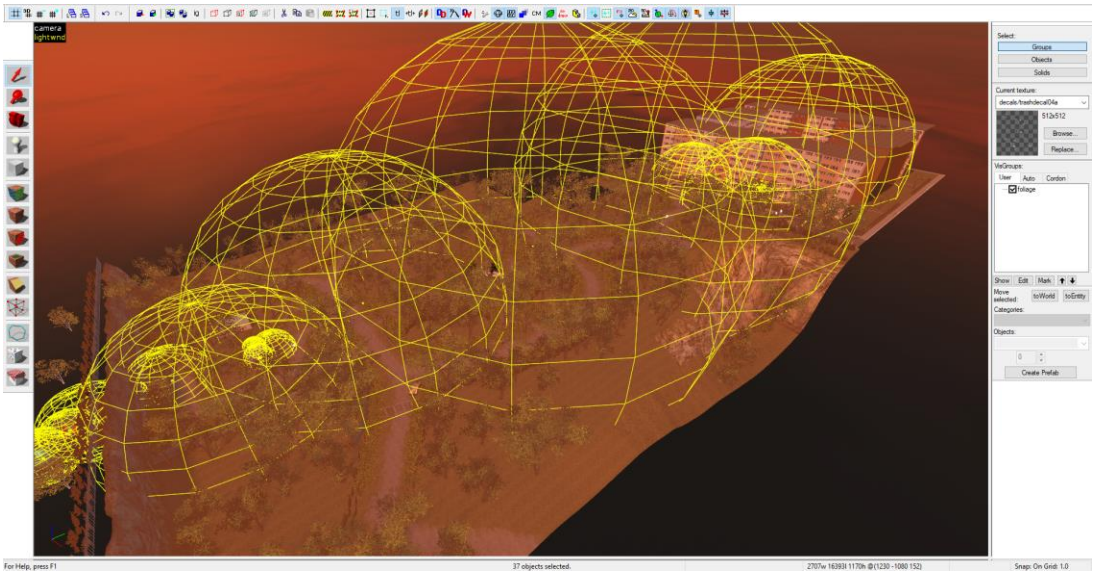
Osvětlení a mlha byly nastaveny na výrazně oranžovou barvu pro zvýraznění posunu času po absolvování předchozích dvou map. Při konfiguraci odrazů bylo potřeba entity env_cubemap relativně rovnoměrně rozmístit po parku a koncentrovat jejich rozmístění na panelové domy (aby odrazy každého okna lépe odpovídaly očekávaným odrazům podle výšek jejich poloh). Výsledné odrazy nebyly perfektně přesné, ovšem využít jednu entitu env_cubemap pro každé okno by exponenciálně zvýšilo velikost souboru této mapy se zanedbatelným zlepšením kvality odrazů.

Obrázek 103 – Výsledné odrazy po kompilaci cubemap (vlastní zpracování)

1.11.5 Zvuková kulisa

Opět bylo převážně vycházeno ze souboru kulis předchozí mapy, v obecné kulise chaosu byly ztišeny zvuky policejních sirén a byly zdůrazněny vzdálené zvuky střelby a explozí. Do kulisy parku bylo přidáno více lesních zvuků a všechny venkovní kulisy této mapy byly obohaceny o tiché vzdálené hřmění.

Konkrétně byly pro tuto mapu vytvořeny kulisy: startovní místnost, podchod pod silnicí, park, stánek s občerstevním, záchod stánku s občerstvením, ulice, panelový dům, byt 1, byt 2, koupelna, vnitroblok sídliště a prádelna panelového domu.



Obrázek 104 – Pokrytí entit env_cubemap – mapa 3 (vlastní zpracování)

1.11.6 Herní logika

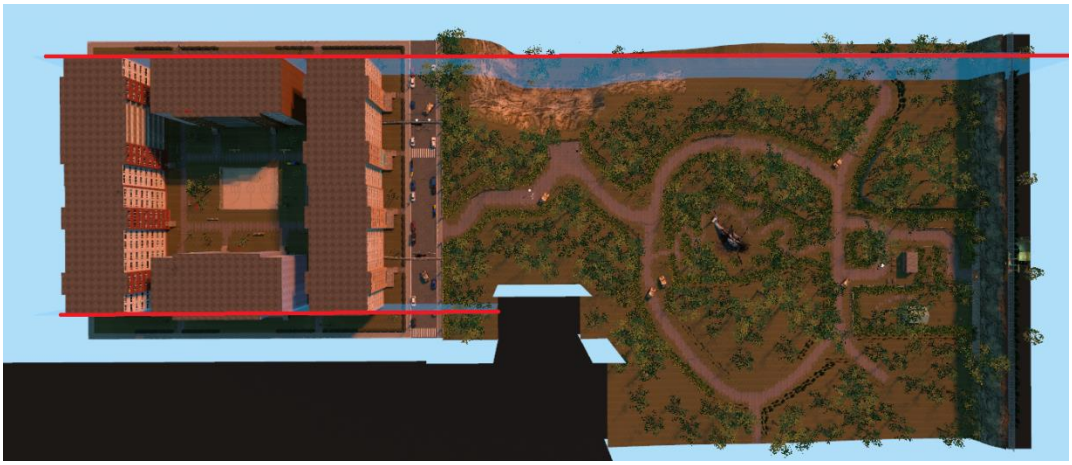
Pro tuto mapu bylo definováno rozbití plotové barikády explozí výbušných barelů v oblasti panoramatického výhledu, které způsobí, že hráči v této oblasti musí určitý čas přežít.

Dále byl definován nekonečný panický event při opuštění prvního panelového domu. Je nejdříve zjišťováno, zda jsou všichni hráči v panelovém domu a pokud ano, tak jsou odemčeny zadní dveře. Po otevření zadních dveří je spuštěn alarm a vlastně vytvořený .nut skript, který definuje, že na hráče mají ve velmi krátkých intervalech útočit nepřátelé do té doby, než se hráči dostanou do koncové místnosti a změní mapu.

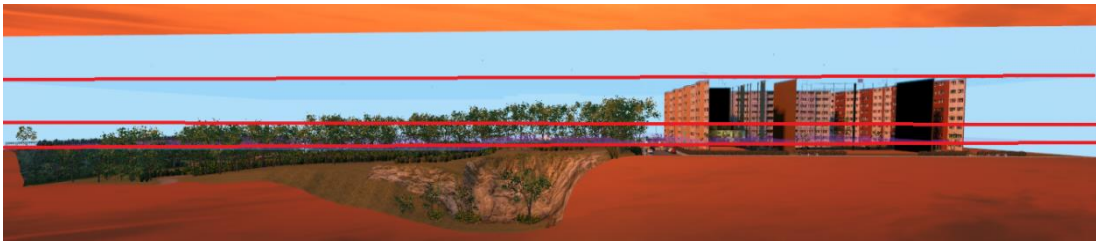
Pro koncovou místnost bylo opět definována a zprovozněna změna mapy pomocí trigger_changelevel a info_landmark.

1.11.7 Optimalizace

Jelikož byla tato mapa zkonstruována velmi otevřeným způsobem, jediné sekání ze shora bylo provedeno podél postranních stěn panelového bloku. Dále byly visleafs mapy opět rozděleny na tři části pohledem ze strany.



Obrázek 105 – Sekání visleafs pohledem ze shora – mapa 3 (vlastní zpracování)



Obrázek 106 – Sekání visleafs pohledem ze strany – mapa 3 (vlastní zpracování)

Byl zde ovšem problém s průchodným panelovým domem, jehož chodby a okna poskytnuly přímý pohled mezi parkem a vnitroblokem. Toto bylo vyřešeno aplikováním areaportálů na obě strany průchodného panelového domu. Při překročení určité vzdálenosti hráči je portál zavřen a nic za panelovým domem není vykreslováno.

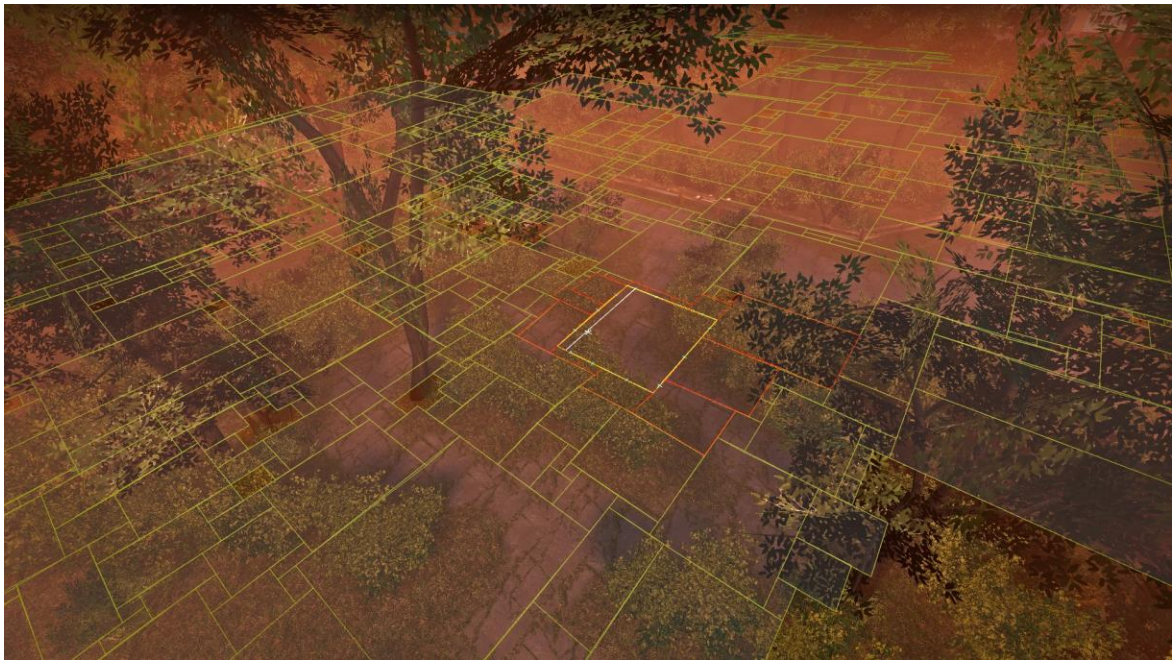
Celé mapě byla nastavena hranice vykreslování na 5000 bodů, jelikož hráči často nevidí příliš daleko kvůli rozsázené zeleni. Zároveň byla všem křovím nastavena vykreslovací vzdálenost tak, aby hráč nikdy přímo neviděl mizející keře.

Díky všem těmto optimalizačním technikám byla vykreslovací náročnost této mapy signifikantně snížena.

1.11.8 Navigace umělé inteligence

Tvorba navigační sítě umělé inteligence se alespoň v parku velmi lišila od předešlých map. Keře byly nastaveny jako průchodné pro umělou inteligence a neprůchodné pro hráče pomocí neviditelných stěn. Jelikož umělá inteligence ignorovala kolize keřů, bylo možné velmi rychle vygenerovat navigační síť parku.

Bylo ovšem potřeba důkladně celou síť projít a označit skryté plochy jako „obscured“ (nepřátelé se zde mohou objevit, i když je hráč blízko) a „empty“ (aby se nepřátelé objevovali především na viditelných plochách).

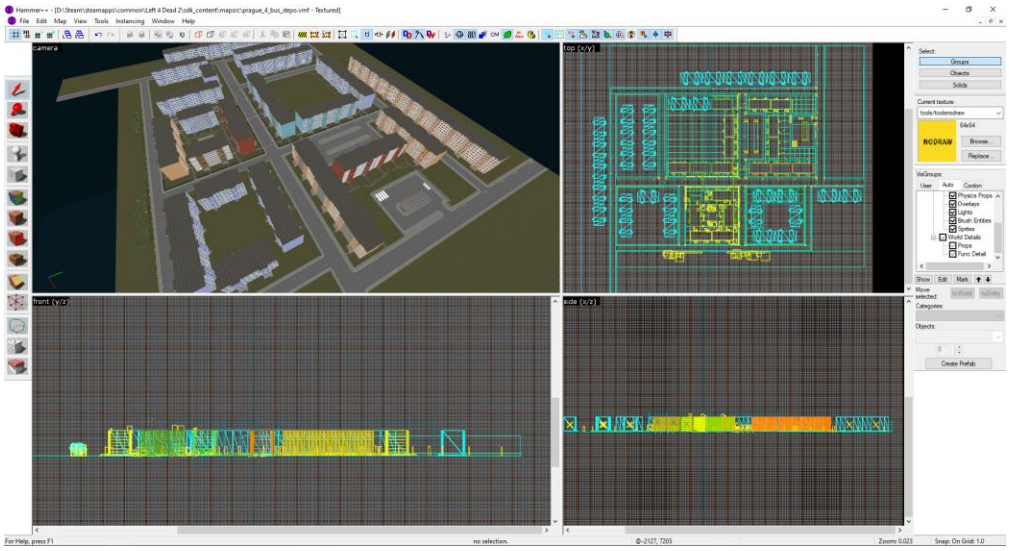


Obrázek 107 – Navigace umělé inteligence v horní části parku (vlastní zpracování)

1.12 Mapa 4 – autobusové depo

1.12.1 Hrubý obrys světa

Tvorba hrubého obrysu poslední mapy byla oproti ostatním relativně jednoduchá. V minulé mapě byly již vytvořeny vzory panelových domů a jejich varianty, jen bylo třeba upravit vzory na noční variantu. U vzorů panelových domů používaných při stavbě hrubého obrysu stačilo jen textury oken nahrazadit vytvořenými variantami oken se zapnutými světly a relativně rychle bylo možné poskládat ulice sídliště.

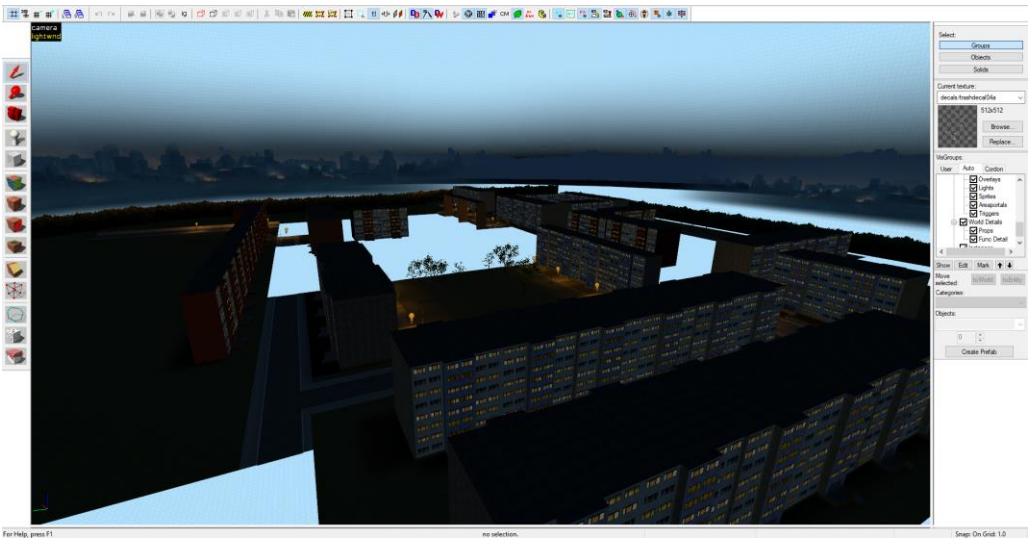


Obrázek 108 – Hrubý obrys světa – mapa 4 (vlastní zpracování)

1.12.2 3D skybox

Objevil se ovšem problém při převodu 3D skybox modelů na noční varianty. Jelikož se stěny modelů skládají pouze z jedné textury (stěny jsou tvořen z fotek celých stěn vzorů panelových domů), bylo potřeba manuálně ztmavit části textury kolem oken a textuře nastavit samo-iluminaci pomocí paramatrů $selfillum a $selfillumtint ve .vmt souboru.

Výsledné textury stěn s okny modelů panelových domů jsou enginem zesvětleny samo-iluminací na potřebnou míru a části textury reprezentující okna (které jsou již dostatečně světlé – nebyly manuálně ztmaveny) jsou viditelně světlejší, než okolní stěna.



Obrázek 109 – 3D skybox – mapa 4 (vlastní zpracování)

Obrázek 110 – Iluze pokračující ulice díky 3D skyboxu – mapa 4 (vlastní zpracování)

1.12.3 Tvorba potřebných assetů

Jediným potřebným assetem pro tuto mapu byl model autobusové zastávky. Tento model byl opět vytvořen nejdříve extrakcí textur z Google Street View pomocí programů Shoebox, Gimp a VTFedit a dále byl pomocí kompilátoru Propper zkompilován model využívající tyto textury. Nakonec byly vytvořeny různé varianty pro čísla autobusů vypsaných na horní straně cedule.

Obrázek 111 – Varianty modelu autobusové zastávky (vlastní zpracování)

1.12.4 Detailování

Výsledky detailování jednotlivých lokací viz příloha č. 1 sekce „Mapa 4“.

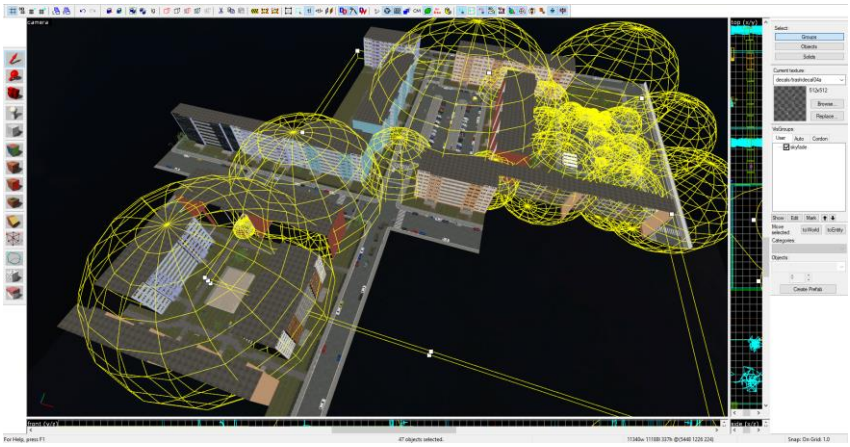
1.12.5 Osvětlení, mlha a odrazy

Pro tuto mapu bylo zvoleno tmavě modré osvětlení s velmi nízkým jasem, dále byla po několika pokusech nalezena taková barva mlhy, která dobře splývala s noční oblohou. Při rozmisťování entit env_cubemap bylo postupováno podobně jako v předešlé mapě – většina entit byla soustředěna na okna panelových domů.

1.12.6 Zvuková kulisa

Při tvorbě této mapy byly definovány konkrétně tyto zvukové kulisy: startovní místnost, byt, koupelna, ulice, parkoviště, park u autobusového depa, autobusové depo, stan autobusového depa, interiér autobusového depa a toalety autobusového depa.

Byly kompletně odstraněny venkovní zvuky sirén a přelety helikoptér, místo nich byly do kulisy převážně zdůrazněny zvuky větru, vzdálených explozí a blížející se bouřky.

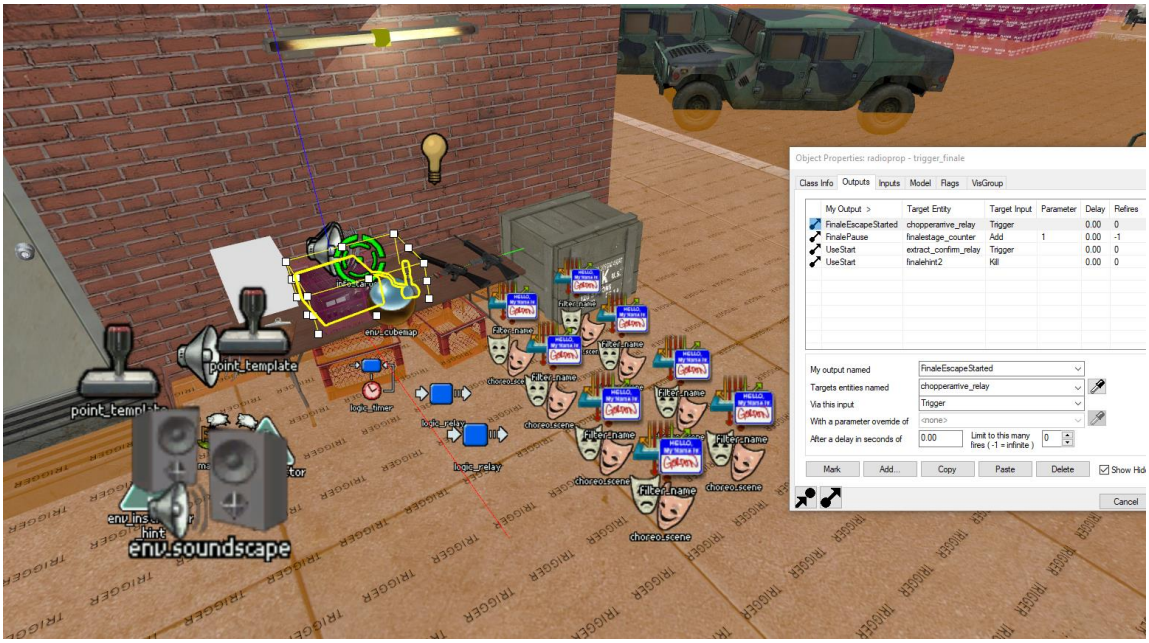


Obrázek 112 – Pokrytí entit env_soundscape – mapa 4 (vlastní zpracování)

1.12.7 Herní logika

Hlavní náplní práce při nastavování herní logiky bylo zprovoznění takzvaného „finále“. Hráči obecně interagují s rádiem a po krátkém rozhovoru je finále spuštěno. Hráči pak musí přežít určité množství vln nepřátel, než mohou oblast opustit a tím dokončit kampaň.

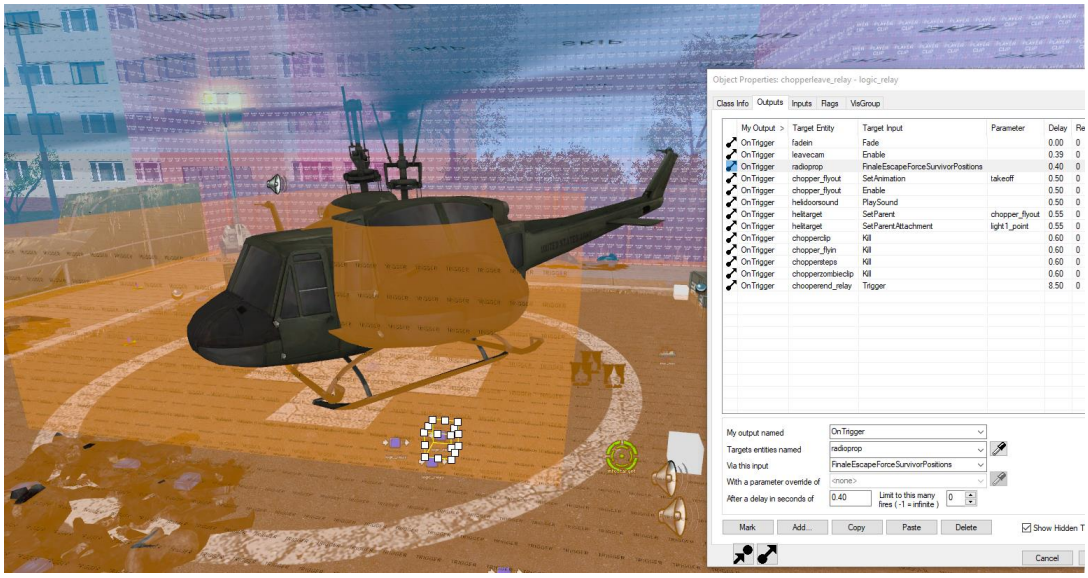
Tato herní sekvence je sestavena takovým způsobem, že je nejdříve zjišťováno, zda jsou všichni hráči v autobusovém depu. Po potvrzení, že jsou všichni hráči přítomni, začnou z rádia vycházet různé hlasy, což hráče především upozorní na lokaci rádia a jeho funkčnost. Při interakci s rádiem proběhne příslušná konverzace (je kontrolováno, jaká z postav má mluvit) a po další interakci je spuštěno finále.



Obrázek 113 – Část herní logiky zprovozňující finále (vlastní zpracování)

Tento konkrétní typ finále byl přesněji definován v externím souboru prague_4_bus_depo_finale.nut, ve kterém bylo nastaveno kratší čekání mezi jednotlivými vlnami nepřátel a také byla nastavena větší intenzita útoků. Zároveň byla definována takzvaná skriptovaná vlna, při které jsou vypnuty téměř všechna okolní světla a hráči mají za úkol je znovu zapnout. Pro tuto vlnu byl definován samostatný .nut skript, který komunikuje s entitami ve světě a spustí relé zprostředkující tuto posloupnost událostí.

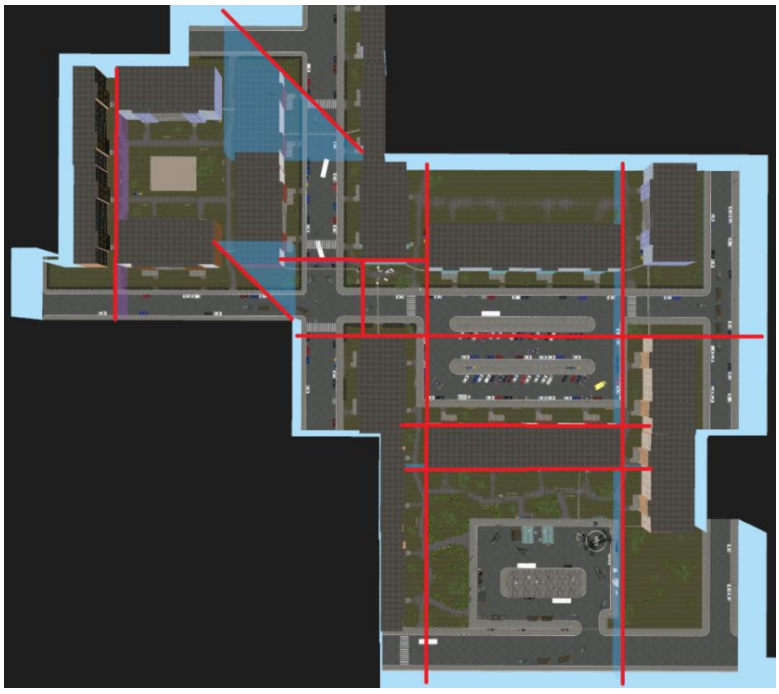
Pro sekvenci únikového dopravního prostředku byl využit model helikoptéry již poskytovaný hrou, jehož animace přistání byla pozicována na improvizovaný helipad v rohu autobusového depa. Po aktivaci triggeru, který zjišťuje, zda jsou v helikoptéře všichni aktuálně živí hráči, je helikoptéra odstraněna a je zapnuta další helikoptéra se stejným modelem. Tato druhá helikoptéra byla napozicována a nasměrována tak, aby při spuštění koncové ukázky prolétla před kamerou.



Obrázek 114 – Část herní logiky zprovozňující přílet a odlet helikoptéry (vlastní zpracování)

1.12.8 Optimalizace

Panelové domy této mapy dostatečně oddělily jednotlivé sekce mapy, bylo pouze potřeba provést několik již rutinních řezů, a to jak ze shora, tak ze strany. Zároveň byl přidán areaportál na startovní panelový dům. Hlavní optimalizační technikou této mapy byla opět kratší vykreslovací vzdálenost a nastavení vykreslovací vzdálenosti malých modelů.



Obrázek 115 – Sekání visleafs pohledem ze shora – mapa 4 (vlastní zpracování)

Obrázek 116 – Sekání visleafs pohledem ze strany – mapa 4 (vlastní zpracování)

1.12.9 Navigace umělé inteligence

Při tvorbě navigační sítě byl brán zřetel na lokaci autobusového depa, jelikož při nedostatečně důkladné tvorbě navigačních sítí mohou být určité plochy nedosažitelné pro umělou inteligenci. Zároveň bylo definováno několik částí okolního plotu, které mohou nepřátelé přelézt, aby bylo pro umělou inteligenci vždy možné se dostat do finální lokace a útočit na hráče.

Obrázek 117 – Část navigační sítě autobusového depa (mapa 4, vlastní zpracování)

Výsledky a diskuse

1.13 Zveřejnění

Kampaň byla zveřejěna na oficiálním workshopu poskytovaným hrou a na externím webovém portálu GameMaps.com. Officiální workshop mohou hráči navštívit ze stejného místa, ze kterého spouští hru, a pro instalování komunitních modifikací stačí stisknout jen jedno tlačítko.

Nevýhodou workshopu ovšem je, že ve většině případů nepovoluje nahrání modifikace větší než přibližně 225 MB. Celá kampaň tedy musela být rozdělena do 5 .vpk adresářů o průměrné velikosti přibližně 150 MB, které byly nahrávány jednotlivě, jelikož celková velikost kampaně činila necelých 900 MB. Z pohledu hráčů stačí pouze stisknout tlačítko „odebírat“ na jedné z publikovaných částí a budou vybídnuti k odběru všech ostatních částí.

Jak bylo ovšem již zmíněno, kampaň byla také zveřejněna na externím webovém portálu GameMaps.com, protože hostitelé serverů vyžadují jeden ucelený .vpk soubor pro využití na dedikovaných serverech a webový portál GameMaps.com umožňuje nahrání adresářů o velikosti až 2 GB.

Dne 2. 1. 2024 byla v 18:13 kampaň zveřejněna jak na workshopu, tak na GameMaps.com. Ve stejném měsíci bylo na základě zpětné vazby herní komunity a testerů kompetetivních serverů „ERROR“ vydáno 9 aktualizací. Obsahem těchto aktualizací bylo převážně vyvážení herní obtížnosti, detailnější dekorace vizuálně slabších lokací a řešení malých problémů.

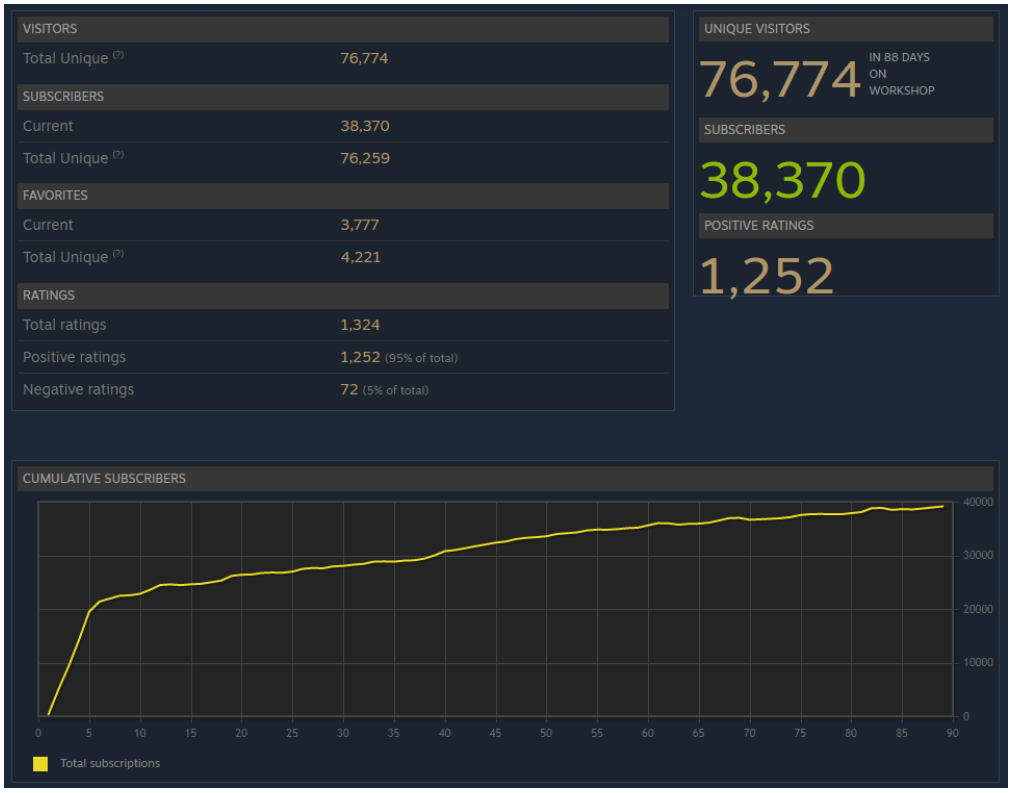
Obrázek 118 – Stránka steam workshopu zveřejněné kampaně ke dni 30. 3. 2024 (vlastní zpracování)

1.14 Statistiky a hodnocení (Steam Workshop)

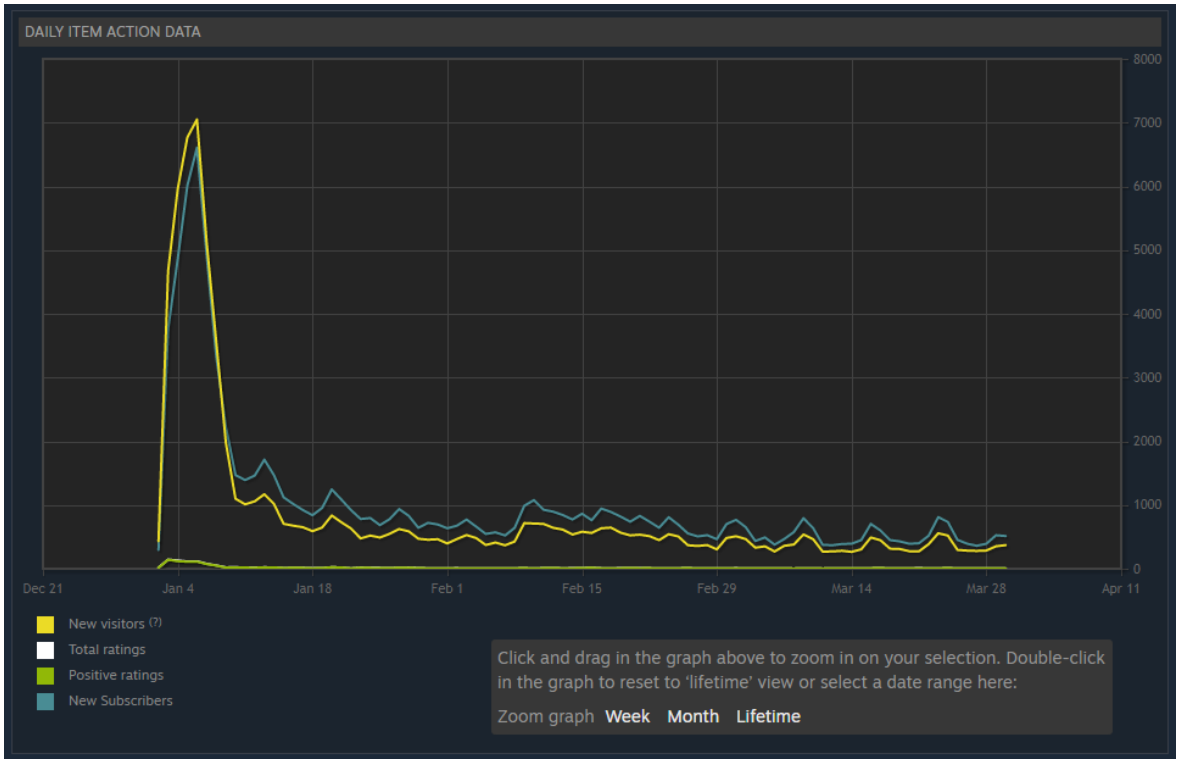
V průběhu prvního týdne po zveřejnění kampaně velmi rapidně vzrostl počet denních odběratelů, a to až na více než 6 500 unikátních odběratelů denně. Steam Workshop algoritmus pravděpodobně prioritizuje nabízení nově vydaných modifikací a tento efekt tedy bylo možné očekávat. V následujících dnech tato denní aktivita klesla na přibližně 1 000 nových odběratelů denně a ke dni 30. 3. 2024 stále kolísala v rozsahu od 500 do 1 000 odběratelů denně.

Celkově (ke dni 30. 3. 2024) kampaň nainstalovalo 76 259 hráčů, ovšem aktuálně ji má nainstalováno pouze 38 370, jelikož hráči často kampaně po dohrání odinstalují. Po celý životní cyklus publikované kampaně se poměr hodnocení držel na 95 % pozitivních a 5 % negativních (1324 hodnocení celkem). Kampaň tedy velmi rychle obdržela 5/5 hvězd, díky čemuž jsou hráči více motivováni tuto kampaň vyzkoušet.

Jak bylo možné očekávat, většina hráčů ocenila sluneční atmosféru a relativně čisté, detailované prostředí, jelikož se velmi mnoho komunitních kampaní odehrává převážně ve tmavých, často uzavřených a špinavých prostorech.



Graf 1 – Kumulativní počet odběratelů kampaně ke dni 30. 3. 2024 (vlastní zpracování)



Graf 2 – Počet návštěvníků, odběratelů a pozitivních hodnocení každý den do dne 30. 3. 2024 (vlastní zpracování)

1.15 Ekonomický odhad

Nelze naprosto přesně určit, jak nákladné by bylo tuto kampaň vytvořit v komerčním prostředí. Vývoj si vyžádal přibližně devět měsíců práce s tím, že časová intenzita práce tvorby byla velmi variabilní. Převážně z počátku vývoje bylo na kampani pracováno velmi intenzivně a jiné dny naopak nebylo na kampani pracováno vůbec.

Vyhledáním rozmezí platů pracovní pozice „designer her“ bylo zjištěno, že se hrubý měsíční plat může pohybovat v intervalu od 33 942 Kč až do 94 701 Kč. Zároveň byla zvážena více než desetiletá zkušenost autora a bylo rozhodnuto použít průměr těchto dvou hodnot, tedy konkrétně 64 321 Kč měsíčně. (Alma Career Czechia, s.r.o.)

Při výpočtu byl jeden pracovní měsíc zjednodušen na přibližně 20 pracovních dnů, přičemž jeden pracovní den se skládá z 8 hodin práce. Touto formou výpočtu vyšlo peněžní ohodnocení jedné hodiny práce na přibližných 400 Kč.

Doba trvání vývoje nebyla optimálně zaznamenávána, nástroj na tvorbu prostředí sice drží informaci o celkové délce používání tohoto programu, ovšem velká část práce se odehrávala mimo tento program a zároveň byl někdy program využíván bez přístupu k internetu, kvůli čemuž nebyla velká část vývoje časově ohodnocena. Samotný nástroj zobrazuje, že byl za přístupu k internetu využíván program po dobu přibližně 300 hodin. Reálný čas strávený na vývoji je ale blíže 800-1000 hodin práce.

Při konzervativním odhadu 800 hodin by vývoj stál zhruba 320 000 Kč a při odhadu 1000 hodin by cena vývoje činila 400 000 Kč. Je nutné podotkout, že v komerčním prostředí by na projektu podobného rozsahu pracoval tým zkušených vývojářů s různými oblastmi specializace a například tvorba vlastních assetů by zabrala daleko méně času.

Komerční tvorba této kampaně by nebyla ekonomicky viabilní, protože komunitní kampaně jsou dostupné zdarma a negenerují žádný peněžní příjem. V praxi by bylo postupováno profesionálním způsobem převážně jen při tvorbě nové hry, za jejíž prodej by mohlo herní studio generovat peněžní příjem.

1.16 Diskuze

Tvorba kampaně vyžadovala mimo samotný architekturální design také tvorbu vlastních assetů, práci se zvukovým systémem, optimalizaci vykreslování, vysokoúrovňové skriptování, práci s herní logikou a mnoho dalších. Práce v těchto oblastech nebyla v praktické části popisována příliš do hloubky, jelikož by přesáhla požadovaný rozsah.

Zároveň je nutné podotknout, že obvykle bývají kampaně podobného rozsahu tvořeny skupinami vývojářů, v ideálním případě by tato vývojářská skupina obsahovala:

- Tvůrce assetů (textury, modely, zvuky a zvukové kulisy)
- Levelového designera zaměřeného na hratelnost (tvroba hrubých obrysů jednotlivých map, práce s herní logikou, tvorba navigační sítě a vysokoúrovňové skriptování)
- Levelového designera zaměřeného na detailování (detailování, definice osvětlení, odrazů a následná optimalizace vykreslování)
- Testery (odhalování grafických a herních problémů)

V rámci této práce bylo ovšem nutno vývoj provést individuálně. Po zveřejnění kampaně a následném navázání spolupráce s hostitelem serverů bylo možné velmi rychle identifikovat problémy a odhalit potenciální možnosti vylepšení kampaně díky externí pomoci testerů těchto serverů. Tímto bylo velmi jasně ukázáno, že může být práce v oblasti tvorby herních světů mnohem efektivnější a méně časově náročná při zapojení více vývojářů.

Závěr

Na základě řešerše informačních zdrojů byla v teoretické části zpracována plánovaná charakteristika schopností a limitací Source Engine včetně rešerše způsobů generování odrazů a osvětlení tohoto enginu. Také byly do této části zpracovány vizuální příklady, a to převážně formou vlastního zpracování díky silnému tématickému propojení těchto dílčích cílů s hlavním praktickým cílem tvorby prostředí v Source Engine.

V rámci vlastní práce bylo nejdříve provedeno porovnání funkčností Source Engine s novejšími hernímy enginy Unreal Engine 5, Unity, Source 2 a s jeho předchůdcem GoldSrc. Výsledky těchto porovnání byly sumarizovány prostřednictvím souhrnné tabulky na konci této části. Dále byl v rámci vlastní práce zpracován hlavní cíl tvorby herní kampaně pro hru Left 4 Dead 2 v Source Engine skládající se ze 4 navazujících map odehrávajících se ve specifických částech Prahy. Obsah této části byl především zaměřen na architekturální tvorbu vykreslovaného světa, ovšem ve skutečnosti bylo pracováno s mnoha komponenty Source Enginu.

Výsledný produkt obdržel velmi pozitivní hodnocení komunity a byla velmi rychle navázána neformální spolupráce s hostitelem kompetetivních serverů, díky jejichž testerům bylo možno nalézt problémové oblasti a postupnými aktualizacemi kampaň učinit více kompetitivně viabilní. Plánem do budoucna je kampaň dále aktualizovat a zvýšit tak šanci jejího využití na kompetetivních turnamentech v této hře.

Vývoj v relativně starém Source Enginu obsahuje mnoho nevýhod. Modernější herní enginy podporují rozsáhlejší herní světy, kvalitnější odrazy a osvětlení, 64bitové architektury (díky kterým mají tyto enginy méně technických limitací) a mnoho dalších funkcionalit velmi užitečných pro herní vývojáře. Jednou siginifikantní výhodou Source Engine je ovšem jeho otevřenost a rozsah možností modifikace her využívajících tento engine. Nezávislí vývojáři mají tedy možnost vytvářet herní obsah bez nutnosti tvorby nové hry. Zároveň mohou být tyto výtvory využity velkým množstvím hráčů, jelikož obecně jsou hry využívající tento engine stále relativně populární.

Tento způsob herního vývoje je velmi vhodný pro nezávislé, nekomerční a mnohdy začínající vývojáře, jelikož si mohou v tomto prostoru osvojit proces vývoje a případně navázat kontakty pro budoucí komerční spolupráci.

Bibliografie

Alma Career Czechia, s.r.o. Designer Her. *Platy.cz*. [Online] [Citace: 3. únor 2024.] https://www.platy.cz/platy/informacni-technologie/designer-her.

Bernier, B. 2014. *Source SDK Game Development Essentials*. Birmingham : Packt Publishing Ltd., 2014. str. 265. ISBN 978-1-84969-592-3.

Epic Games, Inc. 2022. Lumen Global Illumination and Reflections. *Unreal Engine 5.0 Documentation*. [Online] 2022. [Citace: 16. říjen 2023.] https://docs.unrealengine.com/5.0/en-US/lumen-global-illumination-and-reflections-in-unreal-engine/.

Epic Games, Inc. 2023. Nanite Virtualized Geometry. *Unreal Engine 5.2 Documentation*. [Online] 2023. [Citace: 16. říjen 2023.] https://docs.unrealengine.com/5.2/en-US/nanite-virtualized-geometry-in-unreal-engine/.

Epic Games, Inc. 2022. One File Per Actor. *Unreal Engine 5.0 Documentation*. [Online] 2022. [Citace: 16. říjen 2023.] https://docs.unrealengine.com/5.0/en-US/one-file-per-actor-in-unreal-engine/.

Epic Games, Inc. 2023. World Partition. *Unreal Engine 5.3 Documentation*. [Online] 2023. [Citace: 16. říjen 2023.] https://docs.unrealengine.com/5.3/en-US/world-partition-in-unreal-engine/.

Gregory, J. 2018. *Game engine architecture*. Boca Raton : Taylor & Francis Group, 2018. str. 1165. Sv. 3. ISBN 978-1-1380-3545-4.

Hilton, Chris. 2021. Realtime, Mixed and Baked Light Modes — For Static & Dynamic Game Objects in Unity. *Medium*. [Online] 27. Listopad 2021. https://christopherhilton88.medium.com/realtime-mixed-and-baked-light-modes-for-static-dynamic-game-objects-in-unity-896e2721a839.

Lang, Ben. 2020. 'Half-Life: Alyx' Update Adds Impressive Liquid Simulation. *Road to VR*. [Online] 29. Květen 2020. [Citace: 29. říjen 2023.] https://www.roadtovr.com/half-life-alyx-update-1-4-liquid-simulation-subtitles-mod-scripting-lua/.

MangyCarface. 2009. Optimization in Source: A Practical Demonstration. *NODRAW*. [Online] 13. prosinec 2009. [Citace: 25. červen 2023.] https://nodraw.net/2009/12/optimization-in-source-a-practical-demonstration/.

Mariteaux. 2017. Making Use of Source's Water Materials. *Valve Developer Union*. [Online] 3. Prosinec 2017. [Citace: 3. červenec 2023.] https://valvedev.info/guides/making-use-of-sources-water-materials/.

Menard, M. a Wagstaff, B. 2015. *Game Development with Unity*. Boston : Cengage Learning PTR, 2015. str. 432. Sv. 2. ISBN 978-1-305-11054-0.

Satheesh, P. 2016. *Unreal Engine 4 Game Development Essentials*. Birmingham : Packt Publishing Ltd., 2016. ISBN 978-1-78439-196-6.

the_best_flash. 2022. Adding effects to Source textures. *The Whole Half-Life*. [Online] 10. říjen 2022. [Citace: 1. červenec 2023.] https://twhl.info/wiki/page/Tutorial%3A_Adding_effects_to_Source_textures.

Unity Technologies. 2023. Graphics API support. *Unity Documentation*. [Online] 2023. [Citace: 22. říjen 2023.] https://docs.unity3d.com/2020.3/Documentation/Manual/GraphicsAPIs.html.

Unity Technologies. 2023. Parallax Diffuse. *Unity Documentation*. [Online] 2023. [Citace: 22. říjen 2023.] https://docs.unity3d.com/Manual/shader-NormalParallaxDiffuse.html.

Unity Technologies. 2023. Physics. *Unity Documentation*. [Online] 2023. [Citace: 22. říjen 2023.] https://docs.unity3d.com/Manual/PhysicsSection.html.

Unity Technologies. 2023. Reflection Probe. *Unity Documentation*. [Online] 2023. [Citace: 22. říjen 2023.] https://docs.unity3d.com/Manual/class-ReflectionProbe.html.

Valve. 2012. Advanced Lighting. *Valve Developer Community*. [Online] 2. březen 2012. [Citace: 1. červenec 2023.] aktualizováno 2022. https://developer.valvesoftware.com/wiki/Advanced_Lighting.

Valve. 2023. Cubemaps. *Valve Developer Community*. [Online] 19. květen 2023. [Citace: 3. červenec 2023.] https://developer.valvesoftware.com/wiki/Cubemaps.

Valve. 2009. Entity Limit. *Valve Developer Community*. [Online] 15. září 2009. [Citace: 4. červenec 2023.] aktualizováno 2023. https://developer.valvesoftware.com/wiki/Entity_limit.

Valve. 2005. GoldSrc. *Valve Developer Community*. [Online] 25. srpen 2005. [Citace: 24. říjen 2023.] aktualizováno 2023. https://developer.valvesoftware.com/wiki/GoldSrc.

Valve. 2005. Grouping and VisGrouping. *Valve Developer Community*. [Online] 11. duben 2005. [Citace: 25. červen 2023.] aktualizováno 2023. https://developer.valvesoftware.com/wiki/Grouping_and_VisGrouping.

Valve. 2005. Source. *Valve Developer Community*. [Online] 28. červen 2005. [Citace: 3. červenec 2023.] aktualizováno 2023. https://developer.valvesoftware.com/wiki/Source.

Valve. 2017. Source 2. *Valve Developer Community*. [Online] 3. září 2017. [Citace: 29. říjen 2023.] aktualizováno 2023. https://developer.valvesoftware.com/wiki/Source_2.

Zorn, Denis. 2019. Modeling and Animating for Half-Life. *NYU Media Research Lab*. [Online] 2019. [Citace: 24. říjen 2023.] https://mrl.cs.nyu.edu/~dzorin/ig04/lecture22/modeling.pdf.

Seznam obrázků, tabulek, grafů a zkratek

1.17 Seznam obrázků

1.18 Seznam tabulek

1.19 Seznam grafů

1.20 Seznam použitých zkratek

Hra – videohra, interaktivní elektronický software

Editor – mapový editor, software na tvorbu herních světů

LOD – level of detail, variabilní komplexita modelu podle vzdálenosti hráče

Mapa – herní svět

RGB – trojice barev červené, zelené a modré

Brush – geometrický tvar či plocha

Far-z clip – maximální vykreslovací vzdálenost

2D – dvojrozměrný

3D – trojrozměrný

API – aplikační programovací rozhraní

VPK – Valve Pak, adresáře pro herní obsah

VTF – Valve Texture Format

VMT – Valve Material

Skin – variabilní textura modelu

Přílohy

Příloha č. 1 – „vysledky_detailovani.pdf“