Gymnázium, Praha 6, Arabská 14

Hra s prvky neeuklidovské geometrie
vedoucí práce Mgr. Jan Lána

Hra s prvky neeuklidovské geometrie

Maturitní práce

Denis Mašek 4.E

Duben 2024

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

_____

V Praze dne 2. dubna 2024

Anotace

Mým úkolem bylo vytvořit příběhovou hru využívající prvky neeuklidovské geometrie. Má hra je spojením herních žánrů puzzle a horror, tudíž vyzkouší rychlé přemýšlení hráčů pod tlakem. Odehrává se v omezené modelované části prostor Gymnázia Arabská, což ještě pro lidi, kteří toto místo znají, přidává k horrorovému aspektu hry.

Zadání

Zadáním mého maturitního projektu je vytvořit puzzle hru využívající prvky neeuklidovské geometrie. Tato hra by se také dala nazvat „4D hrou“, ale není to úplně přesné, protože nepřidává čtvrtou dimenzi, ale pouze si chytře hraje s třídimenzionálním prostorem. Neeuklidovská geometrie znamená, že hra bude porušovat pátý Euklidův postulát, říkající, že pokud je úsečka protnuta dvěma přímkami, které s ní svírají vnitřní úhly menší než 180°, tak se tyto přímky nakonec protnou. V neeuklidovské geometrii také nemusí mít rovnostranný trojúhelník vnitřní úhly 60° a nejkratší cesta mezi dvěma body nemusí být úsečka, což umožňuje vytvářet věci jako čtvercové pokoje, které mají pouze tři rohy, různé portály a techniky s perspektivou. Příkladem hry s neeuklidovskou geometrií je například Superliminal nebo Portal, moje hra má být v podstatě spojením mechanik těchto dvou her do jedné. Má hra je příběhová puzzle hra, která se odehrává v ohraničených prostorech naší školy a vyzkouší logické přemýšlení jejích hráčů. Bude se jednat o několika minutové demo spíše na předvedení jednotlivých technik, protože příběhové hry jsou velmi časově náročné a skoro zásadně nevznikají jako projekt jednoho člověka, ale jako práce celých velkých týmů.

Obsah

1. Úvod

Tato práce se zabývá nejen samotnou tvorbou mé hry, ale také historií a problematikou neeuklidovské geometrie jako takové. Jedná se o velmi zajímavou oblast matematiky, která není pro její složitost na středních školách probírána, přesto má ale široké využití na poli tvorby simulací a her.

1.1. Cíl práce

Cílem této práce bylo naprogramovat hru, která donutí lidi přemýšlet rychle a pod tlakem, a to v prostředí, které je jim známé a neznámé zároveň. Hra se odehrává v prvním patře krátké chodby Gymnázia Arabská, která se ale postupem času mění k nepoznání právě díky prvkům neeuklidovské geometrie.

Prvky neeuklidovské geometrie jsou zde reprezentovány dvěma způsoby. Prvním z nich je mnoho portálů, jejichž účelem je dezorientovat hráče a pokusit se zmást jeho prostorovou orientaci, což ještě přidává k horrorovému aspektu hry.

Druhým prvkem neeuklidovské geometrie je předmět, který jsem nazval „perspektivní manipulátor“. Pomocí tohoto nástroje můžete zvedat a měnit velikost předmětů tak, že se podíváte na jinak vzdálené pozadí a předmět se perspektivně naškáluje tak, aby byl zdánlivě stále stejně velký.

Hra má také velmi důležitou příběhovou část, která je vyprávěna pouze velmi podvědomě a je částečně vysvětlena až poslední scénou hry. Hra záměrně není vytvořena stylem, že by hráče vedla za ruku, ale počítá se s tím, že se mu podaří danou část projít třeba až na několikátý pokus.

2. Historie neeuklidovské geometrie

Na rozdíl od euklidovské geometrie, kterou popsal řecký matematik a geometr Euklides už kolem roku 300 před naším letopočtem, je neeuklidovská geometrie mnohem mladší. Hlavním důvodem tohoto rozdílu v jejich stáří je fakt, že neeuklidovská geometrie je založena čistě na matematických výpočtech a axiomech, zatímco euklidovskou geometrií se řídí reálný svět, ve kterém žijeme a který můžeme pozorovat.

2.1. Počátky

První teorie o existenci neeuklidovské geometrie vznikají v jedenáctém, dvanáctém a třináctém století našeho letopočtu. Jejich autory jsou významní matematici své doby Ibn al-Haytham, Omar Khayyam a Nasir al-Din al-Tusi, ti však dosáhli pouze velmi nekompletních popsání neeuklidovské geometrie, která obsahují chybné důkazy paralelního postulátu.

2.2. Evropský pokrok

Tyto částečně chybné práce však hrály velmi důležitou roli jako inspirace pro evropské matematiky o mnoho let později. Jedním z těchto důležitých evropských matematiků je Giovanni Girolamo Saccheri, který je považován za otce neeuklidovské geometrie a který vydal v roce 1733 knihu s názvem Euclides ab omni naevo vindicatus neboli Euklides osvobozen od všech chyb.

Hlavní podstata Saccheriho práce spočívala v negaci paralelního postulátu a následného výpočtu chování geometrických útvarů pomocí důkazu sporem. Saccheri tímto způsobem dospěl například k závěru, že přímka není nekonečná, ale má přesně stanovitelnou délku, což porušuje druhý Euklidův postulát. To mu však připadalo nemožné a nesmyslné, tak tento výsledek ve své publikaci zamítnul. Tento princip je nyní jedním z nejdůležitějších stavebních bloků eliptické geometrie, která je jedním z typů neeuklidovské geometrie.

Jeho kniha obsahuje další výpočty založené na podobném postupu, které později přispěly k tvorbě teorií o hyperbolické geometrii. Saccheri však pouze pár měsíců po vydání své knihy umírá, a ve své práci tudíž nepokračuje.

2.3. Kompletní pojednání

První kompletní pojednání o hyperbolické geometrii, což je jeden z typů neeuklidovské geometrie, vydávají mezi lety 1829 a 1932 nezávisle na sobě ruský matematik Nikolaj Ivanovič Lobačevskij a maďarský matematik János Bolyai. Po těchto vědcích se dále tento typ neeuklidovské geometrie nazývá Lobačevkijská nebo Bolyai-Lobačevkijská geometrie.

Zajímavé je, že Lobačevskij a Bolyai dospěli ke svým závěrům každý jiným způsobem. Zatímco Lobačevskij dospěl ke svému výsledku pomocí negace paralelního postulátu, čímž v podstatě dokončil Saccheriho práci, Bolyai vymyslel geometrii, kde se mohou vyskytnout prvky euklidovské i hyperbolické geometrie v závislosti na parametru k. Po srovnání výsledků ale bylo zjištěno, že oba došli k téměř shodným závěrům.

Obrázek 1 Nikolaj Ivanovič Lobačevskij (1)

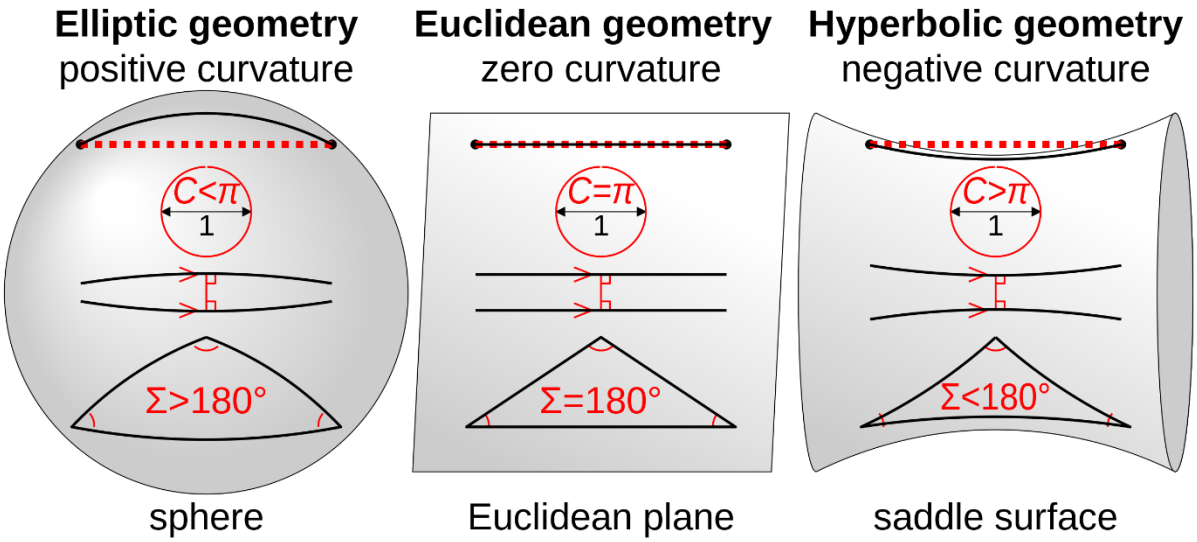
Obrázek 2 János Bolyai (2)

2.4. Ustálené modely

Během posledních několika desítek, až stovek let vzniklo několik ustálených dohodnutých modelů neeuklidovské geometrie, které jsou používány pro další výpočty a v dnešní době také počítačové simulace.

Mezi tyto modely patří právě například nejjednodušší a nejsnáze pochopitelný model eliptické geometrie, jímž je koule. Povrch této koule v tomto případě představuje „rovinu“ této geometrie a přímky jsou představovány hlavními kružnicemi, stejně jako třeba rovník na zeměkouli. To má mimo jiné za následek skutečnost, že trojúhelníky nakreslené na tomto modelu nemusí mít součet vnitřních úhlů 180° a přímky mají konečnou délku rovnou obvodu hlavních kružnic.

Další obecně uznávaný model je model již zmíněné hyperbolické geometrie, který poprvé předvedl v roce 1868 italský matematik Eugenio Beltrami. Tento model je již mnohem složitější, protože používá jako rovinu povrch hyperbolického tělesa. Beltrami představil nejdříve model založený na pseudokouli, která představuje zakřivení hyperbolického prostoru, a o rok později představil takzvaný Kleinův model, který již modeluje celý hyperbolický prostor.



Obrázek 3 Srovnání modelů neeuklidovské geometrie (3)

3. Historie neeuklidovských her

Zajímavostí her s prvky neeuklidovské geometrie je, že žádné z nich nemají samotnou neeuklidovskou rovinu implementovanou, ale pouze používají chytrých vizuálních triků, aby v hráči vyvolaly pocit, že ji doopravdy implementovanou mají. Toho se většinou docílí pomocí portálů, tudíž plochých objektů, které vyvolávají svou texturou pocit, že za nimi svět pokračuje, a po kontaktu s nimi vás přemístí na jiné místo. Takto lze docílit například toho, že v nějakém prostoru je místnost, která se tam velikostně nemůže vejít, a to proto, že tam fyzicky doopravdy není.

3.1. Portal

Portal je puzzle-plošinová hra z pohledu první osoby, vydaná v roce 2007 studiem Valve Software, které také vlastní světově největší herní platformu Steam. Tato hra se skládá převážně ze série puzzle pokojů a úkolem hráče je vymyslet, jak jimi projde. Toho musí dokázat pomocí takzvané portálové pistole, která dokáže střílet dva různé portály, kterými poté může hráč procházet, a přemisťovat objekty.

Hráč v této hře hraje za postavu jménem Chell, která je naprosto němá a neřekne během celé hry ani jediné slovo. Zato mnoho slov řekne GLaDOS, umělá inteligence, která Chell uvěznila v testovacím zařízení Apeture Science, kde se také celá hra odehrává. Hra má také samozřejmě složitý příběh, který hráč postupně plněním úkolů objevuje, a odehrává se ve stejném vesmíru, jako ostatní hry od Valve, například Half-Life, tudíž zapadá do celého velkého a propleteného scénáře.

Obrázek 4 Portálová pistole (Portal Gun) (4)

3.2. Antichamber

Antichamber je puzzle-plošinová hra z pohledu první osoby, vytvořená v roce 2013 australským vývojářem Alexandrem „Demruth“ Brucem. Většina puzzle v této hře je založena na fenoménu nemožných objektů vytvořených herním enginem, které v reálném světě nemohou existovat. Mezi tyto objekty patří například chodby, které hráče zavedou na několik různých míst podle toho, jakým směrem se dívá, a mnoho dalších podobných struktur.

Hráč v této hře ovládá bezejmenného a němého protagonistu, jehož hlavním cílem je dostat se pryč z testovacího zařízení Antichamber, ve kterém je bez vysvětlení důvodu uzavřen. Toho musí dosáhnout plněním různých puzzle hádanek, které naprosto popírají přírodní zákony, na které jsme zvyklí. Celý komplex také obsahuje množství laserů a jiné podobné techniky, která hráčům ztěžuje průchod různými místnostmi, a ještě přidává na složitosti hry.

3.3. Superliminal

Superliminal je surrealistická puzzle hra z pohledu první osoby, vydaná v roce 2019 studiem Pillow Castle Games. Většina mechanik této hry je postavena na optických iluzích a nucené perspektivě, zejména určité předměty mohou být hráčem přesunuty a změní přitom svou reálnou velikost, aby perspektivně vypadala stejně. Cílem hráče je projít množství místností, najít v každé východ a dostat se tak ke konci.

Hra má také velmi důležitou příběhovou stránku. Hráč podle ní vyrazí do Pierceova Institutu, ve kterém chce vyzkoušet jejich technologii SomnaSculpt, která se využívá pro terapie pomocí snů. Němý protagonista je tedy napojen na zařízení a uspán, načež se probere v testovacím středisku vytvořeném ve světě jeho snů, něco však není správně. Kontrolní středisko naprosto ztratí hráčovu pozici a poradí mu, aby se z této situace dostal sám. Na konci hry se ale ukazuje, že k žádné závadě nedošlo a všechno bylo pouze součástí terapie. Hráč byl podle profesora Pierce donucen podívat se na problém, který řešil, z jiné perspektivy, a to mu pomohlo se psychicky posunout.

4. Obsah mé hry

Má hra je puzzle-příběhová hra s protagonistou, který provádí hráče celým zážitkem pomocí svých podvědomých myšlenek. Co se hratelnosti a technik týče, tak je má hra v podstatě spojením všech dosud zmíněných, a zapadá také do stejného žánru.

4.1. Příběh

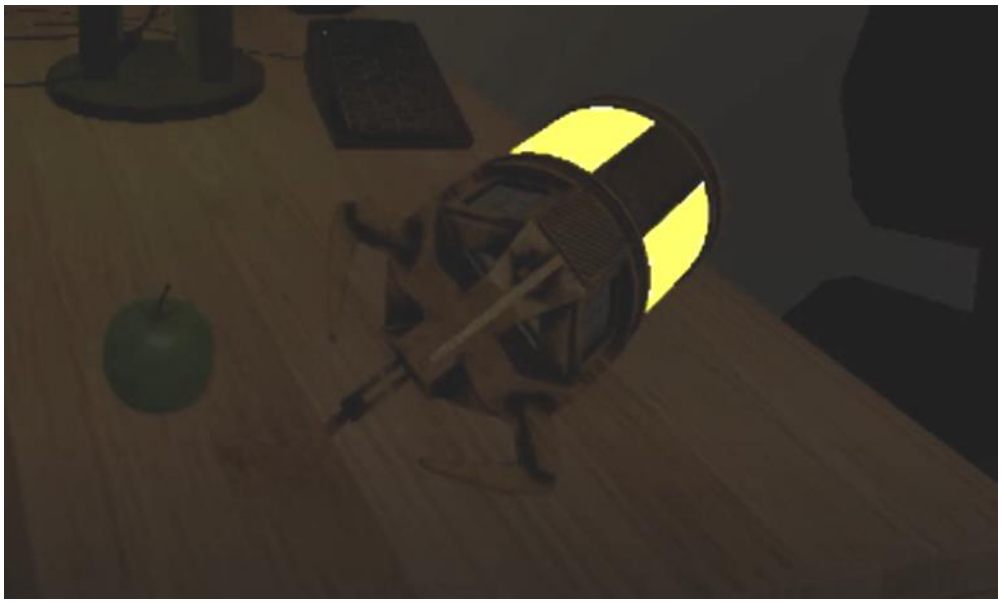
Má hra začíná ranním příchodem protagonisty do školy. Protagonista na začátku již zmiňuje, že si špatně nastavil budík a že je ve škole mnohem dříve, než by měl být. Poté, co se dostane do své učebny a posadí se, uvědomí si, že je velmi unavený a že první hodina začíná až za čtyřicet minut, lehne si tedy na lavici a usne. Vzbudí ho až hlasitý úder blesku, po němž si vystrašený protagonista všímá, že je již venku úplná tma a že nejspíš nějakým způsobem prospal celý školní den. Rozhodne se tedy, že musí školu ihned opustit.

Vyjde ze dveří a vydá se po schodech dolů k východu, v tuto chvíli si ale všimne první anomálie. I když je v prvním patře a od přízemí ho dělí pouze pár metrů, není schopen se do něj dostat, protože pokaždé, když sejde jedno poschodí, tak skončí v tom samém patře. Takhle zkusí sejít několik pater, než koutkem oka všimne něčeho, co ho velice vystraší – temné entity levitující asi metr nad zemí pouze kousek od něho.

Obrázek 5 Temná entita – manuálně zesvětleno

Naneštěstí si entita hráče všimne, a ten se dá ihned na útěk. Nemá důvod utíkat po schodech nahoru, protože by pouze doběhl na to samé místo, kde by na něj už entita čekala, tak se rozhodne pokusit se dostat k druhému poschodí. Když ale zkusí otevřít dveře, které k tomuto poschodí vedou, zjistí, že jsou zamčené a je nucen najít odemčenou učebnu, ve které by se mohl schovat. Jedinou odemčenou učebnou je učebna matematiky, ve které na něj ale čeká další překvapení.

Po vstupu do této učebny si hráč všimne zvláštně vypadající zbraně, která je tu na stole položena. Otestuje ji na jablku, které najde, zjistí, že dokáže měnit velikost objektů, a napadne ho tedy zkusit entitu zmenšit a zabránit tak, aby mu ublížila. V tu chvíli ale naráží na další problém, dveře, které za sebou před chvílí zabouchl, jsou nyní zaseknuté a nejdou otevřít. Může je ale jednoduše vysadit z pantů pomocí svého nového nálezu, což také udělá a vyrazí chodbou hledat entitu, aby se jí mohl postavit.



Obrázek 6 Perspektivní manipulátor – nalezená „zbraň“

Entitu po malé chvíli nalezne a pokusí se ji svým nástrojem zmenšit, nic se ale nestane a protagonista je nucen se rozběhnout a prorazit ramenem zamčené dveře, kterými se předtím chtěl dostat k druhému schodišti. Když se ale urychleně zvedne ze země, je naprosto v šoku, protože právě vidí druhou anomálii.

Na druhé straně dveří spatří tu samou místnost, ze které právě přiběhl, a dochází mu, že nemá, kam před entitou utéct. Běží dále nekonečným, stále se opakujícím se tunelem těch samých místností, ze kterých by se tak chtěl dostat. Při běhu ho napadne zkusit se schovat na jeden ze záchodů, ale to mu nečekaně také nepomůže. Když otevře jedny dveře od záchodu, otevřou se i ty druhé, a po průchodu jedněmi z nich se hráč opět dostane na to samé místo, ze kterého právě vyšel. Napadá ho poslední možnost, a to přetížit svou zmenšovací zbraň pomocí elektrického panelu, který najde poblíž, a zkusit entitu znovu zmenšit. Přetíží ji však moc, což způsobí výbuch a protagonista se probudí v bílé nicotě.

Po probuzení se nejdříve zmatený protagonista dívá kolem sebe a přemýšlí, kde může být, v tom si ale všimne, že entita je s ním v tomto prostoru také, a leknutím trochu uskočí. Hned si ale všimne, že entita se za ním nepohybuje, pouze tam levituje, jako by byla zmražená. V tu chvíli mu začne docházet, co se vlastně stalo. Nikdy se neprobudil, tohle celé se mu pouze zdá a jenom už začíná bláznit z příliš velkého množství stresu, kterému byl během posledního roku vystaven. Protagonista si ale uvědomí, že vše, co ho během toho posledního roku stresovalo, je už v tuto chvíli dokončeno, a rozloučí se s temnou entitou, kterou nazve starým přítelem, a ta odletí do dálky.

I když je možné si příběh interpretovat mnoha možnými způsoby, jeho hlavní interpretací je, že temná entita představuje všechen stres a negativní emoce, které zažívají žáci při přípravě na maturitu, a má odrážet jejich psychický stav, který je v těchto momentech nestabilní a může je snadno pohltit do temnoty. Odpočinek a uvolnění napětí přichází až ve chvíli, kdy maturitu dokončí.

Obrázek 7 Entita opouštějící protagonistu na konci hry

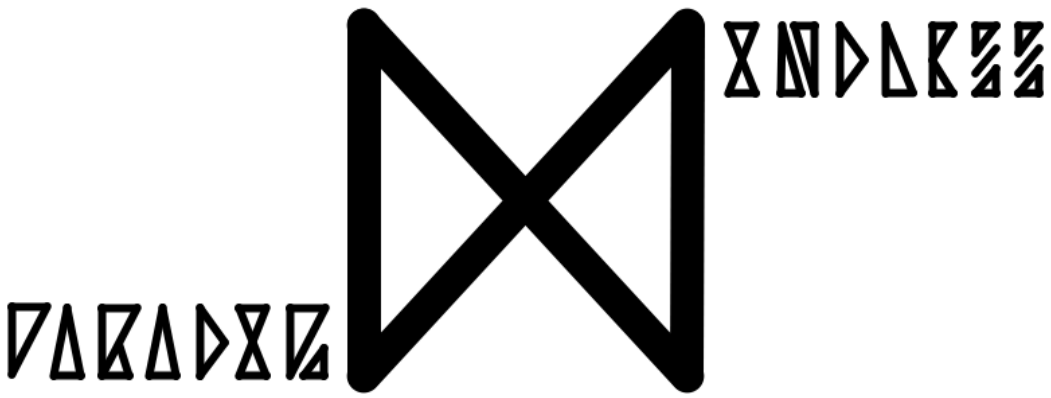
4.2. Název hry

Dlouhou dobu jsem také přemýšlel nad tím, jaký chci dát své hře název. Chtěl jsem, aby název zněl částečně futuristicky, ale aby měl návaznost na příběh a pocit, který má hra v hráči po jejím dohrání zanechat. Nakonec jsem se rozhodl pro název „Mindless Paradigm“, což by se dalo do českého jazyka přeložit jako bezduché paradigma.

Tento název má ukazovat, jak jsme všichni naučeni bezhlavě následovat, co nám naši nadřízení připravili, a nevymýšlet si svoji vlastní cestu. Paradigma znamená obecně přijímané schéma či vzorec myšlení, a to je v tomto případě nutnost udělat maturitu. Ve společnosti je maturita brána jako ukazatel alespoň základního dobrého vzdělání a bez ní by měl člověk problém najít si dobře placené zaměstnání, i když je třeba v něčem lepší než lidé, kteří maturitu mají. Nejedná se o něco, o čem by se člověk rozhodoval, jestli chce, nebo ne, ale jedná se o něco, co musí člověk v této společnosti bezhlavě následovat, ať ho to stojí cokoliv. Třeba i jeho psychické zdraví, a to není dobré.

4.3. Logo

Logo mé hry bylo vytvořeno se stejnou myšlenkou jako její název – aby vypadalo futuristicky a zároveň mělo jistou podobnost s příběhem, a to se mi celkem podařilo.



Obrázek 8 Logo Mindless Paradigm

5. Použité vývojové nástroje

5.1. Godot

Godot je 2D a 3D multiplatformní open source herní engine pod licencí MIT vyvinutý komunitou, byl interně používán v několika společnostech v Latinské Americe, předtím, než byl uvolněn jako open-source a zpřístupněn veřejnosti. Vývojové prostředí běží na Windows, macOS a Linux (oba 32 a 64 bit) a může vytvářet hry cílené na PC, konzole, mobily a web. (2)

5.2. Visual Studio Code

Visual Studio Code je editor zdrojového kódu vyvíjený společností Microsoft pro operační systémy Windows, Linux a macOS. Obsahuje podporu pro Git (a pro GitHub), zvýraznění syntaxe, kontextový našeptávač a podporu pro ladění a refaktorizaci. Zdrojový kód je svobodný software pod licencí MIT. Sestavené verze nabízené přímo Microsoftem jsou freewarem obsahujícím telemetrii, ale existuje i komunitně sestavovaná varianta VSCodium. Editor je naprogramovaný v JavaScriptu a TypeScriptu. (3)

5.3. Blender

Blender je svobodný a otevřený software pro třírozměrnou počítačovou grafiku, který se používá k vytváření animovaných filmů, vizuálních efektů, uměleckých děl, třírozměrných tištěných modelů, pohyblivé grafiky, interaktivních třírozměrných aplikací, virtuální reality a dříve i videoher. Mezi funkce programu Blender patří třírozměrné modelování, animace, motion capture, renderování, pohyblivá grafika, střih videa a kompozice, texturování, digitální kreslení, editace rastrové grafiky, dopředná a inverzní kinematika, simulace tekutin a kouře, simulace částic, simulace měkkých těles, 3D sochařství. (4)

5.4. Paint.NET

Paint.net (Paint.NET nebo paint.net) je freewarový program pro rastrovou grafiku pro Microsoft Windows, vyvinutý na platformě .NET Framework. Paint.net původně vytvořil Rick Brewster jako studentský projekt Washingtonské státní univerzity a z jednoduché náhrady programu Microsoft Paint se vyvinul v program především pro úpravu grafiky s podporou pluginů. (5)

6. Prvky neeuklidovské geometrie

6.1. Portály

V mé hře využívám techniky portálů velmi extensivně, ne však stejným způsobem jako vývojáři hry Portal. Zatímco ve hře Portal jsou portály vytvořeny tak, aby bylo vidět, že se o portály jedná, v mé hře jsou portály vytvořeny tak, aby bylo co nejméně poznat, že tam jsou. Pomocí portálů jsou v mé hře vytvořeny iluze nekonečné chodby a dveří, které vedou na místa, kam vést nemohou, protože by tam na místnosti na druhé straně nebylo místo.

I když by bylo možné najít i jiný způsob, v Godotu se portály nejsnáze dělají pomocí takzvané Viewport texture. Viewport texture je v Godotu texturový soubor, který při spuštění hry mění svůj obsah na základě toho, co ve hře vidí vybraná kamera. Tudíž pokud chci vytvořit portál, přidám do scény stěnu, které nastavím jako texturu tuto Viewport texture, a kameru, kterou dám na cílovou destinaci portálu. Poté pomocí krátkého skriptu zařídím, aby se kamera hýbala a otáčela stejně jako hráč, a ve svém portálu již vidím druhou stranu, kam se dostanu, když portálem proběhnu.

V tuto chvíli sice vidím, co na druhé straně portálu je, ale ještě mě tam portál nepřemístí, na to potřebuji další skript. Tento skript kontroluje každý tick fyzického enginu, jestli se pozice hráče nenachází za povrchem portálu, a pokud ano, tak jej přemístí na druhou stranu. Pro portály, ze kterých hráč nevychází stejným směrem jako vchází, je ještě nutné příslušně změnit hráčovu rotaci a přepočítat jeho lineární rychlost, aby v případě, že do portálu skočí, vyletěl na druhé straně stejnou rychlostí.

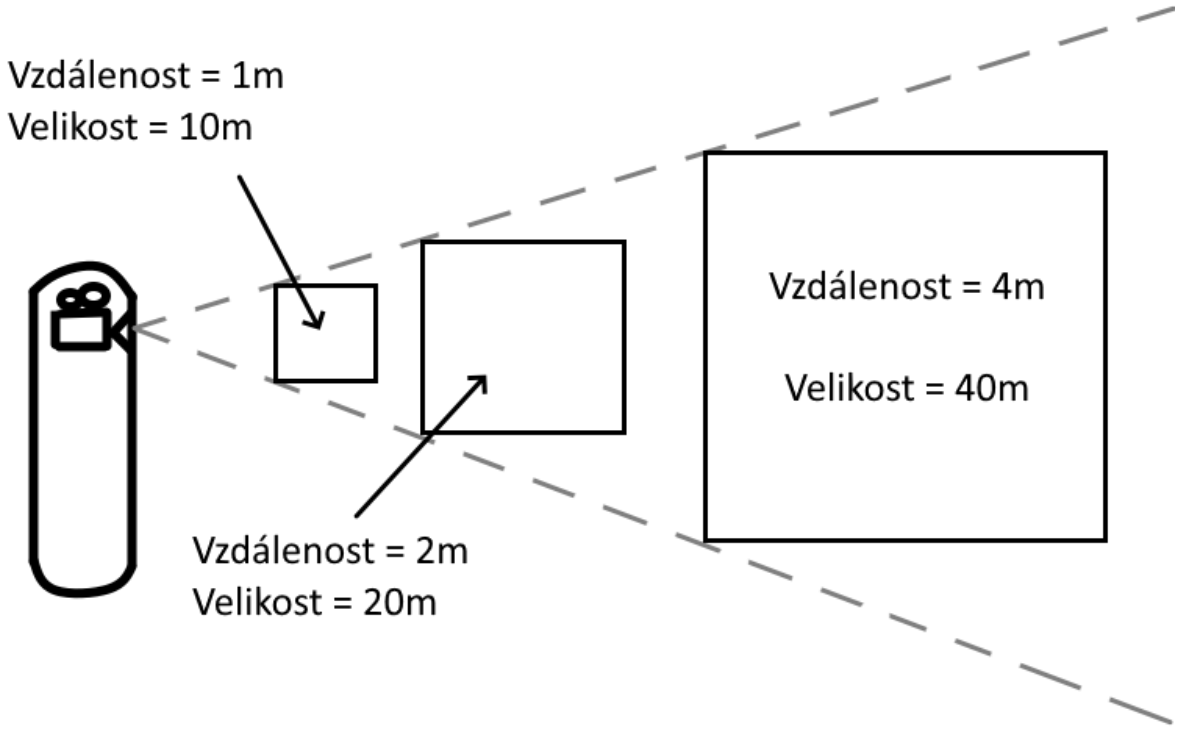
```
subViewport.Size = new Vector2I(GetViewport().GetWindow().Size.X / 2, GetViewport().GetWindow().Size.Y / 2);
camera.GlobalRotationDegrees = new Vector3(player.GlobalRotationDegrees.X, player.GlobalRotationDegrees.Y + 180, 0);
cameraRig.GlobalPosition = new Vector3(GlobalPosition.X - 0.1f - (player.GlobalPosition.X - GlobalPosition.X + 0.1f),
player.GlobalPosition.Y, GlobalPosition.Z - (player.GlobalPosition.Z - GlobalPosition.Z) + z);
if (player.GlobalPosition.X /*+ 0.1*/ > cameraRig.GlobalPosition.X && player.GlobalPosition.Z < GlobalPosition.Z +
Size.Z / 2 && player.GlobalPosition.Z > GlobalPosition.Z - Size.Z / 2) {
    Vector3 tmp = cameraRig.GlobalPosition;
    tmp.Y -= 0.5f;
    Vector3 playerTemp = player.GlobalPosition;
    //tmp.X -= 0.01f;
    playerBody.GlobalPosition = tmp;
    cameraRig.GlobalPosition = playerTemp;
    playerBody.Velocity = new Vector3(-playerBody.Velocity.X, playerBody.Velocity.Y, -playerBody.Velocity.Z);
    Vector3 rot = playerBody.GlobalRotationDegrees;
    rot.Y += 180;
    playerBody.GlobalRotationDegrees = rot;
```

Obrázek 9 Kód pro přemisťování hráče mezi portály

6.2. Perspektivní škálování

Na technice perspektivního škálování a nucené perspektivy je celá založená hra Superliminal, kterou jsem již dříve zmínil. V mé hře sice nezabírá takovou část, ale stále se jedná o jeden z nejdůležitějších prvků, který byl také velmi těžký na implementaci.

Perspektivní škálování v podstatě spočívá v tom, že na monitoru vnímají lidé stejně 3 metry široký předmět, který je jeden metr daleko, jako 30 metrů široký předmět, který je vzdálený deset metrů. Toto mi umožňuje nastavit mou hru tak, že když vezme hráč předmět, tak se posune, co nejdále to jde, a co nejvíce se zvětší, ale hráč si toho ani nevšimne. To vyvolává ve hráči pocit zmatení a nutí ho přemýšlet kromě umístění také o velikosti objektu.



Obrázek 10 Diagram perspektivního škálování

Pro implementaci této techniky do mé hry používám takzvaný raycasting neboli vystřelování paprsků. Když chci vystřelit paprsek, musím určit jeho počáteční pozici, jeho rotaci a jeho délku. U svého hlavního paprsku mám jako počáteční pozici a rotaci nastavenou momentální pozici a rotaci hráčovy kamery, a jako délku mám nastavených 100 metrů, protože delší paprsek potřebovat nikdy nebudu. Paprsek se vystřeluje každý fyzický tick, a když něco trefí, tak dostanu informace o tom, jaký objekt trefil, jak je daleko nebo také jeho velikost.

Ve chvíli, kdy tento paprsek zasáhne fyzický objekt, který je možné přesouvat, změní ikonu hráčova ukazatele na malou ruku, a ten má poté možnost ho zvednout. Ve chvíli, kdy hráč zvedne objekt, se do proměnné *mult* uloží velikost objektu dělená vzdáleností objektu od hráče, což je pro tento proces velmi důležité číslo. Nyní, když je objekt zvednut, se musí každý fyzický tick stát několik různých věcí:

1. Program najde místo, kam se paprsek trefil, a umístí objekt polovinu šířky tohoto objektu před něj, aby nebyl z poloviny ve stěně
2. Nyní spočítá vzdálenost tohoto objektu od hráče a uloží si ji do proměnné
3. Poté nastaví velikost objektu na obsah proměnné *mult* vynásobený spočítanou vzdáleností

Toto by v ideálním případě, tudíž že přesouvaný objekt je koule, stačilo, můj kód jde ale ještě o krok dál a střílí každý fyzický tick další 4 paprsky. Každý z těchto paprsků je namířen na jeden ze zadních rohů boxu vytvořeného kolem přesouvaného objektu a při posouvání objektu kontroluje, jestli není tento roh zaseknutý ve stěně. Pokud tomu tak je, tak objekt posune směrem k hráči o rozdíl délky tohoto paprsku a hlavního paprsku, což způsobí, že pokud hráč objekt pustí, tak nepropadne stěnou.

Nyní už je objekt skoro úspěšně přesunut, zbývá ale ještě jeden důležitý krok. V tuto chvíli by mohl hráč objekt pustit uvnitř svého těla a nechat se jím „promáčknout“ skrze stěnu do místnosti, kam ještě nemá mít přístup, což by rozbilo průběh herní sekvence. Pro vyřešení tohoto problému používám funkci Contact monitor, která dokáže zkontrolovat, jestli spolu tělo hráče a objektu nekolidují. Tato funkce je moc výpočetně náročná na to, aby se spouštěla každý fyzický tick, tak se spouští pouze když chce hráč objekt pustit. Pokud v tuto chvíli objekt koliduje s hráčovým tělem, tak mu ho hra nedovolí pustit a nic se nestane.

```
if (dist < GetCollisionPoint().DistanceTo(GlobalPosition))
{
    float diff = GetCollisionPoint().DistanceTo(GlobalPosition) - dist;
    toCarry.GlobalPosition = GetCollisionPoint() + (GlobalPosition - GetCollisionPoint()).Normalized()
    * (diff + Math.Max(box.Scale.X, Math.Max(box.Scale.Y, box.Scale.Z)) / 2);
    shape.Scale = new Vector3(1f, 1f, 1f) * toCarry.GlobalPosition.DistanceTo(GlobalPosition) * mult;
    box.Scale = new Vector3(1f, 1f, 1f) * toCarry.GlobalPosition.DistanceTo(GlobalPosition) * mult;
}
else
{
    toCarry.GlobalPosition = GetCollisionPoint() + (GlobalPosition - GetCollisionPoint()).Normalized()
    * (Math.Max(box.Scale.X, Math.Max(box.Scale.Y, box.Scale.Z)) / 2);
    shape.Scale = new Vector3(1f, 1f, 1f) * toCarry.GlobalPosition.DistanceTo(GlobalPosition) * mult;
    box.Scale = new Vector3(1f, 1f, 1f) * toCarry.GlobalPosition.DistanceTo(GlobalPosition) * mult;
}
```

Obrázek 11 Nejdůležitější část škálovacího kódu

7. Skripty

V této části dokumentace bych rád představil ty nejdůležitější skripty, které ovládají celý průběh hry.

7.1. sequencer.cs

Sequencer je ten naprosto nejdůležitější skript v celém mém projektu. Ve všech mých minulých projektech fungoval průběh příběhu hry na základě volání metod v jiných skriptech při dokončení minulé části, což vždy vytvářelo velké množství „spaghetti codu“ a obecně tvořilo kód v podstatě nečitelným. Toto jsem se rozhodl ukončit, a proto jsem se rozhodl přenechat celé řízení průběhu hry tomuto skriptu.

Sequencer obsahuje dvě public proměnné, int *seqNum*, který určuje momentální sekvenci, a *bool nextSeq*, který určuje, jestli se má aktivovat další sekvence. Sequencer každý fyzický tick kontroluje, jestli nezměnil nějaký venkovní skript *nextSeq* na true, a pokud ano, tak načte sekvenci, jejíž číslo je nastavené v proměnné *seqNum*. Toto je užitečné třeba v případě, že entita chytí hráče. Když se to stane, tak venkovní skript entity nastaví *seqNum* na poslední checkpoint a změní nextSeq na true, což vrátí hráče a celý stav hry na stav v posledním checkpointu.

Tento skript dále obsahuje také všechny titulky a instrukce, kdy je spouštět, které jsou vždy při načtení nové sekvence uloženy do pole *subtitleArray* na index, který určuje čas spuštění. Kromě titulek ještě spouští také animace a jiné události, které se během hry odehrávají, jako jsou například zvuky a hudba.

```
case 16:
    GetNode<music>("/root/Music").Stream = (AudioStream)GD.Load("res://Audio/end.mp3");
    GetNode<music>("/root/Music").Play();
    GetNode<AnimationPlayer>("/root/Root/CharacterBody3D/PlayerAnimator").Play("anim_15");
    subtitles = GetNode<subtitles>("/root/Root/CanvasLayer/Subtitles");
    informative = GetNode<informative>("/root/Root/CanvasLayer/Informative");
    subtitleArray = new string[2800];
    subtitleArray[0] = "Where am I...";
    subtitleArray[549] = "Woah woah.....";
    subtitleArray[699] = "why are you not attacking me?";
    subtitleArray[999] = "I get it now...I never woke up, did I?";
    subtitleArray[1399] = "We're inside my head and I'm slowly going crazy from all the pressure";
    subtitleArray[1899] = "But now it's all done....";
    subtitleArray[2199] = "the books, the exams, this GAME....it is all finished!";
    subtitleArray[2499] = "You are free to go now old friend.....";
    subtitleArray[2799] = " ";
    time = 0;
    break;
```

Obrázek 12 Příklad nastavení sekvence

7.2. perspective_raycast.cs

Tento skript jsem již v této dokumentaci zmínil v části o perspektivním škálování, tam však jeho využití zdaleka nekončí. Tento skript se stará o všechny interakce hráče s okolním světem, jako například otevírání dveří nebo sedání na židli. Když hráč najede kurzorem na objekt, se kterým může nějak interagovat, tak se kolečko kurzoru změní na ikonu klikající ruky. Když hráč v tuto chvíli klikne, tak tento skript zavolá všechny potřebné funkce, aby se ve hře stalo to, co se stát má. Velmi často spouští další sekvence Sequenceru.

Dále má tento skript spolu se skriptem portal.cs ještě jednu nakonec nevyužitou funkci, kterou bylo paradoxně skoro nejsložitější implementovat, a to přepočítávání paprsků. To umožňuje brát a pokládat předměty i skrze portály, což by bez této funkce nebylo možné, protože by se portál po zasažení paprskem choval stejně jako klasická stěna.

7.3. character_controller.cs

Dalším velmi důležitým skriptem je Character controller, který zajišťuje hráči možnost se pohybovat. Tento skript kontroluje každý fyzický tick, jaké klávesy jsou stisknuty, a z toho spočítá a aplikuje hráčovu konečnou lineární rychlost.

Pro pohyb se v mé hře využívá rozložení pohybových kláves WASD, které se v posledních několika letech stalo standardem pro hry. Dále, pokud se hráč nachází na zemi, je možné vyskočit stiskem tlačítka mezerníku, a pokud je již hráč dále v průběhu hry, tak může zvýšit svou rychlost běhu podržením klávesy Shift. Skript také dále mění hráčovu rychlost na základě toho, jestli stojí nohama na zemi, nebo se pohybuje ve vzduchu.

Tento skript se také dále stará o otáčení hráčova těla a kamery podle pohybů myši, spouštění zvuků chůze a spouštění animace pohupování hlavy při chůzi. A když se přehrává nějaká animace, tak také všechny své funkce dočasně zablokuje, aby nedocházelo ke zvláštním kombinacím animací a pohybů.

8. Modely a textury

Problém příběhových 3D her, jako je ta moje, je, že obsahují opravdu velké množství různých 3D modelů. Ať už se jedná o celý velký stůl nebo jenom zmačkanou plechovku vedle koše, tento model musel být nejdříve vymodelován, texturován a následně ručně umístěn na jeho určené místo. Velká studia jako například Rockstar Games nebo Valve Software řeší tento problém dvěma různými způsoby. Buďto si najmou velký tým grafiků a profesionálů na vizuální efekty, kteří toto zvládnou relativně rychle, to je ale velice drahé a pro takto malý projekt nemožné.

Druhý způsob, jakým toto velké společnosti řeší, je outsourcování této práce nějaké jiné firmě nebo platformě, jejíž uživatelé rádi pokryjí alespoň modelování a texturování objektů. Problémem tohoto přístupu může být opět cena, která se může pohybovat klidně i kolem několika stovek korun za jeden model, naštěstí tomu tak ale nutně být nemusí. Platformy jako Sketchfab, CGTrader nebo třeba TurboSquid nabízí velké množství bezplatných 3D modelů, které mohou tvůrci volně využít pro své projekty, včetně těch komerčních.

8.1. Vlastní modely

V mé hře se nachází velké množství vlastních modelů, které jsem musel vytvořit čistě pro potřeby této hry. Mezi ně zapadá například velké množství cedulí a nástěnek, které se nachází v chodbě a které se na internetu prostě najít nedají. Tvorba každé této cedule vyžadovala hned několik kroků:

1. Vyfotit tuto ceduli ve škole a uložit ji na místo, kde se neztratí
2. Upravit tuto fotografii, aby cedule byla rovně a byla rovnoměrně nasvícena
3. Vytvořit v Godotu plochu velikosti cedule a umístit jí na cílové místo
4. Nastavit upravenou fotografii jako texturu plochy a upravit hodnoty jako kovovost a drsnost, aby cedule vypadala stejně jako v reálném světě

Toto jsou však pouze kroky, které je potřeba udělat vždy. Některé cedule, jako třeba velké nástěnky, potřebovaly ještě další kroky. Mezi ty patří například odstraňování pozadí fotografie, modelování okrajů cedule nebo tvorba normálové mapy, která zařídí, že plocha bude správně odrážet světlo.

Dalším vlastním modelem je třeba celý prostor školy, který nebyl skenován, ale pouze změřen a následně ručně vymodelován. Nějaké textury tohoto prostoru byly staženy ze stránky polyhaven.com, která je nabízí zdarma pod licencí CC0 1.0 DEED, která zbavuje tvůrce všech autorských práv a povoluje volné nakládání s obsahem.

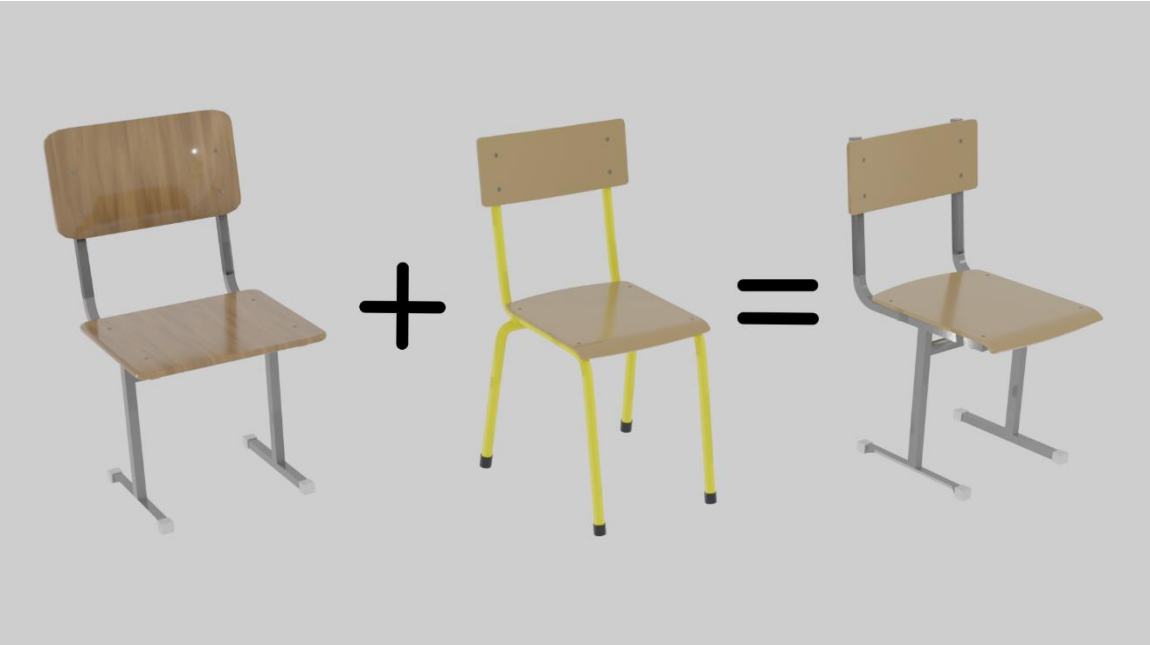
8.2. Sketchfab

Některé složitější modely v mé práci, jako například židle a stoly, byly outsourcovány ze stránky sketchfab.com, která je nabízí pod licencí CC BY 4.0 DEED. Ta umožňuje jejich volnou úpravu a využívání i v komerčních projektech, pokud je uveden jejich původ.

To, že jsou nějaké modely stažené ze Sketchfab, ale přináší své vlastní problémy. Prvním z nich je kvalita modelů, která je často příliš vysoká pro využití v podobných hrách a je určena spíše pro využití ve filmech a animovaných videích. Dalším problémem je velké množství různých formátů, často se mi totiž při vývoji stávalo, že model byl původně vytvořen ve formátu, který Godot nepodporuje, a musel jsem ho přeformátovat pomocí Blenderu.

Modely ze Sketchfab jsem musel také velmi často upravovat, protože v jejich původním stavu pro mne většinou nebyly použitelné. Stávalo se například, že byly moc velké, moc malé, měly špatnou barvu, nebo dokonce měly texturu vytvořenou v úplně špatném formátu a musel jsem si ji celou vytvořit sám.

Ačkoliv mi Sketchfab určitě práci velice ušetřil, také mi mnoho práce přidal. Tímto různým upravováním a přebarvováním modelů jsem strávil hodiny a hodiny práce, které bych jinak mohl využít pro zlepšení své hry a které byly v podstatě zbytečné. Například u již zmíněných židlí jsem musel spojit dva různé modely židlí dohromady, protože jedna měla špatně vymodelované nohy a druhá zase sedátko.



Obrázek 13 Spojení modelů židle

9. Osvětlení

Realistické osvětlení je jednou z nejsložitějších částí vývoje 3D her, protože vývojáři si musí v podstatě vybrat mezi dvěma různými problémy – nerealisticky vypadající osvětlení, nebo vysoké výkonnostní nároky. Většina projektů s realistickým osvětlením, včetně toho mého, se snaží trefit někam doprostřed, tudíž vytvořit relativně realisticky vypadající osvětlení, které ale poběží většině hráčů s průměrně dobrými herními počítači.

9.1. Ray-tracing

Existuje velké množství různých technologií, které se snaží tohoto dosáhnout s co nejméně kompromisy. Během posledních pár let se čím dál častěji objevují hry, které mají podporu pro takzvaný ray-tracing. Ray-tracing je technologie, která se využívá pro realistickou simulaci chování světla v prostoru za pomoci samostatně simulovaných paprsků a která podporuje mnoho různých optických efektů, jako je odraz a lom světla, hloubka ostrosti a různé druhy stínů. Donedávna bylo používání ray-tracingu v reálném čase považováno za nemožné kvůli jeho velkému dopadu na výkon, to se ale změnilo v roce 2019 s příchodem standardizované hardwarové akcelerace této techniky do grafických karet. Tyto karty ale stále mnoho lidí nemá, a proto jsem nakonec techniku ray-tracing ve své hře nevyužil.

9.2. VoxelGI

Technika, kterou jsem se nakonec rozhodl ve své hře využít, se nazývá Voxel Global Illumination neboli VoxelGI. Voxel je v podstatě to samé jako pixel, ale v 3D prostoru, stejně jako krychle je prostorový čtverec. VoxelGI funguje na konceptu předpočítání chování světla v jednotlivých zónách voxelů, jejichž velikost určuje konečnou přesnost osvětlení, ale také jeho dopad na výkon a velikost předpočítaných světelných map. Tím, že se tyto informace spočítají ještě před spuštěním hry, se masivně sníží dopad generace osvětlení na výkon grafické karty, což ve výsledku umožní hru spustit i hráčům s méně výkonnými počítači.

VoxelGI ale není kouzelné a má také mnoho nevýhod. Jeho hlavní nevýhodou je jeho nepřesnost. U tenčích stěn a povrchů často propouští světlo i přes pevné materiály, což je v mé hře hodně vidět přes den v hlavní chodbě, a také někdy vytváří neexistující stíny, které jsou původcem lehkého probliknutí, když hráč vchází do jiného patra. VoxelGI také oproti ray-tracingu postrádá možnost simulovat různé pokročilejší fyzikální vlastnosti světla, jako třeba právě dříve zmíněný lom světla a hloubku ostrosti.

10. Problémy při vývoji

Tuto kapitolu bych rád věnoval těm nejhorším problémům, na které jsem během svého vývoje narazil. I když jsem si za mnohé z nich mohl sám, narazil jsem také na mnoho problémů, které mnou nebyly způsobeny, a většinou byly způsobeny právě mnou využitým herním enginem Godot 4.

10.1. Neznámé prostředí

Jedním z mých hlavních problémů byla má nezkušenost právě s tímto herním enginem. Před začátkem vývoje této hry jsem všechny své hry, ať už komerční, či ne, dělal v herním enginu Unity, který mi bylo, i přes jeho podobnost s Godotem, naprosto nesmyslně zakázáno použít. Právě podobnost obou enginů mi sice masivně pomohla v tom, abych se naučil používat Godot, ale také mne často mátla, protože Godot postrádá mnoho dobrých funkcí, které pro mne byly v Unity samozřejmostí.

10.2. Výkon

Další velký problém, na který jsem během vývoje narazil, byl výkon vykreslovacího prostředí Forward+, které Godot pro počítačové hry využívá. Původně jsem například ve své hře využíval úplně jiné systémy pro osvětlení, které se nazývají SSIL a SDFGI, ale nebyl jsem schopný s jejich pomocí dosáhnout uspokojivých výsledků. Obraz buďto nevypadal dostatečně realisticky, nebo hra běžela i na mém relativně dobrém herním počítači jako na bramboře.

Dalšího problému s výkonem jsem si všiml, když jsem do své hry přidal jistý částečně průhledný předmět, který svou míru průhlednosti každý fyzický tick měnil. Ať jsem použil jakýkoliv osvětlovací systém, tak tento předmět srazil výkon celé hry na kolena, i když ho hráč jenom zahlédl koutkem oka. Jsem si celkem jistý, že se jedná o bug v samotném enginu, protože nakonec jsem pro stejný efekt využil jiný typ objektu, který také takto měnil míru průhlednosti, a s tím už nebyly další problémy.

10.3. Spuštění kódu před vykreslením

Poslední velký problém, na který jsem narazil, byla absence funkce pro spuštění kódu před vykreslením dalšího snímku. Stejně jako ostatní podobné enginy obsahuje Godot funkci *Update*, která se volá pokaždé, když hra aktualizuje snímek na obrazovce, neobsahuje ale funkci, která by se zavolala před vykreslením snímku. Například Unity tuto funkci má, což by mi hodně ušetřilo práci. Nevyužití této funkce způsobovalo problikávání displeje, když hráč prošel portálem, což se mi podařilo částečně zlepšit jiným způsobem, který ale není zdaleka perfektní a má vyšší dopad na výkon.

11. Instalace

Mou hru je možné nainstalovat hned dvěma různými způsoby. Tím snazším je stažení již sestaveného projektu ze záložky *Releases* tohoto repositáře a tím složitějším je samostatné sestavení projektu ze zdroje.

11.1. Stažení sestaveného projektu

Sestavený projekt si můžete stáhnout ze záložky *Releases* na stránce Git repositáře https://github.com/gyarab/2023-4e-masek-neeukleidovska_hra ve formátu zip. Stažený archiv je nutné před spuštěním rozbalit. Jsou dostupné sestavené verze pro operační systémy Windows a Linux.

11.2. Sestavení ze zdroje

Pro sestavení projektu ze zdroje budete nejprve potřebovat klon Git repositáře projektu z https://github.com/gyarab/2023-4e-masek-neeukleidovska_hra a dále také Godot Engine .NET verze 4.2.1 nebo novější a .NET SDK verze 7.0 nebo novější. Naklonovaný projekt otevřete v Godotu a sestavte pomocí tlačítka *Export* v záložce *Project*. Godot vytvoří tři soubory, spustitelný soubor pro spuštění hry, spustitelný soubor pro spuštění hry v debugovacím modu a složku s daty. Po kliknutí na první spustitelný soubor se hra spustí, ten ale musí být ve stejné složce jako složka s daty.

12. Zhodnocení a závěr

Cílem mé práce bylo vytvořit puzzle hru s prvky neeuklidovské geometrie, která vyzkouší rychlé přemýšlení lidí pod tlakem, a s naprostou upřímností mohu říci, že si nedovedu představit, jak bych ještě mohl svou práci vylepšit, aby lépe splňovala své zadání. Hra samozřejmě obsahuje menší množství chyb, které bych mohl opravit, těch si ale běžný hráč většinou ani nevšimne, tudíž jsem to nezařadil do svého vývojového plánu. Také by hra mohla být trochu lépe optimalizována nebo mít delší příběh, už takto ale sahá její rozsah za hranice uspokojivé maturitní práce.

Nechci zde ale hodnotit pouze výsledek svého snažení, ale také cestu k němu a vše, co jsem se na ní naučil. Naučil jsem se pracovat s novým herním enginem, úspěšně během vývoje vyřešil velké množství chyb, což mne jako programátora posunulo dále, ale hlavně jsem si vybral téma, co mě už předtím zajímalo, a tudíž mě bavilo i na něm pracovat. Asi bych si vývoj své hry užil více, kdybych nemusel zároveň s ním pracovat na velkém množství jiných věcí k maturitě, ale to k tomu prostě patří.

Také jsem si velice užil vymýšlení příběhu, které mi také zabralo pár hodin. Chtěl jsem vytvořit hru, kterou si může člověk zahrát bez toho, aby si nějakého příběhu vůbec všiml, ale aby mohl důsledný hráč najít i hlubší význam. Nakonec jsem se rozhodl ve své hře poukázat na psychické zdraví žáků během příprav k maturitě, ať už to mé, nebo někoho z mnoha dalších. Jedná se o důležitý problém, který se v dnešní době řeší různými doktory a psychology, i když by se dal snadno vyřešit už v jeho úplném základu. Stačilo by, aby toho studenti neměli tolik najednou. Kdyby mělo více předmětů možnost nahradit jejich maturitu nějakým certifikátem, který by měl žák možnost udělat kdykoliv během studia a klidně na několik pokusů, stejně jako u cizích jazyků, většina stresu spojeného s maturitami by vymizela. Toto bohužel není něco, co bych mohl sám zařídit, ale doufám, že můj výtvor alespoň částečně vnese toto téma do povědomí lidí, kteří ho vyzkouší.

13. Zdroje

1. **Nikolaj Ivanovič Lobačevskij. *Wikipedia*. [Online] 2024. https://cs.wikipedia.org/wiki/Nikolaj_Ivanovi%C4%8D_Loba%C4%8Devskij.**

2. János Bolyai. *Wikipedia*. [Online] 2024. https://cs.wikipedia.org/wiki/J%C3%A1nos_Bolyai.

3. Non-Euclidean geometry. *Wikipedia*. [Online] 2024. https://en.wikipedia.org/wiki/Non-Euclidean_geometry.

4. Aperture Science Handheld Portal Device. *The Half-Life & Portal Encyclopedia*. [Online] Fandom, 2024. https://half-life.fandom.com/wiki/Aperture_Science_Handheld_Portal_Device.

5. Godot. *Wikipedia*. [Online] 2024. https://cs.wikipedia.org/wiki/Godot.

6. Visual Studio Code. *Wikipedia*. [Online] 2024. https://cs.wikipedia.org/wiki/Visual_Studio_Code.

7. Blender. *Wikipedia*. [Online] 2024. https://cs.wikipedia.org/wiki/Blender.

8. Paint.NET. *Wikipedia*. [Online] 2024. https://cs.wikipedia.org/wiki/Paint.NET.

9. Sketchfab. [Online] 2024. https://sketchfab.com.

10. Poly Haven. [Online] 2024. https://polyhaven.com.

11. Portal. *Wikipedia*. [Online] 2024. https://en.wikipedia.org/wiki/Portal_(video_game).

12. Antichamber. *Wikipedia*. [Online] 2024. https://en.wikipedia.org/wiki/Antichamber.

13. Superliminal. *Wikipedia*. [Online] 2024. https://en.wikipedia.org/wiki/Superliminal.

14. Steam. [Online] 2024. https://steampowered.com.

15. YouTube. [Online] 2024. https://youtube.com.

16. Creative Commons. [Online] 2024. https://creativecommons.org/.

14. Seznam obrázků