Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Nejvhodnější herní engine pro začátečníky

Lukáš Vacula

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Lukáš Vacula

Systémové inženýrství a informatika
Informatika

Název práce

Nejvhodnější herní engine pro začátečníky

Název anglicky

The most suitable gaming engine for beginners

Cíle práce

Hlavním cílem práce je porovnání vybraných herních enginů z hlediska vhodnosti pro začínající vývojáře. Pro dosažení tohoto cíle bude v každém enginu vytvořen shodný prototyp aplikace, jehož implementace bude následně porovnávána s ostatními prototypy. Výsledek komparace bude podporovat rozhodovací proces při výběru vhodného herního enginu nejen pro začátečníky v této oblasti.

Metodika

Metodika zpracování teoretické části práce vychází ze studia odborných informačních zdrojů. Na základě syntézy zjištěných informací budou stanovena východiska pro realizaci praktické části bakalářské práce.

Praktická část bude zaměřena na herní enginy Unity, Unreal Engine, CryEngine a Godot za použití programovacích jazyků C#, C++ a GDScript. Nejprve budou stanoveny vlastnosti a funkce požadovaných prototypů, které budou následně implementovány v každém herním enginu.

Na základě poznatků získaných z literární rešerše budou herní enginy zhodnoceny a podle prototypů v nich vytvořených dále porovnány.

Doporučený rozsah práce

35-40 stran

Klíčová slova

rozhodovací proces, engine, vývoj SW, 3D, UX

Doporučené zdroje informací

BROŽOVÁ, Helena, Milan HOUŠKA a Tomáš ŠUBRT, 2003. Modely pro vícekriteriální rozhodování. Praha: Credit. ISBN 978-80-213-1019-3.

BUCHALCEVOVÁ, Alena, 2005. Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky. Praha: Grada. Management v informační společnosti. ISBN 80-247-1075-7.

FIALA, Petr a Miroslav MAŇAS, 1994. Vícekriteriální rozhodování: Určeno pro stud. všech fak. Praha: Vysoká škola ekonomická. ISBN 80-7079-748-7.

MERCAN, Sahin a Pinar Onay DURDU, 2017. Evaluating the Usability of Unity Game Engine from Developers' Perspective. 2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT) [online]. IEEE, 2017, 1-5. ISBN 978-1-5386-0501-1. Dostupné z: doi:10.1109/ICAICT.2017.8687303.

Předběžný termín obhajoby

2020/21 LS – PEF

Vedoucí práce

Ing. Tomáš Benda

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 10. 03. 2021

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Nejvhodnější herní engine pro začátečníky" jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3. 2021 _____

Lukáš Vacula

Poděkování

Rád bych touto cestou poděkoval vedoucímu bakalářské práce Ing. Tomáši Bendovi za jeho profesionální přístup, cenné rady a časté osobní konzultace.
Dále bych chtěl poděkovat panu Petru Hadrabovi za provedení kontroly matematických výpočtů.

Nejvhodnější herní engine pro začátečníky

Abstrakt

Bakalářská práce pojednává o problematice vývoje her ve vybraných game enginech Unity, Unreal Engine, Godot a CryEngine. Nejprve je v teoretické části práce čtenář seznámen s jednotlivými vývojovými rozhraními. Cílem práce je porovnání enginů vícekriteriální analýzou variant a z výsledků následně čtenáři doporučit nejvýhodnější variantu. V prostředí každého z nich je vytvořen prototyp závodní hry. Za účelem doporučení nejvhodnějšího enginu pro začínající vývojáře je postup vývoje v jednotlivých programech dokumentován a podle dotazníku pro testování použitelnosti systému ohodnocen (Systém Usability Scale, SUS). Dalšími hledisky hodnocení enginů jsou velikosti příslušných komunit a uživatelská přívětivost výchozích skriptovacích jazyků. Ohodnocena je dále implementace souborů z jiných aplikací a rozsah funkcí enginu. Určení nejvhodnějšího game enginu je provedeno matematickou metodou vícekriteriálního rozhodování, jejíž problematice se věnuje teoretická část práce a na základě poznatků jsou v praktické části Saatyho metodou provedeny výpočty vah kritérií. Následně je metodou váženého součtu, který se vztahuje na výpočty vah, proveden výběr nejvhodnější varianty vývojového rozhraní.

Z provedené analýzy vyplívá jako nejvhodnější varianta pro začínajícího vývojáře game engine Unity, kterého v pořadí následují programy Godot, Unreal Engine a CryEngine. Vývojové prostředí Unity se prokázalo jako jednoznačná volba pro začínající vývojáře především díky velké komunitě, efektivnímu skriptovacímu jazyku a uživatelsky přívětivému návrhu UI. Pro realizaci bakalářské práce jsou čtyři hratelné prototypy vytvořené v každém vývojovém prostředí.

Klíčová slova: rozhodovací proces, engine, vývoj SW, 3D, UX, Unity, Unreal Engine, Godot, CryEngine

The most suitable game engine for beginners

Abstract

The bachelor thesis deals with the issue of game development in selected game engines Unity, Unreal Engine, Godot and CryEngine. First, in the theoretical part of the work, the reader is acquainted with the individual development interfaces. The aim of the work is to compare engines by multicriteria analysis of variants and subsequently to recommend the most advantageous variant to the reader from the results. A prototype of a racing game is created in the environment of each of them. In order to recommend the most suitable engine for beginning developers, the development process in individual programs is documented and evaluated according to the questionnaire for testing the usability of the system (System Usability Scale, SUS). Other aspects of engine evaluation are the size of the respective communities and the user-friendliness of the default scripting languages. The implementation of files from other applications and the range of engine functions are also evaluated. The determination of the most suitable game engine is performed by the mathematical method of multicriteria decision-making, the issue of which is addressed in the theoretical part of the work and based on the knowledge, the practical part of the Saaty method calculates the weights of criteria. Subsequently, the most suitable variant of the development interface is selected using the weighted sum method.

The analysis shows that the most suitable variant for the beginning developer of the game engine is Unity, followed by Godot, Unreal Engine and CryEngine. Unity's development environment has proven to be a clear choice for beginning developers, mainly due to its large community, efficient scripting language and user-friendly UI design. For the implementation of the bachelor's thesis, there are four playable prototypes created in each development environment.

Keywords: decision process, engine, SW development, 3D, UX, Unity, Unreal Engine, Godot, CryEngine

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Každým rokem přibývá větší počet tzv. indie game developerů, což jsou nezávislí herní vývojáři, kteří nemají smlouvu se žádnou společností a vytváří hry buď ve skupině, nebo sami. Jedná se o nadšence herního světa, ve kterém se chtějí podílet vlastním dílem. Mnohdy se stane, že jsou to právě tyto produkty, které vzbudí na herní scéně ten největší rozruch. Hry jako The Stanley Parable, Limbo, Dead Cells, Super Meat Boy nebo např. Minecraft jsou jedny z mnoha velmi úspěšných titulů, které byly vytvořeny v malých kolektivech a docílily nesmírné popularity. Hlavními důvody pro volbu tématu bakalářské práce bylo autorovo přání lépe se orientovat a získat praxi v herních enginech. Na internetu koluje mnoho článků, ve kterých se popisují výhody jednotlivých programů, ale žádné se nevěnují vývoji totožného prototypu v každém z nich a následném porovnání z hlediska náročnosti implementace. Tato práce je určena pro začínající herní vývojáře, kterým poskytuje ohodnocení jednotlivých programů a jejich specifikace. Jejím cílem je stát se příručkou pro čtenáře shánějící informace o vlastnostech jednotlivých enginů a zkušenostech začínajícího vývojáře. Dostupných herních enginů je v současnosti nepřeberné množství a je jednoduché se ve všech jejich výhodách a záporech všech programů ztratit. Pokud bychom se chystali investovat velké množství času do práce v jednom enginu, bylo by vhodné se nejprve seznámit s jeho prostředím, rozsahem jeho funkcí a zda v něm budeme schopni vytvořit náš vysněný projekt.

V práci jsou porovnávány programy Unity, Unreal Engine, Godot a CryEngine. Tyto specifické software byly vybrány z toho důvodu, že jsou všechny volně dostupné ke stažení, lze v nich vytvářet 3D aplikace a jedná se o jedny z nejpopulárnějších zástupců ve vývoji her, tomu především odpovídají jejich příslušné tituly.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem práce je porovnání vybraných herních enginů z hlediska vhodnosti pro začínající vývojáře. Pro dosažení tohoto cíle bude v každém enginu vytvořen shodný prototyp aplikace, jehož implementace bude následně porovnávána s ostatními prototypy. Výsledek komparace bude podporovat rozhodovací proces při výběru vhodného herního enginu nejen pro začátečníky v této oblasti.

2.2 Metodika

Metodika zpracování teoretické části práce vychází ze studia odborných informačních zdrojů. Na základě syntézy zjištěných informací jsou stanovena východiska pro realizace praktické části bakalářské práce. Praktická část se zaměřuje na vývoj prototypů v těchto game enginech:

- Unity
- Unreal Engine
- Godot
- CryEngine

Při vývoji prototypů jsou použity programovací jazyky C#, C++ a GDScript. Kritéria hodnocení programů jsou následující:

- Práce v enginech
- Komunita
- Výchozí skriptovací jazyk
- Implementace souborů z jiných programů
- Rozsah funkcí

Výsledné hodnocení je čtenáři prezentováno formou vícekriteriální analýzy variant. Důvody, proč jsou jednotlivé varianty programů odlišně bodově ohodnoceny, se udávají

v jejich příslušných postupech vývoje prototypu enginů. Kritéria „práce v enginech“ a „výchozí skriptovací jazyk“ jsou číselně zastoupeny pomocí dotazníku pro testování použitelnosti Systém Usability Scale neboli Škála použitelnosti systému.

3 Teoretická východiska

V teoretické části práce se autor zabývá seznámením čtenáře s problematikou game enginů a jednotlivých zástupců, vývojářských týmů, programovacích jazyků použitých při vývoji prototypů, definicí AI, uživatelské zkušenosti a vícekriteriálního rozhodování.

3.1 Vývoj software

Vytváření programu je ovlivněno vývojovým a cílovým prostředím. Při vývoji software nás zajímají tyto veličiny: zda jsou dostupní kvalifikovaní specialisté, jaká je stabilita technologie, stabilita a schopnosti strojů, do jaké míry jsou efektivní použité metody, postup a jeho flexibilita, časová stránka, konkurující software, množství zdrojů atd. (Buchalcevová, 2005)

„Celková složitost vývoje software je funkcí těchto proměnných, přičemž tyto proměnné se v průběhu projektu mění:

složitost = f(proměnné prostředí vývoje + proměnné cílového prostředí)

Roste-li složitost projektu, je třeba zařazovat do procesu více kontrolních prvků (například řídit rizika apod.).“ (Buchalcevová, 2005)

3.1.1 Game engine

Také přezdívaný game framework, je software poskytující vývojové prostředí, specifikované především pro tvorbu her. Většina enginů podporuje tvorbu hry i pro ostatní zařízení, jako jsou např. mobilní telefony a konzole. Vývojáři obvykle pro nové tituly využívají stávající engine jako základ a rozvíjejí jej o nové funkce, je-li to zapotřebí.

Architektura a implementace jednotlivých herních enginů je značně rozdílná, přesto lze nalézt podobné vzory v interních a ve veřejně licencovaných enginech. Prakticky všechny herní enginy obsahují známou sadu základních komponent, zahrnujících renderovací engine, animační systém, kolizní a fyzický engine, audio systém, objektový model herního světa, systém umělé inteligence atp. (Gregory, 2018)

3.1.1.1 Unity

Příkladem univerzálního game enginu je Unity. Představen na Apple konferenci 2005, Unity je multiplatformní game engine, který lze využít pro vývoj 2D, 3D VR a AR her. Mezi jeho výhody patří jednoduchost user interface (UI) se strukturou komponent, scéna a prefab. Nový uživatel se tak dokáže rychle orientovat a osvojit si jednotlivé funkce. Programovací neboli skriptovací jazyk aplikace je C#, který je především známý svojí úhlednou a skromnou syntaxí. Obvyklé programovací problémy v jiných jazycích jsou tak vyřešeny na pozadí. S novými technologiemi virtual reality (VR) a augmented reality (AR) lze využívat Unity kromě her i pro trénovací simulace, aplikace, vizualizace architektury, filmy apod. Značně rozsáhlá komunita vývojářů, velké množství návodů a tutoriálů usnadňuje novým uživatelům vyhledat řešení pro jejich specifický problém. Zásluhou za svůj rychlý rozvoj je zastoupení pracovních pozic u Unity na trhu větší než u kteréhokoliv jiného game enginu. (Alejandro Borromeo, 2020)

3.1.1.2 Godot

Godot byl v roce 2014 publikován jako open-source, multiplatformní game engine se zaměřením na vývoj 2D a 3D her, s jehož vývojem začali Juan Linietsky a Ariel Mazur pro společnosti v Jižní Americe. (Godot 2.0: Talking with the Creator, 2016)

Godot podobně jako Unity nabízí jednoduchý UI ovšem se stromovou strukturou scén a uzlů, což umožňuje uživateli vnořovat a/nebo dědit jiné scény. (Linietsky, 2020)

Uzly jsou základními stavebními prvky scény, které zastupují různorodé funkce od 2D do 3D. Jednotlivé uzly mohou například představovat grafické rozhraní scény, animaci nebo objekt jako je krychle, kužel, sféra atp. Dále jsou v nich zahrnuty vlastnosti, podle nichž lze konfigurovat reakce na ostatní tělesa, velikost a chování objektu. Scény jsou v programu využívány pro tvorbu různých těles a objektů ve hře. Jedním takovým příkladem je scéna AI, ve které uživatel upravuje vlastnosti pomocí uzlů a skriptů. Další scénou může být například úroveň, ve které se hráč spolu s AI bude nacházet. Tato scéna obsahuje pouze hrací prostor, ovšem po propojení se scénami hráče a AI se z ní stává již hratelný prototyp. (Bradfield, 2018)

3.1.1.3 Unreal Engine

V roce 1998 publikovala společnost Epic Games hru Unreal, která byla prvním produktem vytvořeným v Unreal Enginu. První verze se skládala z kombinace systémů pro renderování

(vykreslování), detekci kolizí, AI, osvětlení, networking, skriptování a správu souborů. I přesto, že se engine vyvinul pro tvorbu first person shooter (FPS) her, jej vývojáři dokázali využít i pro ostatní žánry jako např. MMORPGS (Massively multiplayer online role playing games), bojové hry, jump 'n' run atp.

Unreal Engine se stal favoritem vývojářů především díky svému modulárnímu návrhu architektury a skriptovacímu jazyku Unreal Script. Tento jazyk byl založen na syntaxi jazyka C++ a umožňoval uživatelům snadno vytvářet modifikace.

Nejnovější verze Unreal Engine 4 byla vydána v roce 2014 a přinesla řadu zásadních změn. Byl zaveden uživatelsky přívětivý Blueprints system, který nahradil stávající vizuální skriptování Kismet. Provedly se také úpravy osvětlení pomocí zavedení nového algoritmu, který snižuje zátěž programu způsobený výpočetními kalkulacemi. Editor prošel generální úpravou, aby byla snížena celková doba sestavení projektu pro zvýšení rychlosti, kterou engine provádí vyhotovení iterací ve hře.

Pokud chce uživatel používat pro tvorbu funkcí skriptovací jazyk C++, musí si na zařízení nainstalovat například Vývojové prostředí (IDE, Integrated development environment) Visual Studio. (Sanders, 2016)

V současnosti je Unreal Engine především známý kvalitou grafického zpracování. Mnoho společností, které chtějí vytvářet tzv. AAA tituly (termín pro hry s vysokým kapitálem) volí právě Unreal Engine pro tuto vlastnost.

Mezi nejznámější tituly patří např. Borderlands 2, BioShock, Final Fantasy VII, Fortnite nebo Gears of War.

3.1.1.4 CryEngine

CryEngine byl v roce 2002 publikován německou firmou Crytek, se zaměřením na vývoj 3D herních technologií. Mezi jeho hlavní výhody patří zvýšení kvality obrazu ve všech populárních žánrech, jako jsou např. Action, Role Playing Games (RPG) a Strategy. Jako první game engine byl schopen poskytnout plynulý přesun z venkovních do vnitřních scén.

Byl revolučním enginem, který stanovil nové standardy v oblasti grafiky a kvality zpracování her. Jeho hlavní části tvoří 3D Renderování, integrovaný skriptovací systém, networking a pokročilý fyzický a audio engine. (Yerli, 2002)

Mezi jeho nejznámější tituly patří například Sniper: Ghost Warrior 2, Kingdom Come: Deliverance publikovaná českou firmou Warhorse Studios s.r.o., SNOW a modifikovanou verzí software také série Far Cry.

3.1.2 Skriptovací jazyk

Bez skriptovacích neboli programovacích jazyků bychom nebyli schopni dodat hře základní prvky hratelnosti a náš prototyp by tak byl pouhým prostředím s postavami. Pomocí skriptů (malé textové dokumenty s kódem) rozvíjíme naši hru o kategorie, jako jsou např. pohyb, AI, časovač, systém dne a noci atp. Ve skriptech vytváříme systémy a funkce, které našim postavám dodávají originalitu a probouzí je k životu.

Skriptovacích jazyků je v současnosti velké množství a mnoho herních enginů nabízí výběr a možnost programování v některých z nich.

Níže jsou v podkapitolách popsány jednotlivé programovací jazyky, které autor využil při tvorbě prototypů.

V testovaných prototypech jsou skripty využity pro pohyb hráče, implementaci soupeřů (AI), resetování a podmínky pro (ne)úspěšné dokončení hry.

3.1.2.1 C++

Autorem programovacího jazyka je Bjarne Stroustrup, který jazyk navrhl jako univerzální, multiparadigmatický a objektově orientovaný. Obsahuje ovšem podporu i pro generické, procedurální programování a datovou abstrakci.

- Abstrakcí dat se programovací jazyk zabývá při celkové reprezentaci, návrhu rozhraní a skrytím podrobností zpracování před uživatelem. Lze v jazyce využívat jak konkrétní, tak abstraktní třídy. Pro definování tříd s konstruktory, destruktory, přidruženými metodami a privátní architekturou.
- Objektově orientovaný jazyk pracuje s hierarchickou strukturou tříd, především s jejím provedením a návrhem. Při spuštěném programu poskytují hierarchie tříd polymorfismus (volání objektu metody s totožným názvem ale rozdílnou implementací) a zapouzdření.

- S obecnými algoritmy je spojen výraz generické programování. Obecným algoritmem se rozumí takový postup, který lze použít při mnoha příležitostech, pokud jsou splněny jeho požadavky v argumentech.
- Procedurální programování se vztahuje na zpracování a návrh stabilních datových struktur. Jedná se o programovací styl, který podporoval i jazyk C, Fortran atd. Přichází ve formě zabudovaných konstruktorů, výroků, operátorů a unií.

V C++ je kladen důraz na využití typově bohatých a nezatěžujících abstrakcí. Vyniká při použití v aplikacích a softwarových strukturách, které jsou omezeny na zdroje.

Základními prvky, na které se C++ soustředí jsou paměť, zpracování chyb, abstrakce, proměnlivost, modularita a správa zdrojů. Toto jsou složky, kterými se zabývají systémový programátoři a obecněji programátoři, kteří mají na starost systémy specifikované na vysoký výkon. Definováním knihoven tříd a jejich hierarchií a šablon může každý uživatel psát k´d na vysoké úrovni. Užití jazyka C++ je v praxi vskutku široké, používá se hojně ve finančních systémech, pro vědecké výpočty a v herním odvětví. (Stroustrup, 2013)

3.1.2.2 C#

C# (vyslovovaný jako „See Sharp") je objektově orientovaný programovací jazyk s podporou pro komponentově orientované programování. Mezi jeho důležité prvky pro tvorbu vydatných a obstojných aplikací patří tzv. garbage collector, který automaticky uvolní paměť obsazenou neaktivními objekty, dále exception handling (výjimka) programům poskytuje strukturovaný způsob detekce a obnovení závad. Další důležitou složkou C# je návrh type-safe (typově bezpečný), který zakazuje programu číst z neinicializovaných proměnných, indexovat „arrays" neboli pole za jejich respektivními hranicemi a provádět tzv. unchecked casts (nekontrolované obsazení). (Hejlsberg, 2010)

C# jako vývoj jazyka C++ zjednodušil jeho mnoho funkcí, ale na výkonnosti nic neztratil. Vyřešil také notorické problémy C++. Zbavil se ukazatelů (pointers), které programátory připravovaly o čas a úsilí vynaložené pro řešení problému memory leak (únik paměti) s nimi spojeným. Odstranil také dědičnost více tříd, která způsobovala více závad než nabízela výhod. Verze z roku 2005 dokonce rozvinula jazyk o další silné vlastnosti jako např. obecné typy, které se staly přívětivější než C++ šablony, dílčí třídy atd. (H. Chaudhary, 2014)

Názorná syntaxe jazyka C# v game enginu Unity:

```
using UnityEngine;

public class PlayerMovement : MonoBehaviour {

   public Rigidbody = rb;

   public float forwardForce = 200f;

   void FixedUpdate (){

      rb.AddForce(0, 0, 2000 * Time.deltaTime); }

}
```

3.1.2.3 GDScript

Dynamický psaný programovací jazyk vysoké úrovně, GDScript je optimalizován a integrován v herním enginu Godot. Tento způsob zabudování skriptovacího jazyka mu umožňuje rozsáhlou flexibilitu při tvorbě a začlenění obsahu. (Jaiswal, 2018)

Syntaxe GDScript se velice podobá syntaxi jazyka Python. Uživatel tedy nepotřebuje deklarovat typy proměnných při jejich tvorbě, pro rozdělení kódu do částí se používá odsazení řádkem (C# a C++ používají rozdělení {} těmito závorkami), což pro uživatele znamená psaní kratších skriptů a rychlejšího vývoje.

Pomocí skriptovacího jazyka GDScript je Godot schopný live edit (dosl. živé úpravy). To znamená, že pokud chce uživatel otestovat integritu produktu, může prototyp upravovat během jeho provozu a tvořit tak úrovně a nové prvky za pochodu bez nutnosti ukončení aplikace. (Linietsky, 2020)

Názorná ukázka kódu v GDScript udává postavě, které je skript přidělen příkaz, aby se pohybovala vpřed a vzad po stisknutí kláves, které jsou s těmito pohyby spojené v nastavení.

```
extends KinematicBody

var gravity = 9.8
```

```
var direction = Vector3()

var acceleration = 5

var speed = 50

func _ready():

    pass

func _physics_process(delta):

    direction = Vector3()

    if Input.is_action_pressed("move_forward"):

        direction += transform.basis.x

    elif Input.is_action_pressed("move_backwards"):

            direction -= transform.basis.x
```

3.1.2.4 Vizuální skriptování

Vizuální skriptování (VS) je výhodná pomůcka pro nejen začínající vývojáře. Její návrh je konstruován takovým způsobem, aby snížila obtížnost klasického programování, ale zachovala si co nejvíce funkcionality. Kód je reprezentován vizuálně, a proto k jeho porozumění nepotřebujeme příliš abstraktního myšlení a můžeme se v systému rychle a snadno orientovat. Pokud se rozhodne skupina umělců pro vývoj hry, je tato varianta programování dostačující.

Argumentem, proč se klasické programování nestává zastaralým je ten, že se VS dost dobře nerozvíjí. Uživateli zabere mnohem více času vytvořit správně pracující kód a jeho modifikace je velice obtížná.

Nejčastěji je VS využíván začátečníky ve vývoji her, kteří se chtějí naučit práci s herními enginy, ale nemají žádnou praxi s programováním. Další skupinou jsou například umělci, kteří chtějí urychleně vytvořit prototyp hry. Někdy je VS použit i zkušenými programátory, kteří usilují o zpřístupnění herní logiky umělcům v týmu, aby si sami vyzkoušeli implementaci jejich umění do hry. (Linietsky, 2020)

Aktuální případ, kdy vývojářské týmy převážně kódují hry pomocí vizuálních skriptů, je hra Hollow Knight od studia Team Cherry. Kromě několika výjimek, které byly napsány klasickým kódem ve skriptech, jsou veškeré funkce protivníků a interaktivní elementy zpracovány programem Playmaker pro VS, který je jedním z mnoha tzv. assetů herního enginu Unity.

3.1.3 Artificial Intelligence

Zkráceně AI, je v určité formě zakomponován do každé herní struktury. Od nepřátel v klasické arkádové hře Pac Man až po tzv. Bots (Robotic Players) ve hře Unreal. Se vznikem nových herních žánrů a vývojem technologií je pojem AI každým rokem rozšiřován. Podle některých vývojářů je samotná detekce kolizí objektů formu AI, ostatní za její součást považují vyhledání cesty (trasy).

Během vývoje hry naším cílem většinou nebývá implementace inteligence podobné lidské, ale snažíme se o přiřadit charakterům, které hráč neovládá, různorodé funkce, aby hráč neupadal do rutiny a byl co nejvíce v pozoru. (M. Bourg, 2004)

3.1.3.1 Deterministické a nedeterministické AI

Deterministické chování AI je specifikováno a lze jej předpovídat. Jednoduchým příkladem deterministického chování je např. algoritmus hlídky. Postavě přidělíme skript, ve kterém specifikujeme jednotlivé koordinace a čas, jak dlouho se na těchto místech musí zdržet. AI se pohybuje směrem k této lokaci a zastaví se v momentě, když jeho souřadnice odpovídají souřadnicím cíle. (M. Bourg, 2004)

Nedeterministické chování postavy obsahuje jistý stupeň nejistoty a jeví se hráči jako nepředvídatelné. Do jaké míry je obtížné chování AI předvídat záleží na použitých metodách vývojářem. Příkladem tohoto typu chování je např. hráčův soupeř, který zaznamenává, jakým způsobem hráč zápasí a které techniky by měl využít, aby hráč musel změnit svůj přístup. (M. Bourg, 2004)

V praxi se nejčastěji využívá deterministického chování. Metody implementace jsou rychlé a jednoduché pro testování. Jejich nadměrné využití má ovšem negativní dopad na vývojáře, kteří musí každé postavě přidělit specifický kód, jak má reagovat na hráče, dále nemotivují hráče k využití všech jeho možností atp. Navíc po krátké době hraní nabývá hráč dojmu, že je hra předvídatelná a jeho nadšení začíná klesat. Zahrnutí nedeterministických metod vývojářům šetří

čas strávený psaním skriptů. Takto naprogramované postavy jsou v jistých případech schopny odvodit si a následně se zachovat samy podle situace bez nutnosti implementace kódem. (M. Bourg, 2004)

Prototypy, jejichž vývoj je v práci ohodnocen obsahují deterministické chování, jelikož implementace nedeterministických metod vyžaduje hlubší vědomosti v programování, kterými obvykle začátečník nedisponuje.

3.1.4 Vývojářský tým

Týmy pro tvorbu her jsou většinou sestaveny ze zástupců disciplín, jako jsou např inženýři, umělci, herní designéři, producenti a další pracovníci managementu a podpory (marketing, právo, informační a technická podpora, administrativa atd.). (Gregory, 2018)

3.1.4.1 Inženýři

Hlavním úkolem inženýrů neboli techniků, je implementace a design softwaru, díky kterému bude finální produkt (hra) pracovat správně a bez problémů. Většinou se dělí na runtime a tools programátory. Runtime inženýři se soustředí na tvorbu hry a používají přitom funkce a nástroje, jejichž správu opatřuje tools odvětví.
Někteří inženýři věnují svoji kariéru ke specializaci v jediném prvku systému, ať už se jedná o renderování nebo fyzikální systémy. Převážná většina programátorů se ovšem neorientuje pouze na jednu složku, ale snaží se osvojit si co nejvíce funkcí.(Gregory, 2018)

3.1.4.2 Umělci

Neboli artisti, jsou zodpovědní za veškerou produkci zvukového a vizuálního obsahu. Na kvalitě zpracování mnohdy závisí celý úspěch produktu.

- Konceptuální umělci vytvářejí náčrty, kterými poskytnou týmu vizi ohledně finálního zpracování hry.
- 3D modeláři produkují veškeré modely, které se ve hře použijí. Člení se na modeláře prostředí (džungle, města atp.) a charakterů, tedy hráče, soupeře a interaktivních entit (NPC).
- Umělci textur navrhují dvojrozměrné obrazy, které se využívají k aplikaci na 3D modely. Poskytují tak větší detail a realistický obraz.

- Pro konfiguraci osvětlení umělci rozloží všechny zdroje světla v herním světě a pomocí nastavení úhlu, barev a intenzity záření se snaží o co největší emocionální dopad na hráče.
- Animátoři dodávají všem postavám ve hře hybnost. Některé postavy mají celý seznam animací, kterými se ve hře reprezentují. Animace dodávají postavám charakter.
- Zvukaři spolupracují s inženýry, aby byli schopni vyprodukovat adekvátní partitury hudby a zvukové efekty.

(Gregory, 2018)

3.1.4.3 Herní designéři

Jejich úkolem je zpracování interaktivní části hry pomocí skriptů a fyzického enginu. Rozhodují o tom, kde a kdy hráč nalezne důležité předměty pro posun příběhu, jakým způsobem se budeme moci bránit proti nepříteli, zda bude implementován systém dne a noci nebo jestli lze hru například dokončit více způsoby a získat tak skrytý obsah. (Gregory, 2018)

3.1.4.4 Producenti

Role producentů se liší v zájmu podniku. V některých řídí plán vývoje hry a obstarávají prostředky pro zaměstnance. Někdy pracují jako vedoucí designéři. V mnoha případech slouží jako prostředník mezi vývojovým týmem a obchodní stránkou společnosti, kde informuje obě strany o průběhu a plánech. (Gregory, 2018)

3.1.4.5 Indie Developer

Je termín, který v doslovném překladu zní „nezávislý vývojář“. Člověk s touto profesí se musí orientovat ve všech prvcích vývojářského týmu, neboť pouze na jeho nebo malé skupině vývojářů závisí celý úspěch produktu. Autor na sebe tuto roli v práci do jisté míry převezme, nebude se pouze zabývat konceptuálními návrhy, audiem a animacemi.

Vytvářet hry v takto úzkém kruhu vývojářů je umožněno právě díky univerzálním game enginům. Jsou si velice blízké aplikacím, které vytváří společnosti pro své zaměstnance, ale obsahují širokou škálu nástrojů a funkcí, pomocí kterých lze vytvářet hry každého žánru.

3.2 User Experience

UX (User Experience), neboli uživatelská zkušenost se soustředí na pochopení potřeb uživatelů, čeho si cení, jejich schopností a limitací. Metody UX se osvědčily jako podpora zlepšování interakce uživatele se systémem a navazujícími službami.
Peter Morville, president Semantic Studios, sděluje, že má-li být pro uživatele zkušenost smysluplná a hodnotná, musí informace splňovat následující kritéria:

- Užitečnost: Obsah je originální a splňuje potřebu uživatele
- Použitelnost: Produkt je snadno použitelný
- Zjistitelnost: Obsah např. na webových stránkách lze snadno lokalizovat
- Přístupnost: Obsah je konfigurován i pro osoby se zdravotním postižením
- Důvěryhodnost: Uživatelé službě důvěřují

UX je poměrně novou disciplínou, ve které se stále definují nové metody. Návrh, který sklízí největší úspěch u uživatelů obsahuje zásady Interakce člověka s počítačem.
(U.S. Dept. of Health and Human Services, 2006)

3.2.1 Usability testing

Neboli testování použitelnosti je vědní obor specifikující metody pro hodnocení aplikací, strojů, webových stránek, software atp., které provádí skupiny uživatelů. Tyto skupiny formou např. dotazníků poskytují informace o zkušenostech při používání produktů a služeb.
Autor v práci používá metodu System usability scale pro ohodnocení game enginů v kritériu Práce v enginu. V praxi tato metoda slouží jako zpětná vazba pro vývojáře, kterému reflektuje použitelnost systému.

3.2.2 Škála použitelnosti systému

Tzv. System Usabilty Scale (dále jen SUS), je ve světě jedním z nejčastěji používaných dotazníků, sloužících jako měřítko použitelnosti systémů. Jeho autorem je John Brooke, který dotazník vyvíjel ve společnosti Digital Equipment Corporation v roce 1986. SUS se v rámci UX podává uživatelům při každém průzkumu online či během testování použitelnosti. (T. Will, 2020)

V současnosti je SUS využíván především webovými stránkami, ale uplatnit jej lze i v dalších odvětvích. Týká se to především kterýchkoliv systémů, aplikací, mobilních software, PC

hardware, strojů atp. SUS ovšem neslouží jako jediná šablona online dotazníku, těch nyní existuje velké množství. Důvody ale, proč je stále tak hojně využíván jsou následující:

- Byl mnohokrát testován a lze se na jeho výsledky spolehnout.
- Lze jej jednoduše spravovat a je lehce k mání.
- Vyplnění dotazníku uživateli nezabere moc času. Rozsah dotazníku je pouze 10 položek.

I přes svoji jednoduchou vizuální stránku a rozsah byl po celou dobu od svého vývoje testován v nezměrném množství průzkumů. (T. Will, 2020)

3.2.3 Šablona SUS

Jak bylo v předešlé kapitole zmíněno, SUS obsahuje 10 jednoduchých otázek, ke kterým uživateli dotazník k výběru nabízí 5 odlišných možností. Níže uvedený SUS dotazník je specifikován pro měření použitelnosti webových stránek firmy Amazon. Podle stupnice od silně nesouhlasím do silně souhlasím uživatel ohodnocuje tato tvrzení:

- Myslím si, že bych chtěl tento web často používat.
- Web jsem shledal zbytečně komplexní.
- Používání webu se mi zdálo jednoduché.
- Pro užívání webu bych potřeboval technickou podporu pracovníka.
- Integritu funkcí na webu jsem shledal adekvátní.
- Myslím si, že se na webu vyskytuje mnoho rozporuplných tvrzení.
- Předpokládám, že se většina lidí rychle naučí používat tyto webové stránky.
- Web se mi jevil neohrabaný.
- Cítil jsem jistotu při používání webu.
- Potřeboval jsem se naučit mnoho věcí, abych mohl používat tento web.

(T. Will, 2020)

Obrázek 1 Šablona dotazníku SUS

	SUS	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1.	I think that I would like to use this website frequently.	☐	☐	☐	☐	☐
2.	I found the website unnecessarily complex.	☐	☐	☐	☐	☐
3.	I thought the website was easy to use.	☐	☐	☐	☐	☐
4.	I think that I would need the support of a technical person to be able to use this website.	☐	☐	☐	☐	☐
5.	I found the various functions in this website were well integrated.	☐	☐	☐	☐	☐
6.	I thought there was too much inconsistency in this website.	☐	☐	☐	☐	☐
7.	I would imagine that most people would learn to use this website very quickly.	☐	☐	☐	☐	☐
8.	I found the website very cumbersome / awkward to use.	☐	☐	☐	☐	☐
9.	I felt very confident using the website.	☐	☐	☐	☐	☐
10.	I needed to learn a lot of things before I could get going with this system.	☐	☐	☐	☐	☐

Zdroj: UIUXTrend.com, 2021

3.2.4 Kalkulace SUS

V prvním kroku výpočtu SUS přidělujeme jednotlivým možnostem výběru bodové ohodnocení. Silně nesouhlasím odpovídá jednomu bodu, Nesouhlasím dvěma, Neutrální je za tři body, Souhlasím za 4 a Silně souhlasím za 5 bodů. (T. Will, 2020)

Následně vytvoříme rovnici

Výsledek SUS = (X + Y) * 2,5

‚při které součet bodů všech lichých otázek a odečtení pěti reprezentuje „X" a součet bodů všech sudých otázek a následné odečtení 25 se značí „Y". Struktura této rovnice vyplývá z odůvodnění, že každá z otázek má váhu 10 a maximální možné ohodnocení dotazníku je 100 bodů. Jelikož jsou všechny liché otázky pozitivní, pokud uživatel zvolí odpověď Silně souhlasím, budeme ji chtít ohodnotit 10 body. Pokud je zvolena možnost Silně nesouhlasím, ohodnotíme ji 0 body. Opačně přistupujeme k sudým otázkám, které jsou negativní. Pokud uživatel zvolí odpověď Silně souhlasím, hodnocena bude nula body atd. (T. Will, 2020)

3.2.5 Interpretace hodnocení

Hodnocení uživatele u každé otázky převádíme na nové číslo, které dále násobíme 2,5, což nám mění bodové ohodnocení na 0 až 100. Pokud se celkové skóre dostane nad 68 bodů, je hodnocení nadprůměrné. (U.S. Dept. of Health and Human Services, 2006)

Tabulka 1: Interpretace hodnocení dotazníku SUS

SUS Skóre	Známka	Hodnocení
> 80,3	A	Výborné
68 - 80,3	B	Velmi dobré
68	C	Dobré
51–68	D	Dostatečné
< 51	F	Nedostatečné

Zdroj: uiuxtrend.com/measuring-system-usability-scale-sus, 2021

3.3 Vícekriteriální rozhodování

Při řešení rozhodovacích problémů, ve kterých se rozhodnutí hodnotí podle většího množství kritérií, využijeme modelů vícekriteriálního rozhodování. Charakteristika více kritérií je přítomná v téměř všech rozhodovacích situacích. Cílem těchto modelů je nalezení nejlepšího řešení vyplývajícím ze všech kritérií, eliminace neúčinných variant, nebo organizace množiny řešení. (Šubrt, 2011)

„Modely vícekriteriálního hodnocení variant jsou zadány pomocí konečného seznamu variant a jejich ohodnocení podle jednotlivých kritérií.
Modely vícekriteriální optimalizace mají množinu variant s nekonečně mnoho prvky, vyjádřenou pomocí omezujících podmínek a ohodnocení jednotlivých variant je dáno jednotlivými kriteriálními funkcemi.“ (Šubrt, 2011)

3.3.1 Model vícekriteriální analýzy variant

Teorie vícekriteriální analýzy variant zkr. VAV, se věnuje řešením sporů při výběru jedné nebo celé skupiny variant přípustných k řešení, které následně doporučuje k provedení. V průběhu rozhodování se snažíme postupovat objektivně. K uplatnění se nám nabízejí metody VAV. (Šubrt, 2011)

Varianty představují různé možnosti volby, které má rozhodovatel k dispozici. Při postupu zpracování nejprve jednotlivé alternativy prověříme, usoudíme jejich prioritu a nakonec ohodnotíme. (Triantaphyllou, 2010)

„Kritérium je hledisko hodnocení variant, může být kvalitativní nebo kvantitativní.“ (Šubrt, 2011) Velmi důležitá je i volba každého kritéria. Aby byl problém přehledný, je zapotřebí stanovit kritéria nezávislá, která pokrývají všechna stanoviska výběru v přiměřeném počtu. (Šubrt, 2011)

V případě velkého množství rozhodovacích kritérií se doporučuje je seřadit v hierarchickém pořadí, tzn. posuzovat podle kritérií s vyšším stupněm ohodnocení. (Triantaphyllou, 2010)

3.3.2 Kritéria hodnocení

Kritéria určená pro volbu nevýhodnější varianty, lze rozdělit na základě rozličných hledisek. Jedním takovým hlediskem je např. povaha. (Šubrt, 2011)

„Kritéria maximalizační: při rozhodování vycházíme z toho, že nejlepší varianty podle tohoto kritéria mají nejvyšší hodnoty. Kritéria minimalizační: opak maximalizačního kritéria, nejlepší varianty mají nejnižší hodnoty podle tohoto kritéria.“ (Šubrt, 2011)

Snažíme se pracovat s množinou kritérií totožné povahy, tedy se všemi maximalizačními nebo minimalizačními, což nám přináší jisté výhody. V počátcích stanovení problému a kritérií tomu ovšem tak často není, a proto se dále řídíme těmito dvěma způsoby: (Šubrt, 2011)

- „Vynásobení celého sloupce kriteriální matice hodnotou -1, transformace

$$y'_{ij} = -y_{ij}$$

- Výpočet hodnot, které udávají zlepšení oproti nejhorší kriteriální hodnotě, transformace

$$y'_{ij} = y_{ij} - \max(y_{ij}).“$$

(Šubrt, 2011)

První způsob je vždy matematicky plně korektní, ale interpretace nového kritéria nemusí být na první pohled jasná.“ (Šubrt, 2011)

Pokud bychom například nahradili cenu produktu, což je minimalizační kritérium, maximalizačním kritériem záporná cena produktu, začali bychom se v problému obtížněji orientovat. Po interpretační stránce je druhý způsob zcela srozumitelný. Nahradíme-li cenu produktu např. úsporou nad nejdražším produktem, získáme nulové ohodnocení nejdražšího produktu a kladné hodnoty u zbylých produktů. V praxi ovšem nelze vždy využít zřetelnější druhý způsob, jelikož by to pro jisté metody VAV zkreslovalo úvodní informace, což by mělo za následek zásadní ovlivnění výsledku. Dalším hlediskem je tzv. kvantifikovatelnost, podle kterého dělíme kritéria na: (Šubrt, 2011)

- „Kritéria kvantitativní: hodnoty variant podle takovýchto kritérií tvoří objektivně měřitelné údaje, proto se také tato kritéria nazývají objektivní.
- Kritéria kvalitativní: hodnoty variant podle těchto kritérií nelze objektivně změřit, velmi často jde o hodnoty subjektivně odhadnuté uživatelem. V těchto případech se používají různé bodovací stupnice nebo relativní hodnocení variant (jedna varianta je zvolena jako základ a uživatel odhaduje procentní vyjádření ostatních variant).“(Šubrt, 2011)

Preference kritéria vyjadřuje důležitost tohoto kritéria v porovnání s kritérii ostatními.“ (Šubrt, 2011)

Existuje několik různých způsobů vyjádření preference, závisejících na těchto údajích:

- „aspirační úrovně kritérií (nominální informace o kritériích)
- pořadí kritérií (ordinální informace o kritériích)
- váhy jednotlivých kritérií (kardinální informace o kritériích)
- způsob kompenzace kriteriálních hodnot
- anebo nemusí být známa vůbec.” (Šubrt, 2011)

Přidělení preference ke kritériím je jedním z nejobtížnějších úkolů pro rozhodovatele, jelikož se vztahuje na jeho subjektivní názor. (Šubrt, 2011)

3.3.3 Aspirační úrovně

Vyjadřuje požadavek, kterého by měla varianta v daném kritériu dosáhnout. Pokud se jedná o maximalizační kritérium, aspirační úroveň odpovídá nejnižší možné hodnotě. V případě minimalizačního kritéria je tomu naopak, hovoříme potom o nejvyšší povolené hodnotě. . (Šubrt, 2011)

Aspiračními úrovněmi nestanovujeme přímo preferenci kritérií, pouze čeho by mělo být dosaženo. Pokud má ovšem kritérium těžce dosažitelnou hodnotu, vypovídá to o jeho důležitosti. (Šubrt, 2011)

3.3.4 Váha kritéria

Hodnoty vah kritérií nám sdělují, jak důležitá jednotlivá kritéria jsou. Obecný postup určuje každou váhu v intervalu <0;1>. Výsledek součtu všech vah je vždy roven jedné. Následným uspořádáním kritérií podle jejich důležitosti získáváme pořadí kritérií. Pokud je varianta ve všech kritériích vyhodnocena lépe než alternativa, jedná se o tzv. dominující variantu a alternativa je v tomto případě variantou dominovanou. . (Šubrt, 2011)

Efektivní varianta neboli paretovská je taková varianta, kterou žádná alternativa nedominuje. Může získat výhodnější hodnocení v určitém kritériu za předpokladu, že se zhorší kritérium jiné. (Šubrt, 2011)

3.3.5 Ideální a bazální varianta

Varianta je ideální, pokud se ve všech kritériích vyskytuje s nejlepším ohodnocením. Pravý opak ideální varianty je varianta bazální, tedy alternativa, která se v jednotlivých kritériích vyskytuje s nejnižšími hodnotami. V praxi se zpravidla s těmito variantami nesetkáváme, jelikož by byla ideální varianta jediná nedominovaná, a proto by pro nás prezentovala optimální řešení. (Šubrt, 2011)

3.3.6 Kompromisní varianta

„Kompromisní varianta je nedominovaná varianta doporučená jako řešení problému." (Šubrt, 2011)

Selekce kompromisní varianty je spojena s postupem řešení. Naším cílem může být nález jedné varianty, či všech efektivních variant. Způsobů stanovení kompromisní varianty existuje několik: (Šubrt, 2011)

- Jedná se např. o variantu, která je k ideální variantě nejblíže. Tím máme na mysli vzdálenost, kterou interpretujeme jako ohodnocení kritéria varianty vůči kritériu ideální varianty.

- Dalším případem je porovnávání v párech. Jedná se tedy o všechny dvojice variant a posuzování hodnocení ve všech jejich kritériích. (Šubrt, 2011)

3.3.7 Definování problému VAV

Úlohy řešené ve VAV rozdělujeme podle těchto dvou hledisek:

- cíl
- informace

Podle hlediska „cíl" dále úlohy třídíme na tři základní kategorie. (Šubrt, 2011)

Výběr jedné varianty, kterou následně označíme za kompromisní se zabývá první kategorie úloh. Z nabízených variant vybereme podle kritérií nejlépe ohodnocenou variantu. Tento výběr probíhá pomocí metod posuzování variant. V této kategorii nám nevyhovují aspirační úrovně kritérií z hlediska informace preferencí mezi kritérii. (Šubrt, 2011)

Seřazení variant, které obvykle bývá od nejlepší varianty po nejhorší. Zde se setkáváme s podobným postupem jako v první kategorii. Po zjištění kompromisní varianty ji ze seznamu vyškrtáváme a proces bez ní znovu opakujeme. Takto z každé nové množiny variant získáváme kompromisní varianty, kterým přidělujeme jejich respektivní umístění (např. druhé místo, třetí místo atd.). Metody a restrikce posuzování variant jsou v této kategorii totožné s kategorií první. (Šubrt, 2011)

Ve třetí kategorii se zabýváme rozdělením variant na paretovská, neboli efektivní a neefektivní. Naším cílem není určení nejlepší varianty ani seřazení variant, ale jejich posouzení, zda vyhovují stanoveným kritériím nebo ne. (Šubrt, 2011)

Podle hlediska „informace" jsou úlohy rozděleny do čtyř okruhů.

- v prvním okruhu se nevyskytuje žádná informace o preferencích kritérií
- informace nominální, která se vyskytuje pouze mezi preferencemi kritérií a je zastoupena aspiračními úrovněmi, ve kterých varianty podle jejich náležitých kritérií dělí na přípustné a nepřípustné
- informace ordinální se zabývá uspořádáním variant podle jejich ohodnocení kritériem nebo uspořádáním kritérií dle jejich významu
- informace kardinální mají kvalitativní a kvantitativní složku. Sdělují rozhodovateli, v jakém poměru je jedno vyhodnocení lepší než ostatní. Pokud se jedná o vyhodnocení kritérií, bavíme se o vahách a v případě vyhodnocení variant podle kritérií o jeho vyjádření obvykle pomocí čísel (Šubrt, 2011)

3.3.8 Stanovení vah kritérií

Rozhodovatel obvykle stanovuje váhy kritérií již v prvním kroku úlohy VAV pro určení důležitosti kritérií. Metody stanovení vah jsou rozděleny dle informací o preferencích kritérií (žádná, nominální, ordinální, kardinální). (Šubrt, 2011)

3.3.8.1 Metoda pořadí

Se nejčastěji používá při větším počtu rozhodovatelů, kteří kritéria jednotlivě ohodnotí. Váha kritéria se určí jako součet bodů kritéria a následné vydělení součtem bodů ze všech kritérií. (Šubrt, 2011)

3.3.8.2 Fullerova metoda

Tato metoda se zabývá porovnáním všech možných párů kritérií, přičemž důležitějšímu z nich je přidělen jeden bod. Proces porovnávání se provádí v tzv. Fullerově trojúhelníku, kdy rozhodovatel zadává údaje do tabulky. Pro výpočet váhy kritéria se používá vzorec:

$$N = \frac{n\,(n-1)}{2}$$

kde „n“ je množství kritérií. (Šubrt, 2011)

3.3.8.3 Bodovací metoda

Bodovací metoda patří mezi nejoblíbenější postup rozhodovatelů pro svoji jednoduchost. Nejprve přiřadíme bodové ohodnocení jednotlivým kritériím, přičemž lze i větší počet kritérií obodovat stejně. Postup výpočtu vah je pak totožný s postupem metody pořadí. (Šubrt, 2011)

3.3.8.4 Saatyho metoda

Je vhodná metoda pro hodnocení jedincem. Rozhodovatel používá stupnici devíti bodů pro určení preferencí kritérií.

1 - rovnocenná kritéria i a j
3 - slabě preferované kritérium i před j
5 - silně preferované kritérium i před j
7 - velmi silně preferované kritérium i před j
9 - absolutně preferované kritérium i před j

Hodnoty porovnání kritérií jsou dále zapsány do Saatyho matice. Na diagonále se vyskytuje pouze jednička, protože si jsou sama sobě kritéria rovna. Pokud je preferováno kritérium j před i, je hodnota v matici převrácena. Řešitel se při přiřazování preferencí může dopustit nekonzistence. Poměr nekonzistence lze získat pomocí vzorce:

$$Is = \frac{Imax - n}{n - 1}$$

kde „I_{max}“ je největší kořen polynomu Saatyho matice a „n“ počet kritérií.
Pokud je míra konzistence větší než 0,1, je nutné Saatyho matici předělat a výpočet provést znovu. Pokud je míra konzistence menší, je matice dostatečně konzistentní a lze pokračovat ve výpočtech. Jelikož by odhad vah byl náročný přímo z matice, navrhl Thomas L. Saaty pro výpočet několik různých způsobů. Z nich je nejčastěji používán geometrický průměr řádků matice. Po výpočtu průměru jsou váhy získány normalizací těchto hodnot. Saatyho metoda je používána i pro stanovení preferencí mezi variantami. (Šubrt, 2011)

3.3.9 Výběr kompromisní varianty

Metod pro výběr kompromisní varianty existuje v praxi velké množství, a proto jsou v práci popsány pouze někteří zástupci z hledisek požadovaných informací ohledně preferencí kritérií.

3.3.9.1 Bodovací metoda a metoda pořadí

Jsou metody, které pro své uplatnění nevyžadují informace o preferencích kritérií. Rozhodovatel nejprve ohodnocuje varianty podle stupnice bodů od jedné do deseti v metodě bodovací, kde nejlepší variantě v kritériu přidělí hodnotu deset. V metodě pořadí rozhodovatel variantám v kritériích přiřazuje jejich příslušné umístění.
Součet bodů se v jednotlivých metodách interpretuje odlišně. Pomocí bodovací metody hledá rozhodovatel tu variantu, jejíž součet bodů je nejvyšší. Tato varianta je v kritériích nejlépe hodnocena oproti alternativám. Metoda pořadí naopak určuje kompromisní variantu podle nejmenšího součtu. (Šubrt, 2011)

3.3.9.2 Konjunktivní a disjunktivní metoda

Tyto metody rozhodovatel použije v případě, že je seznámen s nominální informací o kritériích neboli aspiračními úrovněmi a nehledá nejlepší možnou variantu, ale množinu variant, která vyhovuje aspiračním úrovním kritérií.
Při konjunktivní metodě musí vyhovující varianty splňovat všechny aspirační úrovně kritérií, zatímco v disjunktivní metodě pro je varianta akceptovatelná tehdy, pokud splňuje alespoň jednu aspirační úroveň. (Šubrt, 2011)

3.3.9.3 Lexikografická metoda

Vyžaduje ordinální informace o kritériích anebo variantách. Výběr kompromisní varianty lexikografickou metodou je určen nejdůležitějším kritériem. V případě, že je podle nejdůležitějšího kritéria ohodnoceno více variant stejně, se kompromisní varianta určuje podle dalšího nejdůležitějšího kritéria. Takto se postup může opakovat, dokud v kritériu nezůstává nejlépe ohodnocena jediná varianta. (Šubrt, 2011)

3.3.9.4 Metoda váženého součtu

Již vyžaduje, aby rozhodovatel znal kardinální informace o kritériích neboli jejich vah. Nabízí celkové hodnocení jednotlivých variant a je tedy vhodnou volbou pro výběr jedné nejvýhodnější varianty nebo seřazení množiny variant. Nejprve si rozhodovatel určí ideální a bazální variantu. Následně pro jednotlivé varianty pomocí skalárního součinu řádků standardizované kriteriální matice a řádku vah vytvoříme hodnoty agregovaných funkcí užitku, podle kterých určuje rozhodovatel pořadí variant. (Šubrt, 2011)

4 Vlastní práce

Postup zpracování prototypů a hodnocení jednotlivých vychází z teoretických východisek. Vývoj provádí sám autor za pomoci dokumentace na oficiálních stránkách game enginů a návodů online. Vývojová prostředí jsou v podkapitolách dále popsána a vychází z nich ohodnocení variant ve zvolených kritériích vícekriteriálního rozhodování.

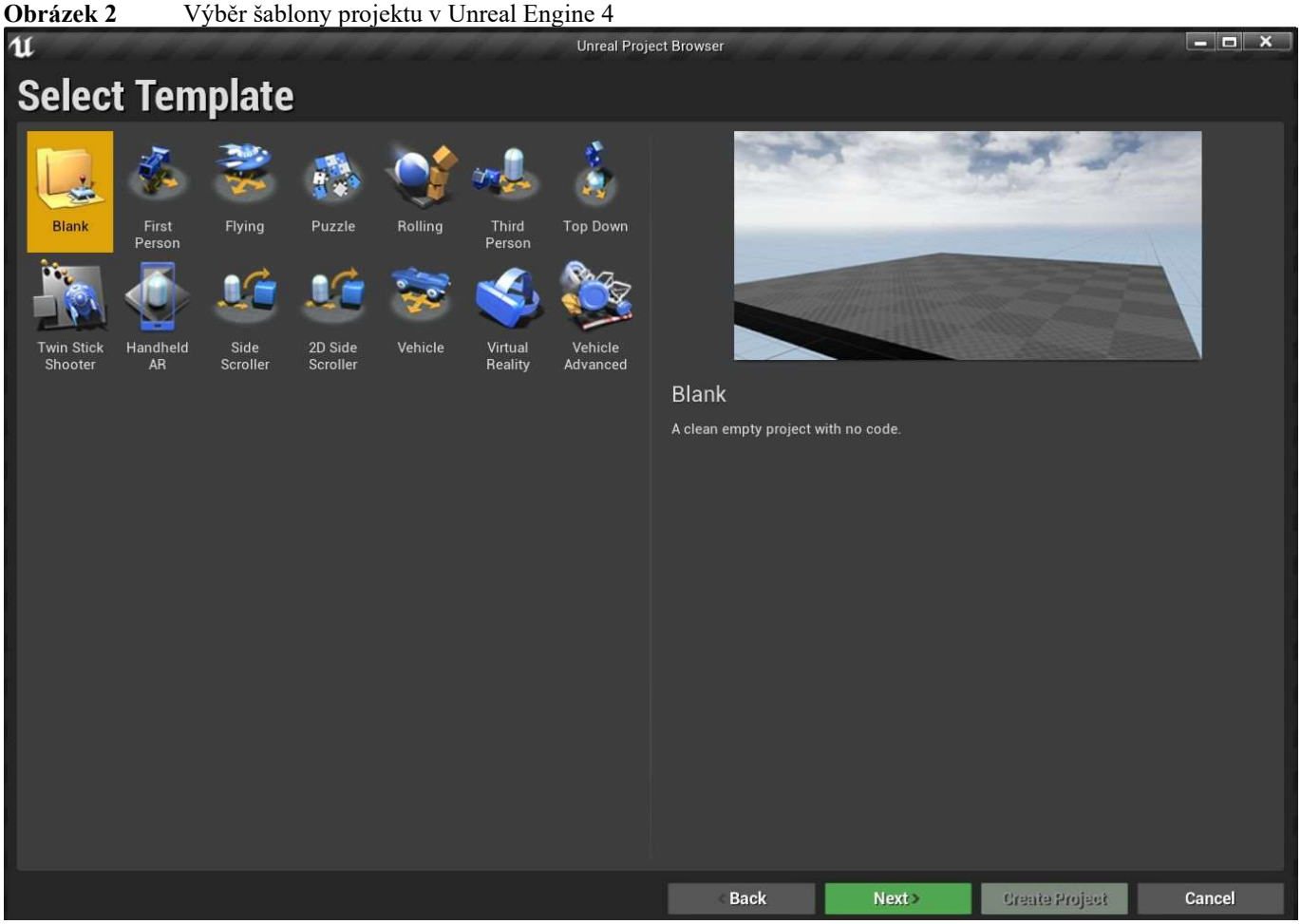
4.1 Vývoj prototypů

Prototypy obsahují nebo v nich jsou zakódovány tyto prvky:

- Hrací plochu ve 3D scéně
- Hratelný objekt hráče
- Soupeře AI
- Závodní dráhu
- Překážky
- Rampy
- Podmínky výhry a prohry
- Importované modely z programu Blender
- Úvodní menu ve 2D scéně

Pro vývoj prototypů her je potřeba instalace programů Unity3D, CryEngine, Unreal Engine 4, Godot, IDE Visual Studio a Blender. Postup vývoje je rozdělen do částí, aby byly získané údaje z jednotlivých programů snadněji interpretovatelné. Každý engine nejprve nainstaloval tzv. launcher pro správu řízení projektů. Zde lze ukládat a znovu otevírat rozpracované projekty.

Obrázek 2 Výběr šablony projektu v Unreal Engine 4



Zdroj: Unreal Engine Browser, 2021

Po vytvoření nového projektu nám launcher nabízí tzv. šablony, které definují první scénu prototypu. Tyto vlastnosti lze během vývoje dále konfigurovat.

Nejprve je v jednotlivých enginech vytvořena 3D scéna, ve které je navržena totožná závodní dráha. Autor vytváří herní plochu, na kterou dále umísťuje překážky, rampy a propasti. Tyto prvky budou po implementaci hráče spolu s AI určovat míru obtížnosti.

Po vybudování trasy přichází na řadu přidání objektu, který plní funkci zastoupení modelu automobilu. Tento automobil bude na závěr modelován v programu Blender a z něj také importován do jednotlivých enginů. Objektu na ploše se přidává první skript, který umožňuje uživateli pohybu po dráze pomocí kláves W, A, S, D.

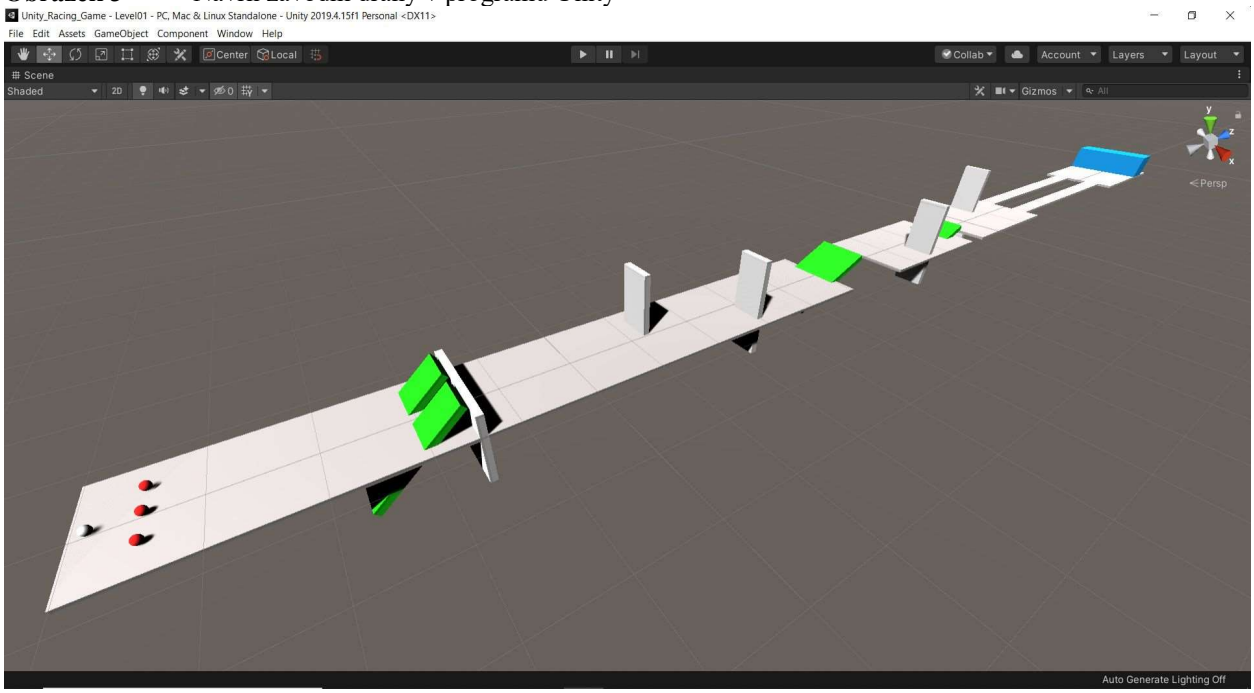
Dalším krokem je vytvoření objektů, které nyní zastupují roli AI. Tyto objekty se od hráče prozatím liší barvou. Aby bylo AI umožněno dostat se do cíle, je nutné umístit na konec dráhy předmět, plnící funkci cílové čáry a tzv. bake doposud vytvořenou dráhu. Tato metoda v game enginech reprezentuje vytvoření pomyslné plochy, na které je AI povoleno se pohybovat.

Následuje přidání skriptu, který po spuštění prototypu automaticky přikazuje přiřazeným objektům pohybovat se do cíle.
Nyní se nacházíme ve fázi, kde je prototyp hratelný, ovšem postrádá ještě několik prvků. Pokud při závodu například hráč odbočí z trasy, fyzický engine způsobí neustálý pád hráče a hru nelze nijak opakovat. Zavedeme tedy nový skript, který bude kontrolovat, zda poloha hráče po ose kolmé k dráze nenabývá záporných hodnot a kdo se jako první dostane do cíle. Pokud některá z těchto podmínek nastane, hráči se zobrazí, zda vyhrál nebo prohrál a po chvilkovém zpoždění se hra znovu opakuje.
Prototypu ještě přidáme novou 2D scénu pro funkci hlavního menu s nadpisem a tlačítkem pro zahájení závodu.
Po otestování prototypu jsou objekty hráče a AI nahrazeny modely z Blenderu.

4.1.1 Vývojové prostředí Unity

Při zahájení nového projektu je uživateli umožněno vybrat si v Unity jednu z pěti nainstalovaných šablon. Seznam obsahuje i šablony, které jsou volně ke stažení. Autorem byla zvolena 3D šablona. Autorův první dojem byl pozitivní, jelikož vývojové prostředí na uživatele nepůsobí přehlceným dojmem. Pomocí panelu hierarchie lze do scény jednoduše přidávat objekty, jejichž obsah, pozici a tvar lze nastavit podle souřadnic v panelu inspektor nebo funkcí v horní liště programu. Autor byl tak schopen brzy navrhnout podle základních objektů plochu, která bude sloužit jako závodní dráha.

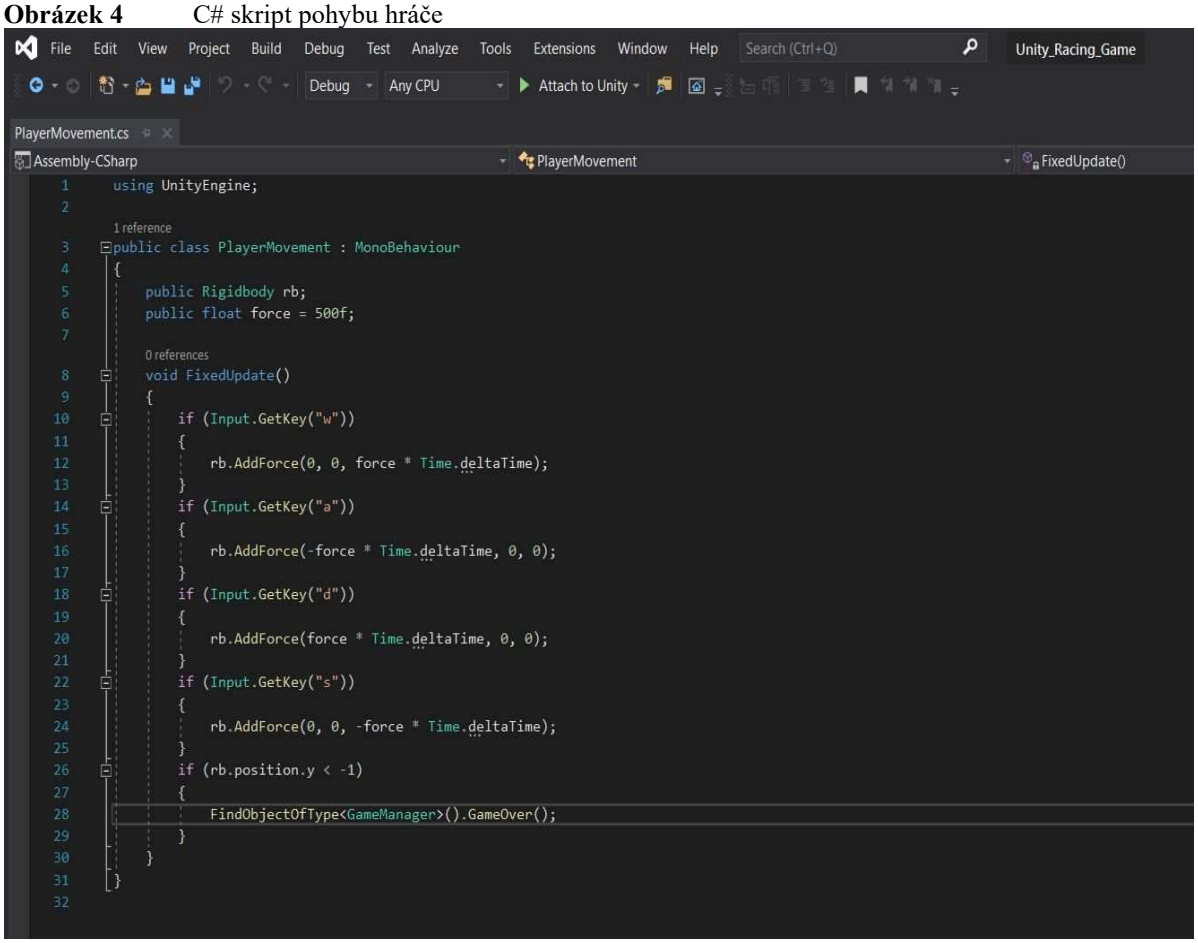
Obrázek 3 Návrh závodní dráhy v programu Unity



Zdroj: Unity3D, 2021

Po ukončení návrhu byl vytvořen objekt hráče, kterému byl přidělen první skript. Jednalo se o zavedení funkce pohybu hráče.

Obrázek 4 C# skript pohybu hráče



Zdroj: Visual Studio, 2021

Postup vývoje pokračoval bez větších nesnází do okamžiku, kdy nenastala řada na implementaci AI. Samotné kódování a stanovení plochy pro pohyb proběhlo úspěšně, ale potíže se vyskytly, když mělo AI překonat překážku pomocí rampy. Z dokumentace se autor dozvěděl, že funkce pro spojení dvou ploch, mezi kterými je mezera, v game enginu není obsažena a je nutné si ji nainstalovat. Bylo tedy nutné využít externích zdrojů.

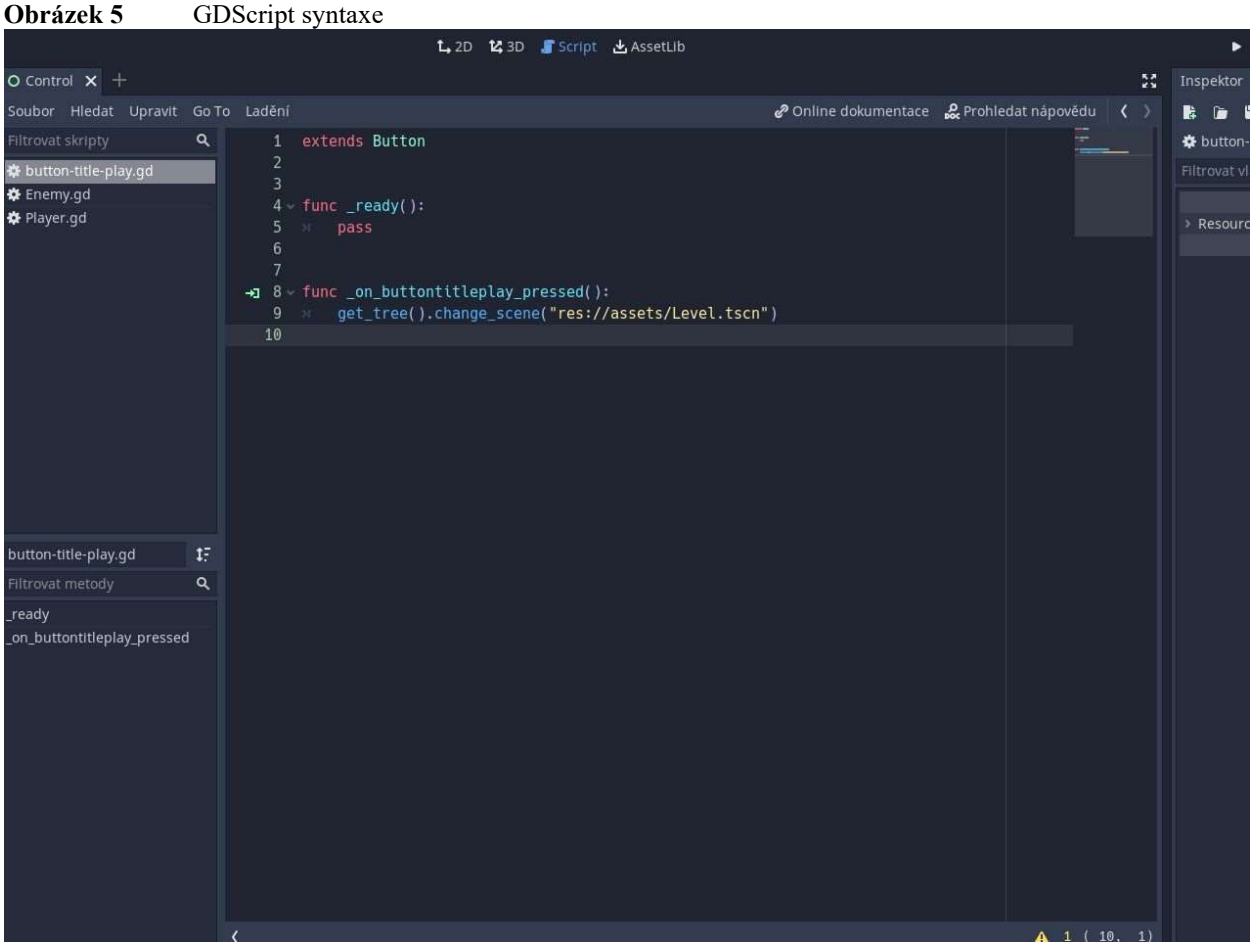
Při vývoji nenastaly další větší úskalí, autor v Unity dokončil prototyp nejrychleji ze všech game enginů především pomocí velkého množství návodů na internetu a intuitivního návrhu UI.

4.1.2 Vývojové prostředí Godot

Při zahajování projektu nabízí game engine Godot velké množství standardních a uživateli vytvořených šablon.

Stromová architektura uzlů a scén se autorovi nejprve jevila příliš komplikovaná. Po krátkém nastudování dokumentace a vytvoření základních objektů si na ni ovšem brzy navykl a vývoj pokračoval plynule. Psaní skryptů v jazyce GDScript autorovi nedělalo potíže. Syntaxe se mu jevila jednoduchá a uživatelsky přívětivá. V Godotu je zabudován IDE přímo pro GDScript. Autor byl schopen psát kód a přidávat tak vlastnosti objektům během spuštěného prototypu. Uživatel proto rychle získává přehled o efektu jednotlivých funkcí. To jsou hlavní důvody, proč získal v dotazníku SUS programovací jazyk nejlepší ohodnocení.

Obrázek 5 GDScript syntaxe



Zdroj: Godot Engine, 2021

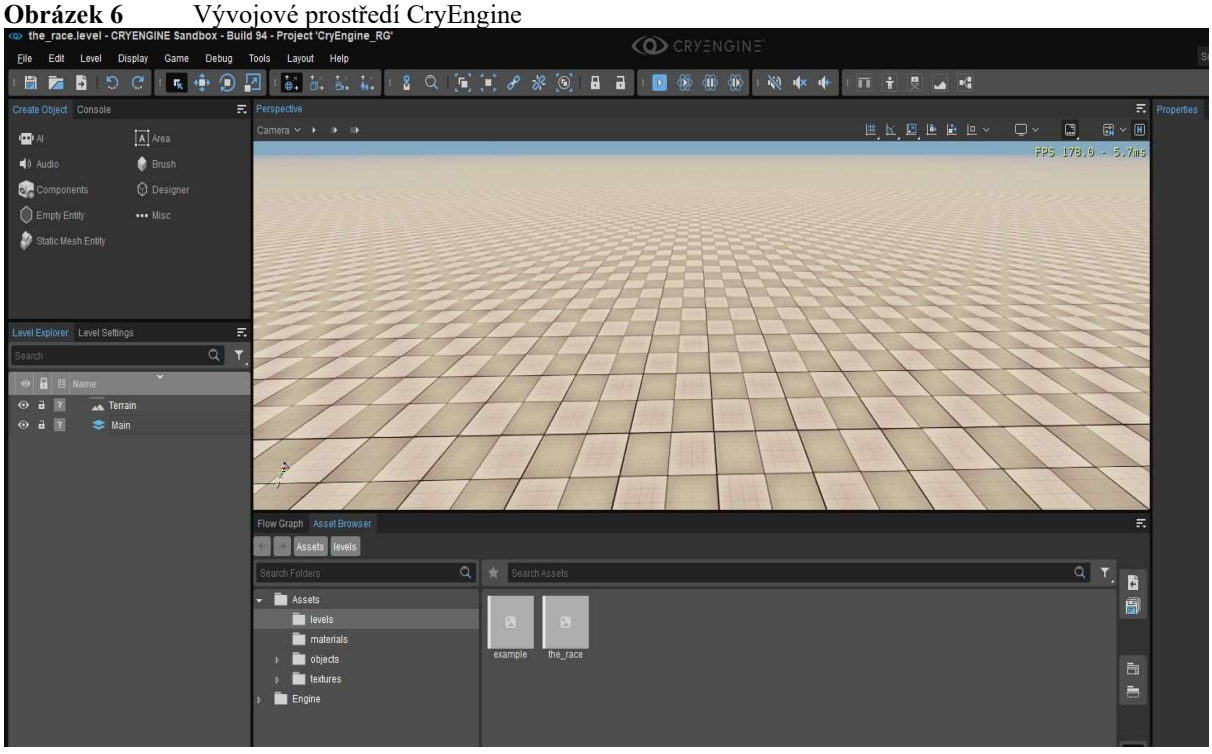
Problém podobně jako v programu Unity nastal při implementaci AI. Ovšem v případě game enginu Godot se nejednalo o chybějící funkce, ale malého množství materiálu online. Protože

jsou názvy funkcí pro navigaci AI v jednotlivých enginech téměř totožné, autor byl schopen dohledat postup pomocí jiných dokumentací. Nicméně pokud by se jednalo o uživatele, který vytváří projekt pouze podle Godot dokumentace, bylo by pro něj velice obtížné dosáhnout optimálních výsledků.

4.1.3 Vývojové prostředí CryEngine

CryEngine se svým designem nejvíce podobá game enginu Unity. Navigovat se v UI autorovi problém nedělalo, ale design zde není tak uživatelsky přívětivý. Obsahuje úzkou paletu barev a oproti Unity, které například obsahuje ikony funkcí pro rychlou orientaci při definování objektů, postrádá CryEngine podobnou implementaci.

Obrázek 6 Vývojové prostředí CryEngine



Zdroj: CryEngine, 2021

Následuje editace plochy, aby model odpovídal návrhu závodní dráhy. Autor nebyl spokojen s prací v enginu a vývoj prototypu mu proto ve vývojovém prostředí CryEngine trval nejdéle. Kromě dokumentace bylo obtížné vyhledat informace o upravení herní plochy nebo například vytvoření hlavního menu. Jednou z mála výhod, které engine uživateli poskytuje je možnost výběru skriptovacího jazyka C#. Dokumentace neuvádí, který programovací jazyk je určen jako výchozí a v launcheru je možnost výběru C++ a C#. Jelikož se autorovi jazyk C# zdál

přívětivější pro začátečníky a při vývoji prototypu v Unity s ním neměl potíže, zvolil si ho i v CryEngine.

4.1.4 Vývojové prostředí Unreal Engine

Návrh UI je v Unreal Engine odlišný od ostatních enginů v tom, že obsahuje velké ikony a široký obsah viditelných funkcí. Jsou zde použity pestré barvy pro orientaci uživatele v rozhraní. Tyto pozitivní prvky ovšem zastínila skutečnost, že Unreal Engine nabízí objekty, se kterými se autor doposud nesetkal, a to tzv. Actors. Jedná se o třídu, která podporuje pohyb po herní ploše a interakci s hráčem. Existuje jí více typů, jako například CameraActor, PlayerStartActor atd. Práce s novými typy objektů byla pro autora zpočátku překážkou. Během vývoje prototypu se jí dokázal naučit ovládat na optimální úrovni, ovšem výhodnější se mu zdál návrh objektů v ostatních enginech. Programování v C++ bylo oproti jazykům GDScript a C# obtížnější kvůli odlišné syntaxi a názvosloví funkcí. Unreal Engine se těší veliké popularitě, a proto byl autor schopen vypracovat skripty i v tomto jazyce. Mnoho tutoriálů, které směřují na začátečníky ovšem programování prezentují ve vizuálním skriptování, a proto není jednoduché nalézt ke všem problémům řešení online v jazyce C++.

4.2 Hodnocení použitelnosti

Tabulka 2 Hodnocení CryEngine dotazníkem SUS

SUS CryEngine	Silně nesouhlasím	Nesouhlasím	Neutrální	Souhlasím	Silně souhlasím
Myslím si, že bych chtěl tento herní engine často používat.		X			
Herní engine jsem shledal zbytečně komplexní.	X				
Používání game enginu se mi zdálo jednoduché.		X			
Pro užívání game enginu bych potřeboval technickou podporu pracovníka.			X		
Integritu funkcí v game enginu jsem shledal adekvátní.		X			
Myslím si, že se v game enginu vyskytuje mnoho rozporů.		X			
Dokážu si představit, že se většina lidí bude schopna rychle naučit používat tento game engine.		X			
Game engine mi připadal velmi těžkopádný.		X			
Při používání game enginu jsem se cítil velmi sebejistě.			X		
Musel jsem se naučit mnoho věcí, než jsem mohl začít pracovat s tímto game enginem.		X			

Zdroj: autor

Pro hodnocení použitelnosti je autorem vyplněn dotazník SUS pro jednotlivé game enginy a programovací jazyky. Výše je názorné vyplnění dotazníku pro vývojové prostředí CryEngine. Výsledek hodnocení dotazníku odpovídá skóre 52,5 bodů, což odpovídá podprůměrnému hodnocení.

Výsledky hodnocení dotazníků pro Práci v enginech jsou následující:

- Unity obdrželo 82,5 bodů.
- CryEngine obdržel 52,5 bodů.
- Godot obdržel 72,5 bodů.
- Unreal Engine obdržel 57,5 bodů.

Hodnocení programovacích jazyků vychází následovně:

- C# obdržel 70 bodů.
- C++ obdržel 55 bodů.
- GDScript obdržel 72,5 bodů.

4.3 Rozhodování podle vícekriteriální analýzy variant

Podle nastudované literatury se nejprve musí definovat množina variant, kritéria hodnocení a cíl. Variantami jsou game enginy Unity3D, Unreal Engine 4, Godot a CryEngine.
Kritéria hodnocení jsou následující:

- práce v enginu
- komunita
- výchozí skriptovací jazyk
- implementace souborů z jiných programů
- rozsah funkcí enginu

Cílem bakalářské práce je zvolit nejvýhodnější game engine pro začátečníky. Autor práce nemá žádné předchozí zkušenosti s tvorbou her v game enginech a zanedbatelnou praxi v programovacím jazyce C#. Výsledky analýzy jsou především cílené na uživatele začínajícího v oboru vývoje her.

4.3.1 Odůvodnění volby kritérií

Kritérium Práce v enginu obsahuje schopnost autora orientovat se ve vývojářském prostředí programů. Do jaké míry autor dokázal intuitivně implementovat objekty, navrhnout dráhu a hráče spolu s AI. Ohodnocení vypovídá o míře spokojenosti autora při tvorbě prototypu. Varianty jsou v kritériu znázorněny číselnými údaji, které vychází z výsledků SUS dotazníků.

Komunita jako kritérium je zvolena z toho důvodu, že některé prvky se repetitivně ve hrách vyskytují a existují osvědčené způsoby, jak je lze do hry zabudovat. Čím větší je komunita programu, tím snazší je vyhledání více specifických požadavků uživatele. Na internetových fórech Reddit aj. může začínající vývojář podat dotaz nebo se podle volně dostupných návodů na YouTube naučit lépe pracovat v enginu. Pro zjednodušení výpočtu jsou číselné údaje komunit reprezentovány aktuálními uživateli fór game enginů na webu Reddit. Hodnoty jsou uvedeny v tisících.

Výchozí skriptovací jazyk je kritériem pro porovnání programovacích jazyků C#, C++, GDscript. V teoretické části práce jsou v jednotlivých kapitolách herních enginů uvedeny možnosti programování. Uživatel může programovat v jednom nebo celé skupině jazyků. V praktické části jsou hodnoceny pouze výchozí skriptovací jazyky, jelikož pro ně existuje v jednotlivých dokumentacích a tutoriálech na internetu největší množství informací. Varianty jsou zde číselně ohodnoceny obdobně jako v kritériu Práce v enginu pomocí dotazníku SUS.

Implementace souborů z jiných programů je hodnocena v rámci importace objektů z Blenderu. Autor v postupech popisuje, jakým způsobem se model v enginech integruje, jaké získává ve vývojovém prostředí vlastnosti a zda je možné objekt dále editovat. Jakým způsobem a zda existují pomůcky pro zabudování modelu do prototypu jsou hlavními otázkami, kterým hodnoty kritérií odpovídají. Varianty jsou v kritériu vyjádřeny slovně, ovšem autor je přeměňuje do číselné podoby podle bodové stupnice od jedné do deseti, kde deset odpovídá nejlepšímu ohodnocení a jedna nejhoršímu.

Rozsah funkcí enginu hodnotí, zda pro tvorbu prototypu jednotlivé enginy obsahují všechny potřebné funkce. Při vývoji se musí přidělit objektům vlastnosti, podle kterých s nimi zachází fyzický engine. Dalším příkladem je implementace AI. Engine musí pro splnění požadavků prototypu obsahovat funkce, pomocí kterých uživatel určí plochu pohybu soupeřů. Pokud engine některé funkce při instalaci neobsahuje, ovšem jsou volně dostupné na internetu, je hodnocení negativně ovlivněno. Údaje jsou vyjádřeny slovně a v tabulce jsou do číselné podoby

převedeny stupnicí od jedné do deseti identickým způsobem, jako je tomu v kritériu Implementace souborů z jiných programů.

4.3.2 Metoda stanovení vah kritérií

Na základě poznatků z teoretické části byla autorem zvolena Saatyho metoda. Důvodem je především důsledný postup, komplexní posouzení kritérií a specifikace, že je metoda prováděna jedincem.

4.3.3 Metoda výběru kompromisní varianty

Varianta, která bude reprezentovat nejvhodnější game engine se určí metodou váženého součtu. Tato metoda se jeví jako adekvátní, protože je cílem bakalářské práce nalézt nejvýhodnější variantu. Výsledek určí pořadí variant a čtenář má tak k dispozici podrobnější přehled .

4.4 Postup rozhodování

Tabulka 3 Údaje game enginů v kriteriální tabulce

	Práce v enginu	Komunita	Výchozí skriptovací jazyk	Implementace souborů z jiných programů	Rozsah funkcí
CryEngine	52,5	1,6	65	5	8
Godot	72,5	60,8	72,5	8	9
Unity	82,5	240	65	7	6
Unreal Engine	57,5	113	50	6	10
povaha	max	max	max	max	max
váhy					

Zdroj: autor

4.4.1 Hodnocení důležitosti kritérií

Je zadáno 5 kritérií, u kterých se volí priority pomocí devítibodové stupnice.

Nejprve se navrhne tabulka s jedničkami na hlavní diagonále.

Tabulka 4 Návrh Saatyho matice

	K1	K2	K3	K4	K5
K1	1				
K2		1			3
K3			1		
K4				1	
K5		1/3			1

Zdroj: autor

Do tabulky se následně doplňují hodnoty kritérií podle priorit. Například K2 je slabě preferováno před K4 znamená, že se do pole K2K5 doplní číslo 3 a do pole opačného, tedy K5K2 se doplní převrácená hodnota ⅓.

Autor se rozhodl především preferovat objektivní kritéria před subjektivními. Proto jsou priority slovně vyjádřeny takto:

K1 je rovnocennou K3 a slabě preferováno před K4.

K2 je slabě preferováno před K5, velmi silně preferováno před K4 a silně preferováno před K1 a K3.

K3 je slabě preferováno před K4.

K5 je silně preferováno před K4 a slabě preferováno před K1 a K3.

Výsledná Saatyho tabulka vypadá tedy takto.

Tabulka 5 Saatyho matice

	K1	K2	K3	K4	K5
K1	1	1/5	1	3	1/3
K2	5	1	5	7	3
K3	1	1/5	1	3	1/3
K4	1/3	1/7	1/3	1	1/5
K5	3	1/3	3	5	1

Zdroj: autor

Pro ověření správnosti zvolených preferencí kritérií je nutné provést výpočet míry konzistence. Po dosazení největšího kořene polynomu 5,12689 do následujícího vzorce:

$$I = \frac{Imax - n}{n - 1}$$

vychází míra konzistence 0,0317225.

Míra konzistence splňuje podmínku, je tedy menší než 0,1 a lze postupovat v rozhodování.

Dále se přidá do tabulky sloupec součin, geometrický průměr řádků a váhy.

Tabulka 6 Výsledná Saatyho matice

	K1	K2	K3	K4	K5	Součin	Geometrický průměr řádků	Váhy
K1	1	1/5	1	3	1/3	0,20	0,73	0,10
K2	5	1	5	7	3	525,00	3,50	0,50
K3	1	1/5	1	3	1/3	0,20	0,73	0,10
K4	1/3	1/7	1/3	1	1/5	0,003	0,32	0,05
K5	3	1/3	3	5	1	15,00	1,72	0,25

Zdroj: autor

Po kontrole součtu vah kritérií, který se musí rovnat jedné, lze v úloze pokračovat pomocí metody výběru kompromisních variant.

Tabulka 7 Doplnění vah do kriteriální matice

	Práce v enginu	Komunita	Výchozí skriptovací jazyk	Implementace souborů z jiných programů	Rozsah funkcí
CryEngine	52,5	1,6	65	5	8
Godot	72,5	60,8	72,5	8	9
Unity	82,5	240	65	7	6
Unreal Engine	57,5	113	50	6	10
povaha	max	max	max	max	max
váhy	0,10	0,50	0,10	0,05	0,25

Zdroj: autor

Podle hodnot v tabulce se stanovují ideální (I) a bazální (B) varianty.

I = (82,5; 240; 72,5; 7; 10)

B = (52,5; 1,6; 50; 5; 6)

Jelikož jsou všechna kritéria maximální povahy, jsou v ideální variantě maximální hodnoty a v bazální minimální hodnoty. Tabulka se převede na standardizovanou a provede se skalární součin řádků variant a vah.

- Pro CryEngine vychází hodnota 0,182338507
- Pro Godot vychází hodnota 0,527248464
- Pro Unity vychází hodnota 0,709444076
- Pro Unreal Engine vychází hodnota 0,513180026
- Hodnoty jsou v tabulce zaokrouhleny na poslední 2 desetinná čísla.

Tabulka 8 Standardizovaná matice

	Práce v enginu	Komunita	Výchozí skriptovací jazyk	Implementace souborů z jiných programů	Rozsah funkcí	Výsledná hodnota (skalární součin)
CryEngine	0	0	0,67	0	0,50	0,19
Godot	0,67	0,25	1,00	1,00	0,75	0,53
Unity	1,00	1,00	0,67	0,67	0	0,70
Unreal Engine	0,17	0,47	0	0,33	1,00	0,51

Zdroj: autor

4.5 Výsledek rozhodování metodou váženého součtu

Tabulka 9 Výsledná tabulka

	Užitek	Pořadí
CryEngine	0,19	4
Godot	0,53	2
Unity	0,70	1
Unreal Engine	0,51	3

Zdroj: autor

Nejlepší varianta je určena nejvyšším užitkem.

4.6 Výsledek vícekriteriální analýzy variant

Pomocí Saatyho metody stanovení vah kritérií a metodou váženého součtu pro výběr kompromisní varianty, byl program Unity zvolen jako nejvýhodnější game engine pro začátečníky. Na druhém místě se usadil program Unreal Engine, na třetím Godot a na posledním software CryEngine.

Unity je nejvhodnější variantou především díky své ohromné komunitě, přívětivému skriptovacímu jazyku a návrhu UI.

5 Závěr

Cílem bakalářské práce bylo informovat čtenáře o game enginech Unity, Unreal Engine, CryEngine a Godot a z uvedených vývojových prostředí doporučit začínajícím vývojářům nejvhodnější variantu. Teoretická část práce se věnovala důležitým prvkům obsažených ve vývoji her, jako jsou specifikace game enginů, skriptovací jazyky, artificial inteligence a profese, ze kterých jsou obvykle sestaveny vývojářské týmy. Dále jsou v teoretické části uvedeny metody vícekriteriálního rozhodování a User Experience, které jsou uplatněny při hodnocení stanovených vývojových prostředí.

Pro dosažení zvolených cílů byly vytvořeny čtyři hratelné prototypy závodní hry v jednotlivých game enginech. Podle těchto prototypů jsou vývojová prostředí porovnána z hledisek: práce v enginu, komunita, výchozí skriptovací jazyk, implementace souborů z jiných programů a rozsah funkcí. Hodnoty v kritériích práce v enginu a výchozí skriptovací jazyk jsou získány z výsledků dotazníku Škála použitelnosti systému. Výchozím krokem vícekriteriální analýzy variant bylo stanovení vah neboli preferencí jednotlivých kritérií Saatyho metodou. Kritéria komunita a rozsah funkcí byla preferována nad ostatními, jelikož se v nich vyskytují objektivní hodnoty game enginů. Podmínka míry konzistence Saatyho matice byla splněna a následně se podle vah uplatnila metoda váženého součtu. Výsledkem analýzy je seřazení game enginů v následujícím pořadí: Unity, Godot, Unreal Engine a CryEngine.

Cílů bakalářské práce bylo tedy dosaženo. Vývojové prostředí Unity se prokázalo jako jednoznačná volba pro začínající vývojáře především díky téměř neomezenému množství tutoriálů a návodů, které jsou volně dostupné na internetu, efektivnímu skriptovacímu jazyku a uživatelsky přívětivému návrhu UI. Ačkoliv Unity obdrželo nejhorší ohodnocení v kritériu rozsah funkcí, více než dvojnásobná hodnota(240 tisíc) při porovnání s Unreal Enginem(113 tisíc) z hlediska komunit je hlavní důvod, proč je Unity na prvním místě. Hodnocení alternativ Godot a Unreal Engine bylo relativně vyrovnané. Godot, ačkoliv nemá tak velkou komunitu jako Unity či Unreal Engine, se umístil na druhém místě zásluhou za svůj skriptovací jazyk GDScript, pro který je ve vývojovém prostředí přímo zabudovaný IDE. Jednoduchá syntaxe jazyka a možnost konfigurovat prototyp za běhu aplikace jej činí vhodnou variantou pro uživatele, kterým vývojové prostředí Unity nevyhovuje. Unreal Engine obdržel nejméně bodů v kritériu výchozí skriptovací jazyk, jelikož se jedná o jazyk C++, který je pro začátečníky náročný a autor s ním měl nejvíce

potíží. Vývojové prostředí CryEngine obsahuje nejnižší hodnotu v kritériu komunita (pouhé 1,8 tisíc). Začínajícím vývojářům by proto dělalo nejvíce potíží obstarat si specifický materiál nebo řešení pro jejich problémy. Toto vývojové prostředí se tedy začátečníkům nedoporučuje.

6 Seznam použitých zdrojů

ALEJANDRO BORROMEO, Nicolas, 2020. *Hands-On Unity 2020 Game Development* [online]. Birmingham: Packt Publishing. ISBN 978-1-83864-200-6.

BUCHALCEVOVÁ, Alena, 2005. Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky. Praha: Grada. Management v informační společnosti. ISBN 80-247-1075-7.

BRADFIELD, Chris, 2018. *Godot Engine Game Development Projects*. Birmingham: Packt Publishing. ISBN 978-1-78883-150-5.

HEJLSBERG, Anders, Mads TORGERSEN, Scott WILTAMUTH a Peter GOLDE, 2010. *C# Programming Language (Covering C# 4.0)*. Boston: Addison-Wesley Professional. ISBN 978-0-321-74176-9.

H. CHAUDHARY, Harry, 2014. *C# Programming: The ultimate way to learn the fundamentals of the C# language*. Createspace O-D Publishing. ISBN 978-1500193515.

JAISWAL, Sonoo, 2018. Introduction of GDScript. *Java T point* [online]. Ghaziabad: Sonoo Jaiswal [cit. 2021-03-06]. Dostupné z: https://www.javatpoint.com/godot-introduction-of-gdscript

LINIETSKY, Juan a Ariel MANZUR, 2020. From Unity to Godot. *Godot Engine* [online]. [cit. 2021-03-02]. Dostupné z: https://docs.godotengine.org/en/stable/getting_started/editor/unity_to_godot.html?

LINIETSKY, Juan a Ariel MANZUR, 2020. What is Visual Scripting. *Docs.godotengine* [online]. Argentina: Linietsky [cit. 2021-03-06]. Dostupné z: https://docs.godotengine.org/en/stable/getting_started/scripting/visual_script/what_is_visual_scripting.html

M. BOURG, David a Glenn SEEMANN, 2004. *AI for Game Developers*. Sebastopol (Kalifornie): O'Reilly Media. ISBN 9780596005559.

STROUSTRUP, Bjarne, 2013. *C++ Programming Language*. 4th edition. Amsterdam: Addison-Wesley. ISBN 978-0321563842.

ŠUBRT, Tomáš, 2011. *Ekonomicko-matematické metody*. Plzeň: Vydavatelství a nakladatelství Aleš Čeněk. ISBN 978-80-7380-345-2.

T. WILL, 2020. Measuring and Interpreting System Usability Scale (SUS). *UIUXTrend* [online]. Singapore: Epitomist [cit. 2021-03-04]. Dostupné z: https://uiuxtrend.com/measuring-system-usability-scale-sus/

SANDERS, Andrew, 2016. *An Introduction to Unreal Engine 4*. Boca Raton (Florida): A K Peters/CRC Press. ISBN 978-1498765091.

TOKAREV, Kirill a Arti SERGEEV, 2016. *Godot 2.0: Talking with the Creator* [online]. 1 [cit. 2021-03-02]. Dostupné z: https://80.lv/articles/godot2-interview/GREGORY, Jason, 2018. *Game Engine Architecture*. 3rd ed. New York: Taylor & Francis. ISBN 9781138035454.

TRIANTAPHYLLOU, Evangelos, 2010. *Multi-Criteria Decision Making Methods: A comparative Study* [online]. Baton Rouge: Springer. ISBN 978-1441948380.

U.S. Dept. of Health and Human Services, 2006. The Research-Based Web Design & Usability Guidelines, Enlarged/Expanded edition. Washington: U.S. Government Printing Office [online]. [cit. 2021-03-10]. Dostupné z: https://www.usability.gov/what-and-why/user-experience.html

U.S. Dept. of Health and Human Services, 2006. The Research-Based Web Design & Usability Guidelines, Enlarged/Expanded edition. Washington: U.S. Government Printing Office [online]. [cit. 2021-03-10]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

YERLI, Faruk, 2002. Crytek announces its Game Engine CryENGINE. *Crytek* [online]. [cit. 2021-03-05]. Dostupné z: https://web.archive.org/web/20081115062536/http://www.crytek.com/news/news/?tx_ttnews%5Bpointer%5D=17&tx_ttnews%5Btt_news%5D=76&tx_ttnews%5BbackPid%5D=9&cHash=e4dea47a80

7 Přílohy