# Užité nástroje

## Engine

Veřejnosti je přístupno nepřeberné množství herních enginů, každý z nich se svými plusy a mínusy, orientovaný na jiný segment trhu. Například Unigine exceluje ve velkých scénách díky 64bitovým souřadnicím, zatímco Clausewitz je zaměřený na top-down *grand-strategy* hry.

Jelikož Astra je mým prvním velkým projektem, co se vývinu her týče, bylo pro mne důležité, aby pro zvolený engine byla dostupná rozsáhlá a kvalitní dokumentace a aktivní komunita, na kterou by se bylo možná obrátit v případě potíží. Tím z výběru vypadávají všechny málo známé enginy a také ty, které sice známé jsou, ale většinou na nich vyvíjejí pouze velká studia, jako například CryEngine.

Po tomto prvním kroku zbývali dva seriózní kandidáti, a to Unity a Unreal Engine. Finální volba nakonec padla na novější Unreal Engine 5 (nadále v této práci zkracován *UE*). Ten se jevil jako lepší volba díky jeho systémům jako je například engine osvětlení Lumen, který by mi značně usnadnil tvorbu realisticky vypadající grafiky.

## Další užité nástroje

UE má ve své základní distribuci zabudováno rozsáhlé množství nástrojů pro tvorbu a nakládání s různými soubory. I přesto bylo v průběhu vývoje nutné využít několik dalších programů:

### GIMP

GNU Image Manipulation Program (GIMP) je bezplatný rastrový grafický editor, který slouží k úpravě a tvorbě obrázků. Tento program jsem využil na tvorbu některých textur a design prvků uživatelského rozhraní.

Oproti konkurenčním programům, např. Adobe Photoshop, jsem GIMP zvolil zejména kvůli předchozí znalosti tohoto programu.

### Audacity

Audacity je open-source program pro úpravu zvukových souborů. Vzhledem k možnostem UE základně upravovat zvukové stopy (hlasitost, výška, filtry) přímo v enginu jsem Audacity užil jen zřídka a to na úpravu hlasových stop pro postavu hráčovy AI pomocnice a pro přípravu dalších zvukových stop převodem na podporovaný kodek.

### Git

### Git je distribuovaný systém správy verzí, který se používá pro sledování změn v kódu a koordinaci práce více lidí na společném projektu. Je to nástroj, který umožňuje uživatelům ukládat, verzovat a spravovat svůj kód a jeho historii.

### Git umožňuje uživatelům pracovat na kopii kódu, kterou si mohou upravovat bez toho, aby se to projevilo na hlavní větvi (tzv. "master branch"). Poté, co provedou své změny a jsou s nimi spokojeni, mohou je zahrnout do hlavní větve. Git také umožňuje ukládat různé verze kódu v různých větvích, což usnadňuje vývoj a testování nových funkcí nebo náprav chyb. Díky němu je také snadnější zpětné získání předchozích verzí kódu a práce na nich.

Jelikož jsem na projektu pracoval sám, řady z vlastností Gitu jsem vůbec nevyužil. Stejně tak zůstala nevyužitá i možnost vývoje ve více větvích (z důvodu, že jsem dříve nebyl se systémy sledování verzí), což není optimální.

### Blender

Blender je otevřený a zdarma dostupný software pro 3D modelování, animaci a vizualizaci, široce používaný v oblasti filmového a herního průmyslu, architektury, designu a mnoha dalších. Uživatelům umožňuje vytvářet 3D objekty, animovat je, texturovat, osvětlovat a renderovat. Další funkcí Blenderu jsou například simulace fyziky, jako jsou srážky a deformace, simulace tekutin a plamenů apod.

Uživatelé také mohou vytvářet složité animace pomocí animačních klíčů, sledování kamer a světel, a dalších funkcí. Právě tuto konkrétní funkci jsem při vývoji použil, a to na vytvoření krátkého videa s logem hry, které se přehrává, když je hra spuštěna.

# Průběh vývoje

## Počáteční cíle

Celkový koncept hry se během vývoje několikrát změnil. Původní plánování jsem ale provedl s následujícími cíli:

* Střílečka z pohledu třetí osoby (*Third-person shooter*), zasazený ve vesmírném prostředí.
* Pohybový model hráčovy vesmírné loďe, který umožňuje pohyb po všech šesti osách (tj. 3 osy trojrozměrného prostoru + 3 osy otáčení)
* Absence konečného úkolu – hra skončí pouze selháním či přáním hráče
* Počítačoví protivníci, kteří se dokáží v prostoru pohybovat stejným způsobem jako hráč a jsou schopni na něj efektivně útočit
* Realisticky vypadající (nestylizovaná) grafika

Za inspiraci měli sloužit podobné hry, jako například nedávno vydané *Star Wars: Squadrons* a starší *Star Wars: Battlefront II* od Electronic Arts nebo *Elite: Dangerous* od Frontier Developments.

## Vývoj

UE uživatelům nabízí několik předem zhotovených šablon, které obsahují všechny základní nezbytné prvky a dovolují uživatelům pouze rozšiřovat z tohoto funkčního základu. Této možnosti jsem nevyužil, jelikož žádná z nabízených šablon nevyhovovala výše uvedeným požadavkům. Zejména problematická by byla implementace volného pohybu v trojrozměrném prostoru – ve všech šablonách se počítalo s přítomností země a gravitace.

### 3D

Prvním úkolem pak tedy bylo vytvořit prázdnou úroveň a naprogramovat základní funkce. Původní letecký model z této prvotní fáze byl ovládaný kombinací myši a klávesnice. Klávesnice zajišťovala rotaci okolo vektoru pohybu, zatímco myš zajišťovala stáčení. Loď cestovala neměnnou rychlostí (pokud nedošlo ke kolizi). V úrovni se také vyskytoval jeden „nepřítel“, který sloužil jako test systému střelby. Nepřítel samotný se nijak nepohyboval ani neútočil. Jediná jeho implementovaná funkce bylo jeho zničení, kdy po určitém počtu zásahů zmizel.

Další verze zaznamenaly několik zlepšení k funkcím hráčovy lodě. Nově hráč mohl změnit rychlost letu v rámci povoleného rozsahu. Zároveň se objevila chyba v zobrazování uživatelského rozhraní, která zůstala dlouhou dobu nespravena. Naopak několik chyb bylo odstraněno, a to převážně v konzistentnosti systému střelby. V neposlední řadě došlo ke změnění osvětlení úrovně a přidání několika překážek v podobě asteroidů.

### Přechod na 2D

Problém s původní 3D koncepcí se objevil při práci na počítačových protivnících. Bohužel, pathfinding v trojrozměrném prostoru se ukázal jako extrémně obtížný na implementaci, alespoň pro začátečníka. Došel jsem k názoru, že jelikož není možné hru v tehdejší koncepci dokončit, a je tedy nutné změnit cíle. Nesmírnou nevýhodou bohužel bylo, že zhruba polovina dosavadní práce tak musela být zahozena. Zůstat mohlo jenom pozadí (tzv. skybox) a modely lodí. Veškerá logika a kód ale musely být kompletně přepsány.

Jako první bylo nutné určit, kterým žánrem 2D her bude Astra inspirována. Z důvodu osobní preference tohoto žánru jsem nakonec zvolil tzv. „top-down shoot-em-up“, konkrétně podžánr „bullet hell“.

Tyto hry mají několik identifikujících znaků. Rozeberme si proto jednotlivé pojmy:

„Top-down“, doslovně „zezhora dolů“, značí hru, ve které je kamera fixně umístěna nad herním polem, a ve které se hráč pohybuje po obrazovce (typicky po osách X a Y, ne však do hloubky) nezávisle.

„Shoot-em-up“ je takový typ hry, která obsahuje velké množství v poměru s hráčem velmi slabých protivníků. Z toho vyplývá, že hráč za určitý čas porazí větší množství protivníků, než je tomu u jiných žánrů. Shoot-em-up hry jsou proto často vnímány jako adrenalinové, a byly velmi populární v raných letech videoher na arkádových automatech. Jednou z prvních takových her byl např. Galaxian od společnosti Bandai-Namco z roku 1979, nebo proslulí *Space Invaders* z předcházejícího roku, kteří jsou často považováni za zakladatele žánru, mimo jiné[[1]](#footnote-1).

Bullet hell je podžánr shoot-em-up her, který se vyznačuje tím, že je obrazovka „zaplavena“ nepřátelskými projektily, které sice nepůsobí valné poškození, ale jejich nebezpečí se skýtá v jejich množství. Hráč tak musí obratně manévrovat, aby se vyhnul co největšímu množství střel. Velmi častou vlastností je menší kolizní rámec projektilů, než jaká je jejich velikost na obrazovce. To umožňuje hráči provádět smělejší manévry a pomáhá balancovat obtížnost ve prospěch hráče.

Při tvorbě jsem se samozřejmě inspiroval i současnými tituly, mezi nimiž byla nejdůležitější NieR: Automata od Platinum Games.

## Algoritmus AI počítačových protivníků

Systém umělé intelegence protivníků je vlastně rozdělen do dvou, na sobě minimálně závislých systémů. Jeden se stará o nepřátele jako celek. Druhý pak ovládá jednotlivce. Nejdříve se budu zabývat prvním zmíněným.

### MasterSpawner

Tento Actor se stará převážně o to, aby počet nepřátel na poli nepřesáhl určitou hranici (která je stanovena úrovní obtížnosti). Je vyobrazen na obrázku v příloze číslo 1.

Nejprve ale krátké shrnutí základů vizuálního programování Blueprint. Program je rozdělen na jednotlivé nodes, z nichž některé obsahují příkazy, jiné operace či proměnné. Nodes s příkazy s bílým vstupem a výstupem *exec* se vyhodnocují zleva doprava, podle toho jak jsou propojeny. Ostatní nodes bez vstupu *exec* se vyhodnocují v ten okamžik, kdy je vyhodnocen node, do kterého vede jejich výstup.

Na začátku hry tento blueprint nejdříve nastaví některé důležité proměnné, a následně na každém z Actorů *EnemySpawner* zavolá funkci *SpawnEnemy*.

EnemyCount = 0

Spawners = [...]

#contains object references to all actors of class 'EnemySpawner'

for item in Spawners:

item.SpawnEnemy(CombatStyle="ShootThreeRoundBurst")

EnemyCount += 1

Na konci této setup fáze aktivuje node *Gate* v hlavním těle tohoto blueprintu.

Node Gate umožňuje projít signálu pouze v takovém případě, že se nachází ve stavu *Open*.

Pokud se toto stane, následují dvě věci. Zaprvé, pokud je současný počet nepřátel nižší než maximální dovolený, zavolá funkci spawn v náhodném spawneru ve hře, s náhodnou hodnotou argumentu *CombatStyle*.

Za druhé přičte 1 k počtu nepřátel ve hře, a pokud je nový počet roven maximálnímu počtu, zavře *Gate*.

Zavřený Gate se otevře v takovém případě, že dojde ke smrti některého z nepřátel.

Tento princip je shrnutý v následujícím pseudokódu:

EnemyCount = 0

MaximumEnemyCount = 0

GateIsOpen = True

Spawners = [...]

CombatStyles = [

ShootContinuous,

ShootThreeRoundBurst,

ShootSeeking,

]

def main():

if GateIsOpen:

if EnemyCount < MaximumEnemyCount:

spawner = Spawners[random]

spawner.SpawnEnemy(CombatStyles[random])

EnemyCount += 1

if EnemyCount >= MaximumEnemyCount:

GateIsOpen = False

else:

GateIsOpen = False

def EnemyDied():

# called from outside the blueprint,

# by the enemy that has died

EnemyCount -= 1

wait(randint(2,5))

GateIsOpen = True

while True: #called every tick, ie. every new frame

main()

### AI Controller

V okamžiku, kdy MasterSpawner vytvoří a do úrovně umístí nepřítele, přebírá kontrolu nad ním tzv. *Behaviour Tree*, tedy doslovně „strom chování“. Jedná se o mocný systém, který dovuleje vytvořit velmi komplexní vzorce chování, spolupracovat s ostatními jeho instacemi a další. Vzhledem k povaze této hry jsem však těchto možností nevyužil.

Nepřátelé ve hře musí být schopni dvou věcí – pohybu a útoku. Behaviour tree se z těchto dvou stará o pohyb, a to v celku primitivním způsobem.

1. Vybere náhodně lokaci v rámci předurčeného herního pole
2. Posune se do ní
3. Čeká náhodnou dobu
4. Opakuje od bodu 1

Druhým systémem je střelba, která je již o něco složitější.

Existují dva typy nepřátelských lodí. Jedny útočí v krátkých dávkách, jedny neustále. Podle při stvoření určeného typu se tak zavolá jedna ze dvou nekonečných smyček.

# Příručka

## Systémové požadavky

* Volné úložiště: ~500 MB
* Grafická karta podporující DirectX 11 či vyšší
* Windows 7 64-bit či novější

## Instalace

Spusťte soubor JMÉNO INSTALLERU ZDE a následujte pokyny na obrazovce.

## Prvotní nastavení

V hlavním menu hry zvolte nastavení. Posléze nastavte vámi preferované rozlišení a grafické předvolby.

Pokud si nejste jistí, jaké nastavení zvolit, je doporučeno rozlišení shodné s rozlišením vašeho monitoru. Nedojde tak ke ztrátě kvality z důvodu škálování. Pokud při hře nemáte problémy s nízkými FPS, doporučuje se pro nejlepší zážitek zvolit grafické nastavení „Ultra“.

Na této obrazovce je také možnost nastavit obtížnost hry, tou se budeme zabývat v dalších sekcích.

## Ovládání

Hra podporuje dvě vstupní zařízení – klávesnice/myš a ovladač. V této době není možné ovládání uživatelsky upravit.

Pro hru na ovladači je stále vyžadována myš pro interakci s uživatelským rozhraním.

### Klávesnice/myš

Pohyb: WASD, šipky

Střelba: Levé tlačítko myši, mezerník

### Ovladač

Pohyb: Levý stick

Střelba: B, Pravý trigger

## Herní rozhraní

Herní rozhraní se sestává ze dvou prvků. V levém horním rohu obrazovky se nachází počitadlo bodů. V levém dolním rohu se pak nachází ukazatel zdraví hráče.

Stiskem klávesy ESC může hráč hru kdykoli pozastavit. Z menu pauzy může ve hře pokračovat, restartovat úroveň, či hru vypnout. V tomto případě nebude jeho skóre uloženo do žebříčku.

## Cíl hry

Cílem hry je nasbírat co nejvyšší skóre.

Skóre hráč získává za každé zasažení nepřítele. V zájmu zajištění svižného průběhu hry hráč o malý počet bodů každou vteřinu přichází.

Svoje současné skóre může hráč sledovat v levém horním rohu obrazovky. Na konci se zobrazí jeho konečné skóre a žebříček pěti nejvyšších dosažených skóre (na tomto zařízení).

## Průběh hry

Hráč se může v průběhu hry setkat s dvěma typy nepřátelských lodí. Jedny střílí fialovo-modré projektily v dávkách, zatímco druhé, ty nebezpečnější, střílí konstantně dokud nejsou zničeny. Obecně platí, že poměr těchto typů je 2:1 ve výše uvedeném pořadí.

Projektily cestují vždy v přímce a hráč je může zásahem zničit. Pokud je hráč zasažen, ztrácí určitý počet bodů života (HP). Pokud HP hráči dojdou, je jeho loď zničena a hra končí.

Na herním poli se mohou objevit také větší zelené projektily. Ty hráči neškodí, naopak mu obnoví několik HP.

# Delet

### Build 0.0.4

Tato verze byla prozatím poslední, ve které jsem se plánoval zabývat lodí hráče. Byla zvýšena rychlost zatáčení lodě i její akcelerace. Debugové přímky při střelbě ze zbraní byly nahrazeny hezčími lasery, které mimo jiné vydávají slabé světlo. Byl také zpraven bug, který nedovolil zobrazení HUDu ve zkompilované hře. Hra také v této verzi dostala současné jméno Astra.

Hráče jsem tímto považoval za hotového a další položkou na seznamou byla implementace nepřátel.

1. Této hře je také přisuzován koncept více životů a udržování žebříčku nejvyšších skóre [↑](#footnote-ref-1)