Česká zemědělská univerzita v Praze

Technická fakulta

Katedra technologických zařízení staveb



Bakalářská práce

Návrh univerzální programové logiky pro vývoj her

Martin Novák

2022 ČZU v Praze





**Čestné prohlášení**

Prohlašuji, že svou bakalářskou práci "Návrh univerzální programové logiky pro vývoj her" jsem vypracoval(a) 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(ka) uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne datum odevzdání \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Poděkování**

Rád(a) bych touto cestou poděkoval(a) jméno vedoucího, případně dalších osob, a informace, za co děkujete.

Návrh univerzální programové logiky pro vývoj her

**Abstrakt**

Souhrn práce (cca 15 řádek textu).

**Klíčová slova:** herní enginy, uživatelská rozhraní, programovací jazyky, objektový model, vývoj herDesign of universal program logic for game development

**Abstract**

Anglický překlad českého souhrnu

**Keywords**: game engines, user interface, programing languages, object model, game development

**Obsah**

[1. Úvod 1](#_Toc101350911)

[2. Cíl práce a metodika 1](#_Toc101350912)

[2.1 Cíl práce 1](#_Toc101350913)

[2.2 Metodika 1](#_Toc101350914)

[3. Výběr vhodných programovacích jazyků pro vývoj her 1](#_Toc101350915)

[3.1 C++ 2](#_Toc101350916)

[3.2 Java 2](#_Toc101350917)

[3.3 C# 2](#_Toc101350918)

[3.4 výběr 2](#_Toc101350919)

[4. Výběr herních žánrů vhodných pro implementaci 2](#_Toc101350920)

[4.1 RPG 3](#_Toc101350921)

[4.2 akční 3](#_Toc101350922)

[4.3 strategie 3](#_Toc101350923)

[4.4 závodní 3](#_Toc101350924)

[5. Grafické výstupy aplikací 3](#_Toc101350925)

[5.1 konzolová aplikace 3](#_Toc101350926)

[5.2 okenní aplikace 3](#_Toc101350927)

[5.2.1 WinForm 3](#_Toc101350928)

[5.2.2 WPF 3](#_Toc101350929)

[6. Návrh aplikačního modelu 3](#_Toc101350930)

[Te 3](#_Toc101350931)

[7. Návrh vzorového řešení 3](#_Toc101350932)

[8. Zhodnocení realizace aplikace 4](#_Toc101350933)

[9. Závěr 4](#_Toc101350934)

[Seznam použitých zdrojů 5](#_Toc101350935)

[Přílohy 6](#_Toc101350936)

**Seznam obrázků**

Odkazovaný seznam obrázků

**Seznam tabulek**

Odkazovaný seznam tabulek

**Seznam použitých zkratek**

Soupis a definování zkratek (vyskytuje-li se jich v textu velké množství)

# 1. Úvod

Text text text text text text text text text text text text text text text text text text text text text text text.

# 2. Cíl práce a metodika

## 2.1 Cíl práce

Cílem práce je popsat aktuální dostupné herní enginy, uživatelská rozhraní a programovací jazyky vhodné pro návrh vzorového řešení. Nejprve na návrhu aplikačního modelu popsat objektový model aplikace. Následně vytvořit návrh vzorového řešení, které bude univerzální v oblasti vývoje her typu RPG.

## 2.2 Metodika

Text text text

# 3. Výběr vhodných programovacích jazyků pro vývoj her

Programovací jazyky dělíme na 2 základní skupiny. První jsou imperativní (např. C++), kam patří většina jazyků a jejich rysem je, že kód je sekvence instrukcí a je z něj čitelné co se v jaký okamžik bude provádět. Druhá skupina jsou deklarativní (např. HTML), které říkají jen co se musí vyřešit, ale ne konkrétní instrukce potřebné k provedení a z toho důvodu často nejsou považovány za programovací jazyky, ale používá se pro ně označení kódovací. Další skupina jsou funkcionální (např. Haskell), které ačkoliv se řadí mezi deklarativní mají znaky obou skupin a je možné jejich přístup použít i v imperativních jazycích. Na rozdíl od imperativních nevyužívají žádné globální proměnné a vše je prováděno uvnitř funkcí. Na Obr. 1 je porovnání sumy zapsané pomocí imperativního a funkcionálního jazyku.[1, 2]



Obr. imperativní vs. funkcionální jazyk [2]

Z popisu základních paradigmat je vidět, že jazyk bude vybírán z imperativních jazyků, které se dále dělí na 2 podskupiny. Procedurální(*Procedural and Object Oriented Programming*, n.d.) (např. C) pracují s funkcemi přijímajícími data pouze z parametrů nebo globálních proměnných. Pro svázání více souvisejících hodnot je možné použít strukturu, která je jako pole umožňující ukládat různé datové typy. Objektové (např. Java) mají třídy sloužící jako předlohy pro instance nazývané objekty, které stejně jako struktury mohou ukládat více hodnot různých typů, ale mají vlastní metody, a proto není potřeba všechna data předávat pomocí parametrů, protože si je může načíst z objektu kde se nachází. Objektově orientované programování (OOP) má 4 základní principy: zapouzdření, abstrakce, dědičnost a polymorfismus. Zapouzdření umožňuje omezit viditelnost proměnných a metod mimo třídu, kontrolovat přístup k jejich hodnotám a ověřit, zda je zapisována platná hodnota. Abstrakce znamená, že pro práci s objektem není nutné znát vnitřní funkci jeho metod a při práci v týmu kolegovi stačí znát název, parametry a výstup metody. Použitím dědičnosti třída, která je potomek získá všechny proměnné a metody rodiče, ale je možné přidat nové, či změnit chování metody. Polymorfismus souvisí s dědičností, kde do proměnné typu rodič je možné vložit potomka, ale při volání metody se zavolá její přetížená verze, která má stejné jméno, typ a parametry, ale jiné tělo. Dále je možné přetěžovat metody změnou parametrů nebo návratové typu.(FreeCodeCamp.org, 2020) Na Obr. 2 je porovnání počítání obsahu čtverců a obdélníků napsané v procedurálním a objektovém jazyce (kvůli délce vynecháno zadávání hodnot). Je evidentní, že pro hry se nejvíce hodí objektové jazyky, a proto ty nejpoužívanější nyní budou probrány více do hloubky.



Obr. procedurální vs objektový jazyk-vlastní

## 3.1 C++

Text…

## 3.2 Java

Text…

## 3.3 C#

Text…

## 3.4 výběr

Text…

# 4. Výběr herních žánrů vhodných pro implementaci

Text

## 4.1 RPG

Te

## 4.2 akční

te

## 4.3 strategie

te

## 4.4 závodní

te

5. Grafické výstupy aplikací

Te

## 5.1 konzolová aplikace

Te

## 5.2 okenní aplikace

Te

### 5.2.1 WinForm

Te

### 5.2.2 WPF

Te

6. Návrh aplikačního modelu

Te

7. Návrh vzorového řešení

Te

8. Zhodnocení realizace aplikace

Te

# 9. Závěr

Text…

# Seznam použitých zdrojů

[1] imperative programming [online]. nedatováno [vid. 2021-03-16]. Dostupné z: https://whatis.techtarget.com/definition/imperative-programming

[2] COMPUTERPHILE. *Programming Paradigms - Computerphile* [online]. 2013 [vid. 2021-03-29]. Dostupné z: https://www.youtube.com/watch?v=sqV3pL5x8PI

# Přílohy

Odkazovaný seznam příloh