Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Počítačová hra založená na principech algoritmizace

2022

Petr Vykoukal

Vedoucí práce: Mgr. Tomáš Kühr, Ph.D.

Studijní obor: Aplikovaná informatika, kombinovaná forma

Bibliografické údaje

Autor:	Petr Vykoukal
Název práce:	Počítačová hra založená na principech algoritmizace
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2022
Studijní obor:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	Mgr. Tomáš Kühr, Ph.D.
Počet stran:	37
Přílohy:	1 DVD
Jazyk práce:	český

Bibliograhic info

Author:	Petr Vykoukal
Title:	Computer Game Based on Algorithm Design Principles
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2022
Study field:	Applied Computer Science, combined form
Supervisor:	Mgr. Tomáš Kühr, Ph.D.
Page count:	37
Supplements:	1 DVD
Thesis language:	Czech

Anotace

Tato bakalářská práce popisuje implementaci počítačové hry v prostředí Unity a slouží i jako její dokumentace. Jedná se o hru založenou na principech algoritmizace. Hráč v grafické podobě vytváří krátké programy, podle kterých se následně vozidlo pohybuje po mapě a plní zadané úkoly. Cílem hry je přiblížit hráči tvorbu jednoduchých algoritmů.

Synopsis

This bachelor thesis describes the implemetation of a computer game in the Unity engine and also serves as its documentation. This is a game based on the algorithmization principles. The player creates short programs in graphical form, according to which the vehicle then moves around the map and performs the assigned tasks. The aim of the game is to introduce the player to the creation of simple algorithms.

Klíčová slova: Unity, Počítačová hra, Algoritmus, Programování

Keywords: Unity, Computer game, Algorithm, Programming

Děkuji svému vedoucímu práce Mgr. Tomáši Kührovi, Ph.D. za jeho podnětné rady a pomoc při řešení problémů. Dále děkuji všem svým blízkým za jejich podporu během mého studia. Zejména pak svým dcerám.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

V této práci pojednávám o implementaci počítačové hry, která hráče seznámí s tvorbou jednoduchých algoritmů. Výsledná hra je tak využitelná právě při výuce základů algoritmizace a programování. V tomto textu představím některá další existující řešení. Dále čtenáře seznámím s programátorskou dokumentací, kde popíšu prostřední editoru Unity a detaily výsledné aplikace. V poslední části pak představím uživatelskou dokumentaci, kde popíšu jednotlivé části výsledné aplikace.

1.1 Požadavky na výslednou aplikaci

Cílem práce bylo vytvořit počítačovou hru, která bude splňovat tyto požadavky:

- přívětivé uživatelské prostředí
- vytváření programu pro vozítko skládáním příkazů, které by měly mít ideálně grafickou podobu
- součástí programu bude volání funkcí, smyčka typu opakuj n-krát a podmínka
- bude obsahovat sadu úkolů se stupňující se obtížností

1.2 Průzkum existujících řešení

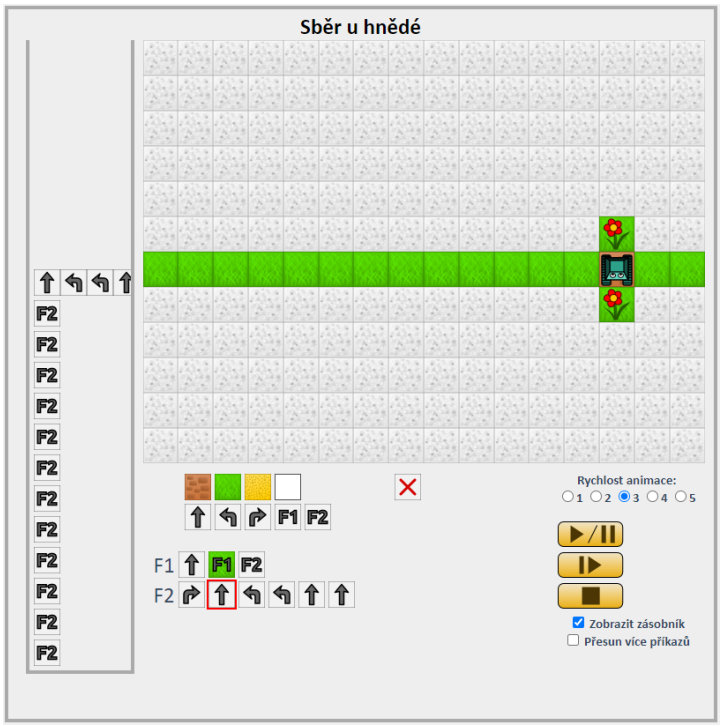
V průběhu své práce jsem nalezl několik her, které mají stejné zaměření. Některé z nich jsem vyzkoušel.

1.2.1 Robotanik

Hra je součástí webové stránky zaměřené na výuku informatiky „Umíme informatiku“[1]. Stránka je součástí projektu „Umíme to“[2], který se věnuje podpoře výuky více předmětů. Všechny hry v rámci projektu „Umíme to“ jsou vytvořeny jako webové aplikace.

Ve hře Robotanik hráč programuje robota, který následně prochází mapou a sbírá květiny. Program hráč vytváří v grafické podobě. K dispozici jsou příkazy pro pohyb, nebo volání funkcí. Každý tento příkaz je možné podmínit materiálem podlahy. Příkaz se tak vykoná, jen když robot v tu chvíli stojí na konkrétním typu podlahy. V pokročilejších úrovních je hra zaměřená na rekurzi. Ukázka ze hry je pak k vidění na obrázku č. 1. V levé části obrázku je vidět zásobník rekurzivního volání.

Hra zdarma je limitovaná počtem pokusů za 1 den[3]. Pro neomezené hraní je třeba mít zaplacenou časově omezenou licenci pro domácnost, nebo multilicenci pro školu.



Obrázek 1: Ukázka ze hry Robotanik

1.2.2 Code Combat

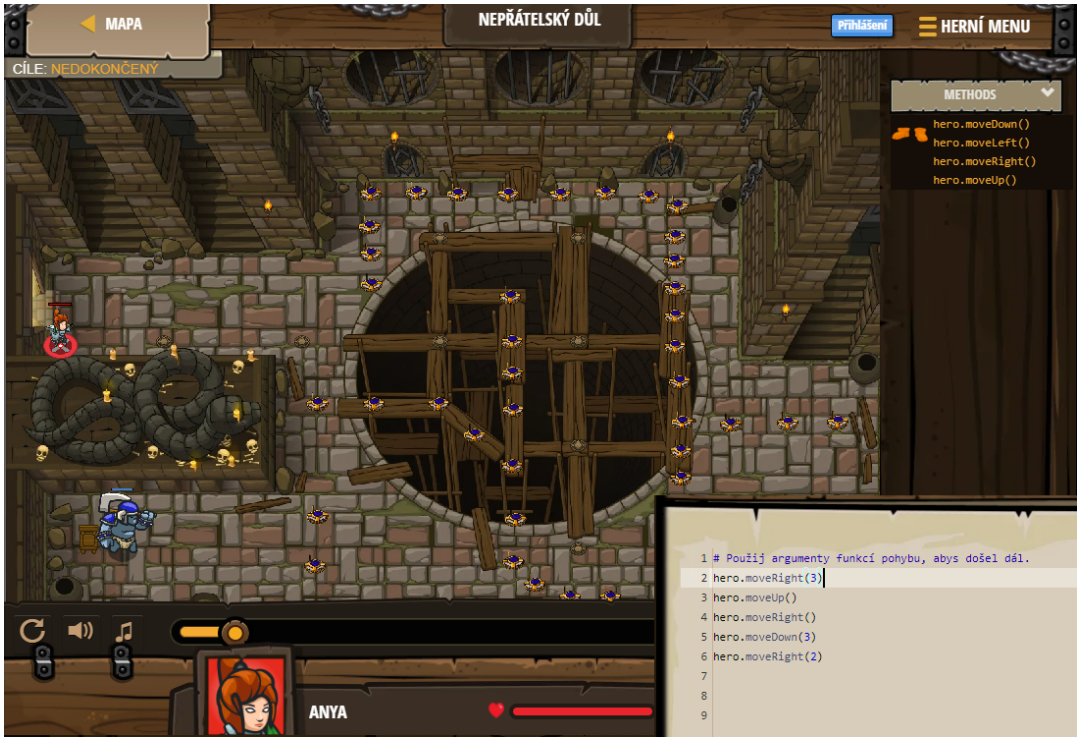
Hra Code Combat[4] je webová aplikace a slouží k výuce jazyků Python a JavaScript. V nastavení jsou ale dostupné i další jazyky. CoffeeScript, Lua, C++ a Java. Ty ale nemusí fungovat stoprocentně a jsou označeny jako „pokusné“.

V této hře musí hráč vytvořit kód v textové podobě, který následně vykonává postava ve hře, a tím plní úkoly jednotlivých úrovní. Oblast pokrytí tématu je zde poměrně rozsáhlá. Od jednoduchých instrukcí typu „Jdi rovně“, přes cykly a podmínky až po práci s řetězci a poli. V rámci této práce jsem ale vyzkoušel jen prvních 5 úrovní. Ukázka ze hry je k vidění na obrázku č. 2.

Zdarma je možné vyzkoušet jen prvních 5 úrovní. Pro další hraní je třeba mít měsíční, nebo roční předplatné.

1.2.3 Light Bot

Hra Light Bot[5] je dostupná jako aplikace pro mobilní telefony a tablety s operačním systémem Android, nebo iOS. Ukázka ze hry je zobrazena na obrázku č. 3. V této hře se hráč ujme robota, pro kterého vytvoří program v grafické podobě. Robot se následně pohybuje po mapě a na určitých místech musí vykonat instrukci „rozsvícení“. K dispozici jsou instrukce pro pohyb do čtyř směrů, skok, volání procesu a zmíněné rozsvícení.



Obrázek 2: Ukázka ze hry Codecombat

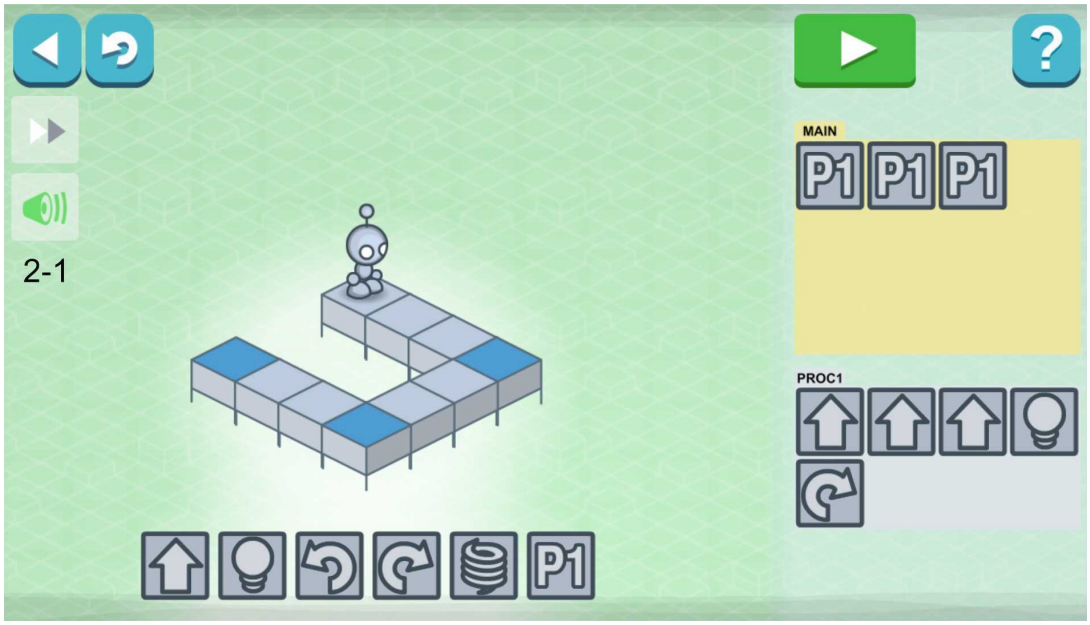
Aplikace je dostupná ke stažení v obchodech jednotlivých platforem za jednorázový poplatek pod názvem „Lightbot: Programming Puzzles“, nebo zdarma k vyzkoušení několika úrovní pod názvem „Lightbot: Code Hour“.

1.3 Návrh řešení

Rozhodl jsem se, jít cestou desktopové aplikace, namísto webové. Zejména z toho důvodu, že pro spuštění aplikace pak není nutné připojení k internetu, případně webový server v lokální síti. K vytvoření jsem se rozhodl použít herní engine. Herní engine poskytuje vývojářům abstrakční vrstvu nad grafickým Application programming interface (API) daného operačního systému. Poskytuje dále vývojářům služby pro práci s fyzikou, detekci kolizí a mnoho dalších funkcí, které se běžně ve hrách používají. Vybral jsem si engine Unity[6], vyvíjený společností Unity Technologies. Aktuální verze v době zahájení mé práce byla 2020.3.12f1.

Unity mi umožnil importovat 3D modely vytvořené v aplikaci Blender[7] a následně mi, díky výše uvedené abstrakční vrstvě, usnadnil práci při jejich vykreslování a animování. Prvky Graphical User Interface (GUI) jsem pak vytvářel ve vektorovém grafickém editoru Affinity Designer[8]. Pro import prvků do Unity ale bylo nutné uložit tyto prvky jako rastrové. Samotné herní funkce jsem následně vytvářel v jazyce C#, který je v současnosti v Unity nativně podporován[1].

[1]Podporovány jsou i další jazyky .NET, které dokáží zkompilovat kompatibilní DLL[9]



Obrázek 3: Ukázka ze hry Lightbot: Code Hour

Samotné zadávání programu jsem se rozhodl zpracovat v grafické podobě. Hráč se tak oprostí od nutnosti pamatovat si syntaxi dalšího jazyka a výuka prostřednictvím této aplikace tak může být soustředěna právě na rozvoj algoritmického myšlení. Od tohoto principu, kdy hráč skládá svůj program z bloků, jsem odvodil i název aplikace, tedy Block Code. Zkráceně pak jen B-Code.

2 Programátorská dokumentace

V této kapitole představím čtenáři vývojové prostředí Unity, podmínky a nároky jeho užití. Dále čtenáře seznámím s detaily aplikace vytvořené v rámci mé bakalářské práce.

2.1 Podmínky užití Unity

Unity je možné za použití konkrétních podmínek[10] používat zdarma. A to pro tvorbu nekomerčních aplikací. V případě komerčních aplikací je možné používat Unity zdarma pouze pokud subjekt, který Unity používá, nevykazuje roční zisk vyšší než 100 000 USD. V opačném případě je již nutné zaplatit licenční poplatky[11], které jsou v době psaní tohoto textu 1 800 USD ročně za jednoho vývojáře.

Pro potřeby této práce jsem tedy využil licenci zdarma, kde platí jediné omezení. A tím je zobrazení loga Unity při spuštění aplikace.

2.2 Systémové požadavky

Minimální systémové požadavky editoru Unity[12] pro operační systém Windows jsou uvedeny v tabulce č. 1. Vynechal jsem operační systém Windows 7 z důvodu ukončení jeho podpory ze strany společnosti Microsoft[13]. V tabulce je dále zmíněna instrukční sada Streaming SIMD Extension ver. 2 (SSE2)[2]. Toto rozšíření zavedla společnost Intel již v roce 2000. Tuto instrukční sadu tak v dnešní době podporují všechny moderní procesory.

Dále v tabulce č. 2 uvádím systémové požadavky pro desktop aplikaci sestavenou pro operační systém Windows. Opět jsem vynechal operační systém Windows 7.

Tabulka 1: Systémové požadavky editoru Unity

Operační systém	64 bitová verze Windows 10, nebo Windows 11
CPU	Procesor architektury x64 s podporou instrukční sady SSE2
Grafické API	Grafická karta podporující DirectX 10, DirectX 11, nebo DirectX 12

2.3 Popis editoru Unity

Na obrázku č. 4 je zobrazen náhled editoru Unity. Umístění podoken v okně aplikace je volně nastavitelné. V případě práce na více monitorech je možné podokna

[2]Instrukční sada umožňující jednou instrukcí zpracovat data ve více registrech současně.

Tabulka 2: Systémové požadavky pro spuštění výsledné aplikace

Operační systém	32 bitová, nebo 64 bitová verze Windows 10, nebo Windows 11
CPU	Procesor architektury x86, nebo x64 s podporou instrukční sady SSE2
Grafické API	Grafická karta podporující DirectX 10, DirectX 11, nebo DirectX 12

oddělit od hlavního okna a rozmístit na jinou pracovní plochu. V případě mého nastavení se v levé horní části nachází podokno „Hiearchy“. To obsahuje strom objektů typu `GameObject`[3] načtené scény.

V horní části uprostřed je 3D náhled scény, kde je možné pracovat s objekty vykreslenými přímo ve scéně. Uživateli tak pomáhá s orientací a představou, jak bude výsledná scéna vypadat. V horní části je ještě záložka „Game“, která umožňuje vidět scénu přímo z pohledu kamery. Tedy přesně tak, jak bude vypadat ve spuštěné aplikaci. Včetně interakcí objektů na pohyb myši, nebo stisk kláves.

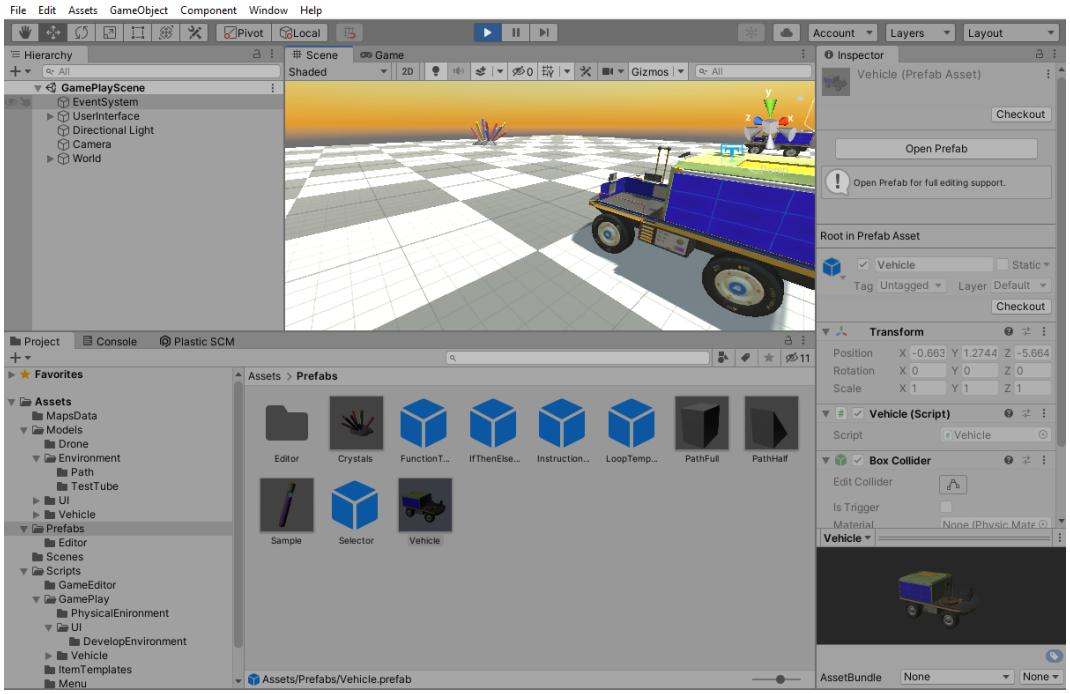
V pravé části je podokno s názvem „Inspector“, kde je možné provádět obecná nastavení vlastností objektu vybraného v okně „Hierarchy“. Každý objekt zde má kartézské souřadnice umístění ve 3D, nebo 2D prostoru. Dále je zde možné k objektům přiřadit komponenty[4], které pak definují chování a vlastnosti daného objektu. Unity obsahuje spoustu již hotových komponent. Taková komponenta může například definovat, že daný objekt bude obrázek, nebo tlačítko, které reaguje na interakci myši, případně kamera atd... Dále se zde jako komponenty přiřazují i uživatelsky vytvořené scripty.

Nakonec ve spodní části se nachází podokno „Project“, kde je v jeho levé části adresářová struktura projektu a v jeho pravé části obsah vybraného adresáře. Kořenovým adresářem je zde adresář `Assets`. V adresářovém stromě se mohou kdekoliv nacházet i speciální adresáře[14], které mají zvláštní význam. Ve svém projektu jsem využil jen adresář `Resources`. Objekty, které jsou uloženy v tomto adresáři, mohou být nalezeny a použity pomocí kódu ve scriptech. V mém případě takto načítám například obrázky ikon, nebo předem vytvořených objektů a přidávám je do seznamu, jak ukazuje zdrojový kód č. 1. Jedná se o výtažek ze souboru `ResourcesManager.cs`. V opačném případě musí být objekty vloženy do scény již v editoru, ještě před sestavením aplikace.

Editor dále umožňuje vytvářet animace 2D i 3D objektů. Mezi těmito animacemi se následně definují přechody. Animace spouští komponenta `Animator` a díky ní je možné spouštět animace přímo ze scriptů. Komponenta `Animator`

[3]Typ `GameObject` může být jakýkoliv objekt ve scéně. Tedy například kamera, 3D objekty ve scéně, nebo plátno a jednotlivé prvky grafického uživatelského rozhraní – GUI

[4]Komponenty jsou C# scripty přiřazené k objektům `GameObject`



Obrázek 4: Náhled editoru Unity

```
1 private static void LoadIcons()
2 {
3     icons.Add( Resources.Load<Sprite>( "UI\\Icons\\IcoWarning" ) );
4     icons.Add( Resources.Load<Sprite>( "UI\\Icons\\IcoFWD" ) );
5 }
```

Zdrojový kód 1: Načtení zdrojů z adresáře `Resources`

poskytuje i zpětnou vazbu v závislosti na průběhu animace, kdy může být vyvolána událost v závislosti na konkrétním klíčovém snímku animace.

2.4 Tvorba apliakce

Jak jsem zmínil v kapitole 2.3, tak pro vytvoření nějakého objektu ve hře je nutné tento objekt zařadit do stromové struktury podokna „Hierarchy“. Stačí kliknout pravým tlačítkem v podokně „Hierarchy“ a v kontextovém menu zvolit požadovaný objekt. Je možné vytvářet rovnou nějaké předdefinované 2D, nebo 3D objekty, dále objekty pro GUI, které vyžadují, aby byly vloženy do objektu třídy `Canvas` atd... Další možností, jak vytvořit objekt, je přetáhnout jej myší z podokna „Project“.

V podokně „Project“, kde se nachází adresářová struktura projektu, totiž mohou být soubory, ze kterých Unity automaticky vytvoří objekt a založí jej

do „Hierarchy“. Mohou to být obrázky, importované 3D modely, nebo tzv. „Prefab Asset“. To jsou předem připravené objekty, které se mohou skládat z více vnořených objektů, a které již mohou mít přiřazeny nějaké komponenty, nastaveny nějaké parametry, atd... Z takového objektu je pak možné jednoduše vytvářet kopie (*instance*), kdy při změně originálního souboru „Prefab Asset“ dojde ke změně i u všech jeho kopií.

V podokně „Project“ se pak nachází také uživatelsky definované C# scripty. Vytvářet se dají kliknutím pravým tlačítkem ve zvoleném adresáři a v kontextové nabídce výběrem volby `Create > C# Script`. Pro editaci scriptů je v Unity vestavěná podpora pro externí editory[15] Visual Studio, Visual Studio Code a JetBrains Rider. To znamená, že při použití editoru s vestavěnou podporou, Unity automaticky předává správné argumenty externímu editoru a spolupráce Unity s editorem je tak provázanější. Je ale možné použít i jiný editor.

Vytvořený script musí mít unikátní název a definovat třídu shodnou se svým názvem. Na rozdíl od běžné .NET aplikace, zde není definice jmenného prostoru. Pokud má třída být komponentou nějakého objektu, pak musí být potomkem třídy `MonoBehaviour`. Taková třída nesmí mít vlastní konstruktor. Unity automaticky volá za určitých okolností některé metody[16], které je možné v této třídě implementovat, jak ukazuje ukázkový zdrojový kód 2. V projektu používám následující[5]:

Awake Je volána jednorázově při inicializaci třídy.

Start Je volána jednorázově při inicializaci třídy. Je ale volána později, než `Awake`. Nelze definovat v jakém pořadí se tato metoda volá u jednotlivých tříd a tak může dojít ke kolizi, kdy jedna třída odkazuje na vlastnosti jiné třídy, které ale zatím nebyly inicializovány. V takovém případě se u druhé třídy použije metoda `Awake`, aby se inicializovala dřív.

Update Je volána opakovaně po zaznamenání vstupních událostí před provedením herní logiky (*vykonáním scriptů*) a před vykreslením snímku scény.

LateUpdate Je volána opakovaně po provedení herní logiky a před vykreslením snímku scény.

2.5 Vlastní aplikace

V této kapitole vysvětlím konkrétní detaily mého programového řešení. Nejdříve popíšu adresářovou strukturu projektu a strukturu souboru Extensible Markup Language (XML), který využívám k ukládání map jednotlivých úrovní. Následně popíšu tvorbu 2D grafických prvků pro GUI a 3D grafických objektů a popíšu hlavní třídy v mém projektu, kde se zaměřím na vytváření a vykonávání programu hráčem. Nakonec popíšu nezbytné kroky pro rozšíření hry o další obsah –

[5] Pro jednoduchost uvádím jen názvy. Všechny jsou typu `void` a všechny jsou bez parametrů.

```
1  using UnityEngine;
2
3  public class SampleClass : MonoBehaviour
4  {
5    void Awake()
6    {
7      // Tento kód je vykonán při inicializaci jako první
8    }
9    void Start()
10   {
11     // Tento kód je vykonán při inicializaci jako druhý
12   }
13   void Update()
14   {
15     // Tento kód se vykonává cyklicky a obsluhuje herní logiku
16   }
17   void LateUpdate()
18   {
19     // Tento kód se vykonává cyklicky až po skončení Update()
20     // všech tříd
21   }
22 }
```

Zdrojový kód 2: Vytvoření scriptu vlastní komponenty

tedy možné instrukce pro program, nebo interaktivní objekty, a navrhnu možná rozšíření.

2.5.1 Struktura projektu

Jak jsem již uvedl v kapitole 2.3, tak kořenový adresář projektu je `Assets`. Nyní tedy popíšu základní adresářovou strukturu od kořene.

- `MapsData` - zde jsou uloženy XML soubory s nastavením jednotlivých úrovní
- `Models` - zde jsou uloženy veškeré surové modely - tedy 2D i 3D objekty a jejich animace, pokud jsou z nich následně vytvořeny sestavy „Prefab Asset“
- `Resources` - zde jsou uloženy hotové sestavy „Prefab Asset“, které jsou připravené pro použití ve hře
  - `GameObjects` - obsahuje 3D sestavy pro editor a samotnou hru
  - `UI` - obsahuje 2D sestavy používané jako grafické prvky v GUI, tedy celé dialogy, nebo jen jednotlivé prvky a ikony
- `Scenes` - obsahuje 3 scény a to hlavní menu, scénu editoru a scénu samotné hry

- `Scripts` - zde jsou uloženy veškeré skripty C#
  - `EventArgs` - adresář se scripty tříd používaných jako návratové hodnoty vyvolaných událostí
  - `GameEditor` - scripty, které obsluhují prvky v editoru map a slouží jako šablony vytvářených objektů pro samotnou hru
    - `UI` - scripty obsluhující formuláře dialogů
  - `GamePlay` - scripty pro samotnou hru, které jsou dále ještě rozděleny podle prostředí, ke kterému se vztahují
    - `PhysicalEnvironment` - fyzické prostředí ve kterém probíhá samotná hra, tedy podlaha a interaktivní objekty
    - `UI` - uživatelské rozhranní, které je dále rozděleno na grafické prvky reprezentující uživatelský program a další scripty obsluhující formuláře dialogů a interakce s ovládacími prvky GUI
    - `Vehicle` - scripty obsluhující funkcionality vozidla, jako je přístup k animacím, pohyb a vykonávání programu
      - `ProgramController` - scripty, které reprezentují jednotlivé elementy uživatelského programu, paměť programu a vykonávání programu
  - `Menu` - scripty obsluhující scénu hlavního menu, tedy hlavní dialog a vykreslování map v okně náhledu

Jak jsem již zmínil, tak soubory map jsou ukládány ve formátu XML. V tomto souboru jsou uloženy základní informace o dané mapě. Ve zdrojovém kódu č. 3 je ukázkový soubor s jednoduchou mapou velikosti 4x4 body s jedním vozidlem a jedním objektem krystalů. Kořenovým uzlem je `MapData`. Do něj jsou vnořeny další uzly.

- `MapSize` - definuje rozměr mapy
- `Camera` - pozice ve 3D prostoru, úhel natočení ve 3 osách a přiblížení kamery
- `MapLines` - zde jednotlivé uzly reprezentují řádky mapy a hodnoty znamenají různé typy podlahy. Hodnota 0 znamená, že zde podlaha není, 1 je pak plná podlaha. Čísla 2 – 5 pak znamenají rohovou podlahu a její různá otočení.
- `GameObjects` - v tomto uzlu jsou uloženy všechny objekty jako jednotlivé uzly
  - `Item` - tento uzel pak od sebe rozlišuje jednotlivé objekty, konkrétně vozidlo a „harvestable“[6] objekty. Každý z těchto objektů má své nastavitelné parametry.

[6]Těmito objekty mám na mysli krystaly, vzorky a překážky, protože ve vlastní hře jsou jejich objekty opatřeny komponentou `HarvestableObject`.

```
<?xml version="1.0" encoding="utf-8"?>
<MapData>
  <MapSize height="4" width="3" />
  <Camera positionX="27,5" positionY="-22,6" rotationX="32,0"
     rotationY="327,4" zoom="30,0" />
  <MapLines>
    <Line cells="000" />
    <Line cells="010" />
    <Line cells="010" />
    <Line cells="000" />
  </MapLines>
  <GameObjects>
    <Item type="VehicleTemplate" name="Vozidlo" positionX="1"
       positionY="1" direction="North">
      <Settings vehicleNumber="1" availableInstructions="63"
         isRepairable="False" isDamaged="False" energyConsume="False"
          functionsLimit="0" instructionsLimit="0" energyFunCall="0"
         energyMoveFwd="0" energyMoveBwd="0" energyTrnLft="0"
         energyTrnLftMul="0" energyTrnRght="0" energyTrnRghtMul="0"
         energyHarvest="0" energyCollect="0" energyDrop="0"
         energyInspect="0" energyRepair="0" />
    </Item>
    <Item type="HarvestableTemplate" name="Krystaly" positionX="1"
       positionY="2" direction="North">
      <Settings model="Crystals" isObservable="True"
         observeIsObjective="True" observedName="Prozkoumané krystaly
         " isPickupable="False" pickupIsObjective="False"
         isHarvestable="False" harvestIsObjective="False" />
    </Item>
  </GameObjects>
</MapData>
```

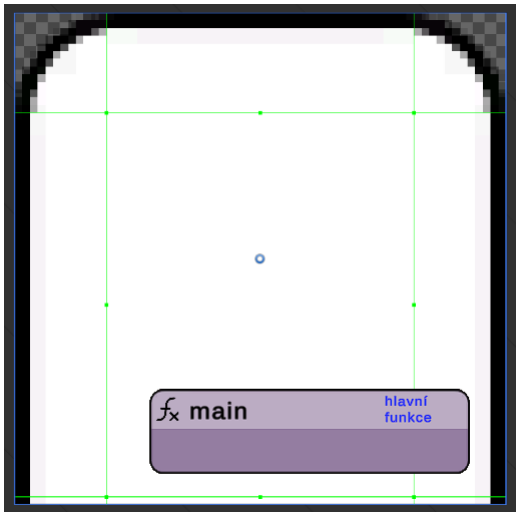
Zdrojový kód 3: Ukázka XML souboru s uloženou mapou

2.5.2 2D grafika GUI

Grafické prvky pro GUI jsem vytvářel jako vektorovou grafiku v programu Affinity Designer[8]. Pro použití v Unity jsem však jednotlivé grafické prvky musel uložit ve formátu Portable Network Graphics (PNG) a následné z nich vytvořit tzv. „sprite“. Výhoda užití „sprite“ spočívá v tom, že obrázek nemusí mít shodné rozměry, jako prvek, kde ho chci použít. Například „sprite“ pro pozadí grafického prvku „funkce“ má rozlišení 64x64 bodů. Černá barva obrázku zůstane zachována a bílá barva může být změněna. „Sprite“ umožňuje rozdělení na části, které v případě změny velikosti obrázku mohou být zachovány, pokud by došlo ke změně poměru stran. Na obrázku č. 5 předvádím použití „sprite“ na horní části grafického prvku „funkce“, kde je změněna barva pozadí, velikost je transformována do obdélníku, ale zakulacené rohy nejsou zkresleny.

Sprite je pak přiřazen komponentě `Image` jako hodnota vlastnosti `Source Image`. Objekt `GameObject` může mimo zmíněné komponenty `Image` obsa-

hovat i další komponenty. Například `Animator`, který umožňuje tento objekt animovat, nebo jiný uživatelský script.



Obrázek 5: Ukáza použití „sprite“

2.5.3 3D grafika

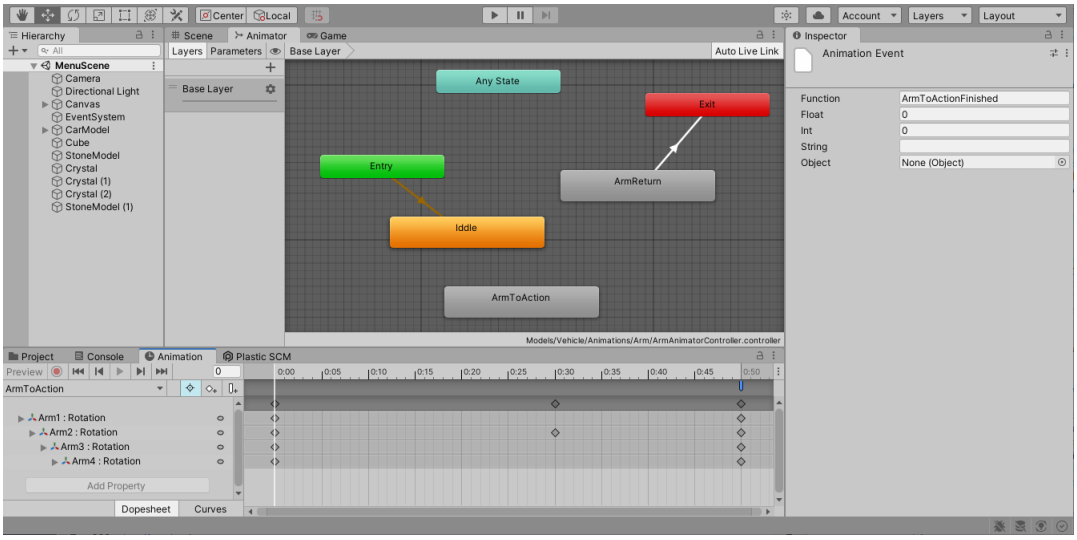
Pro tvorbu 3D modelů jsem si vybral program Blender[7]. Tento nástroj mi umožnil vytvořit základní model každého objektu i jeho texturu. Blender umožňuje i tvorbu animací, které mohou být do Unity importovány. Přesto jsem ale animace vytvářel až v Unity. Pro samotný import `.blend` souboru do Unity je nutné v Blenderu nastavit, aby uváděl všechny cesty jako relativní[7] a zabalit všechny další soubory, včetně textur, do `.blend` souboru[8].

Přetažením importovaného souboru do scény dojde k vytvoření objektu typu `GameObject`. Pro vykreslování 3D objektů slouží komponenty `MeshFilter` a `MeshRenderer`. První z komponent má přiřazený samotný model k vykreslení a druhá definuje jeho další vlastnosti, jako jsou materiály, vrhání stínů atd...

Pro animaci objektu je třeba danému objektu přidat komponentu `Animator` a jí přiřadit `Animator Controller`. Ten se vytvoří například v podokně „Project“ kliknutím pravým tlačítkem myši a v kontextovém menu výběrem `Create > Animator Controller`. Nyní je možné v podokně „Animation“ vytvářet nové animace pomocí klíčových snímků obdobně, jako v Blenderu. K nastavení přechodů mezi animacemi pak slouží podokno „Animator“.

Na obrázku č. 6 je pak vidět práce s Animatorem. Uprostřed jsou jednotlivé animace, které jsou obsaženy v jedné komponentě `Animator` a pomocí šipek se nastavují jejich návaznosti a přechody. Ve spodní části, v podokně `Animation` jsou pak detaily jednotlivých animací. Tedy klíčové snímky a objekty, kterých

[7] File > Externat Data > Make All Paths Relative
[8] File > External Data > Pack All Into .blend



Obrázek 6: Ukázka práce s Animatorem

se to týká. V pravé části je pak v podokně „Inspector“ nastavení události, která se vyvolá po dokončení animace. Událostí během animace lze vyvolat více, vždy na konkrétním čísle snímku v podokně `Animation`.

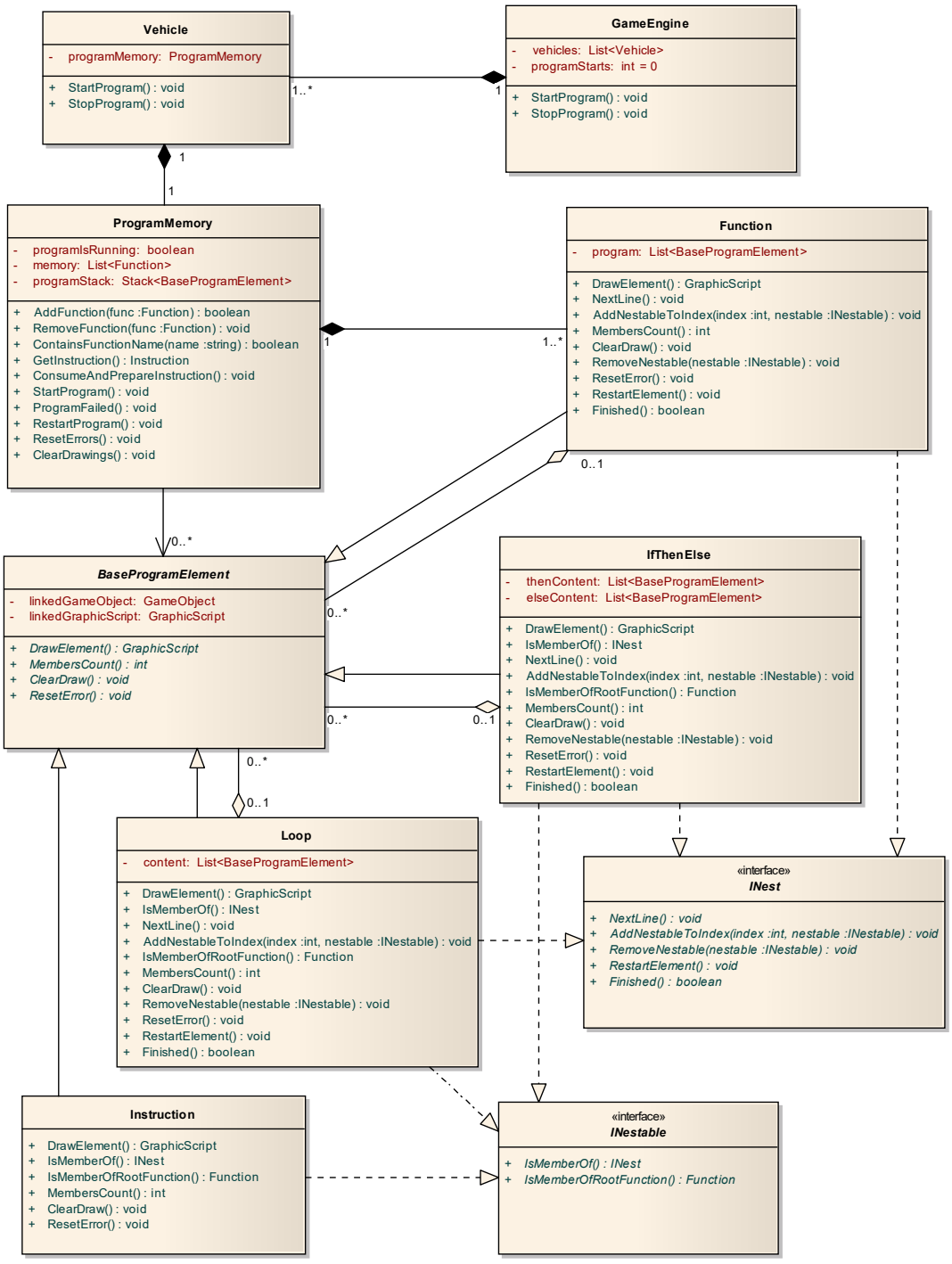
2.5.4 Popis hlavních tříd

Hlavní třídy, reprezentující uživatelský program, jsem zakreslil do diagramu tříd na obrázku č. 7. To se týká jen scény `GamePlayScene`. V ostatních scénách se uživatelský program nevytváří. Při načtení scény je vytvořen `GameObject` s názvem „World“. V editoru se nachází v podokně „Hierarchy“. Tento objekt má přiřazenou komponentu `GameEngine`, která zprostředkovává interakci tříd obstarávajících GUI, vozidel ve hře a dalších objektů. Při načtení mapy z XML souboru si třída uloží reference na všechna vytvořená vozidla třídy `Vehicle` do seznamu.

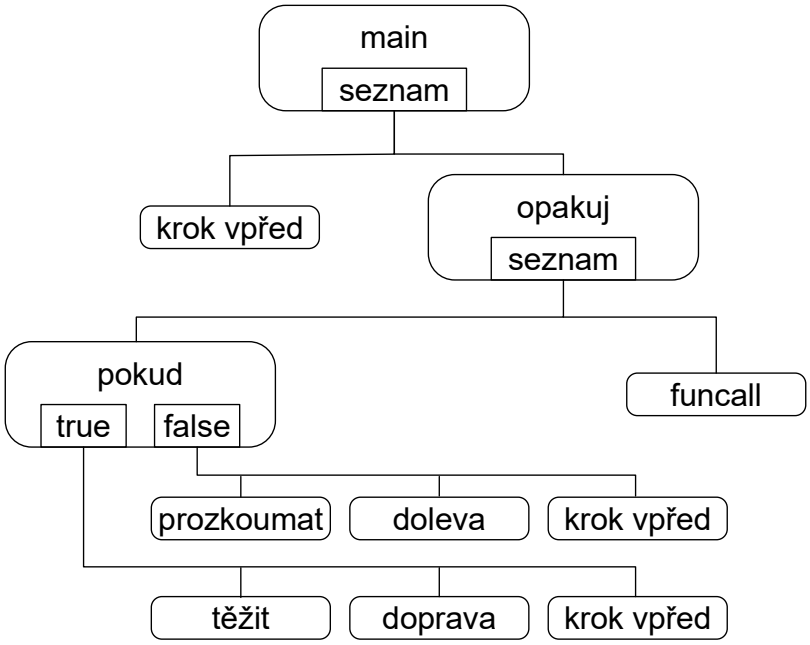
Každý objekt třídy `Vehicle` obsahuje jeden objekt třídy `ProgramMemory`, který reprezentuje paměť pro uživatelský program. Tato paměť se následně stará o vykonávání programu, kdy metodou `ConsumeAndPrepareInstruction` se prochází uložený program a metodou `GetInstruction` se získá instrukce, která je aktuálně na řadě.

Paměť obsahuje vždy minimálně jeden objekt třídy `Function`. Funkce pak mohou obsahovat další objekty, dědící abstraktní třídu `BaseProgramElement` a současně implementují rozhraní `INestable`. Tedy funkce samotná nemůže obsahovat další funkce. Objekty, které pak implementují rozhraní `INest`, mohou obsahovat další objekty třídy `BaseProgramElement`.

Tímto vzniká stromová struktura programu, která je při vykonávání procházena do hloubky. Názornou ukázku vzniklé stromové struktury jsem znázornil na obrázku č. 8.



Obrázek 7: Diagram tříd ve vztahu k interní reprezentaci programu



Obrázek 8: Stromová struktura programu

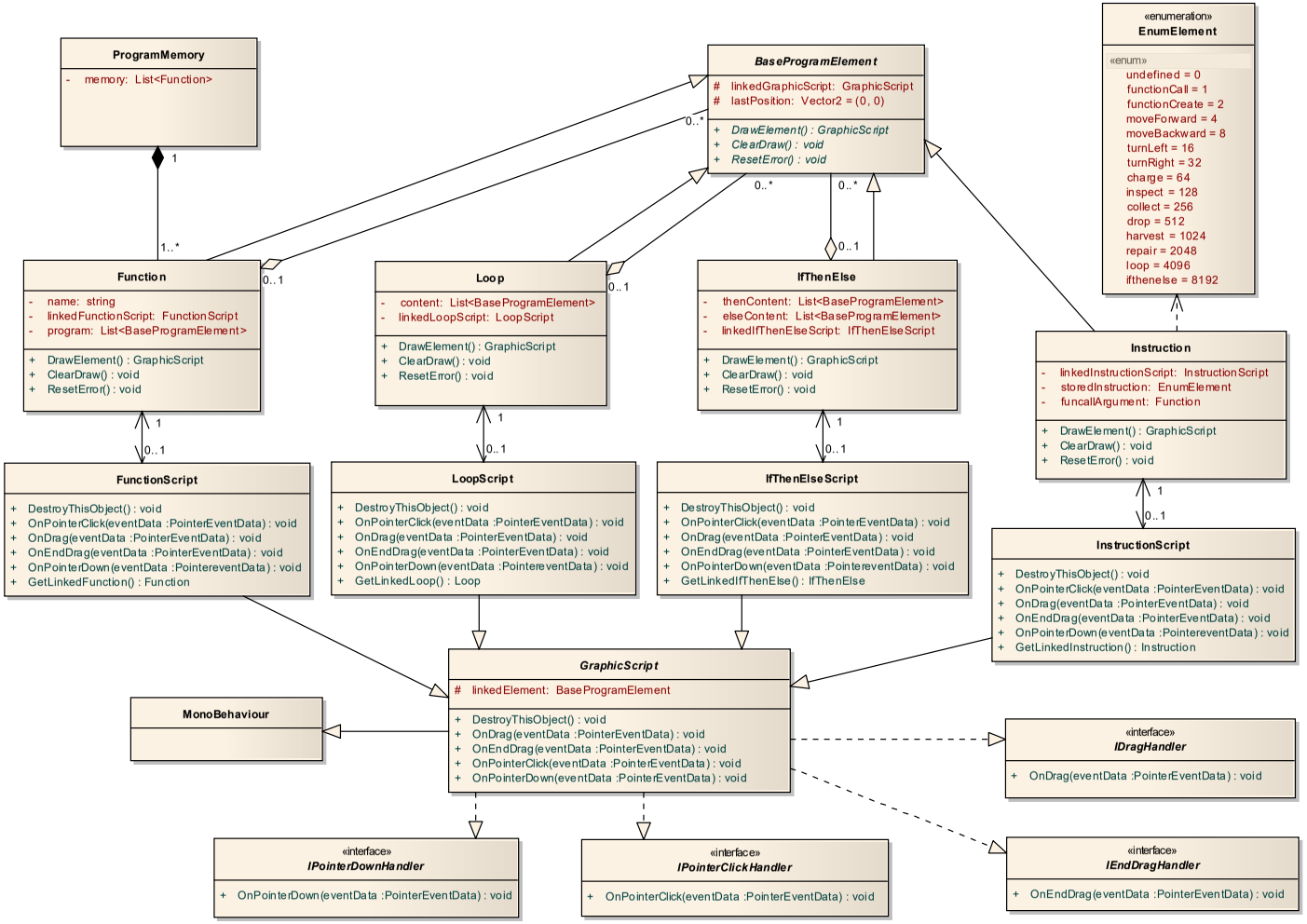
Samotný průchod stromem do hloubky probíhá tak, že na začátku je na zásobník vložena funkce „main“. Funkce je pomocí metody `GetInstruction` žádána o další instrukci, která je na řadě. Pokud je tou další instrukcí „volání funce“, pak je na zásobník vložena volaná funkce. Odstraněna je ze zásobníku, pokud objekt implementující rozhraní `INest` vrátí metodou `Finished` pravdu. Ve chvíli, kdy je ze zásobníku odstraněna funkce „main“, tak program skončil.

Na obrázku č. 9 jsem znázornil diagram tříd, ze kterého je patrná provázanost grafických elementů `GraphicScript` a objektů, které reprezentují jednotlivé části programu v paměti vozidla. Každý objekt třídy `BaseProgramElement` pak obsahuje referenci na objekt třídy `GraphicScript`. Pokud reference nemá správnou hodnotu – je „null“, pak objekt není vykreslen.

Obdobně, jako objekty, dědící z abstraktní třídy `BaseProgramElement` implementují rozhraní `INest` a `INestable`, tak objekty, které dědí abstraktní třídu `GraphicScript`, implementují rozhraní `IDock` a `IDockable`.

Abstraktní třída `GraphicScript` poskytuje svým potomkům virtuální metody, které definují, jak má třída reagovat na vstupy od uživatele. Tedy co má dělat při kliknutí myší, nebo tažení. Základní funkcionality jsou pohyb při tažení a smazání při kliknutí pravým tlačítkem. Tyto funkcionality si mohou třídy dědící `GraphicScript` přepsat. Tedy například `FunctionScript` musí při události kliknutí pravým tlačítkem ověřit, jestli se hráč nesnaží smazat funkci „main“ a v takovém případě událost zamítnout.

Tyto grafické elementy pak reagují podle toho jak uživatel kliká na jednotlivé elementy a posouvá je po pracovní ploše, a tím ovládá přiřazené objekty, tedy potomky třídy `BaseProgramElement`. Ukládá si v nich svou poslední polohu pro



Obrázek 9: Diagram provázanosti tříd vnitřní reprezentace programu a GUI

opětovné vykreslení a nebo volá jejich metody, zajišťující vkládání elementů jako obsah jiného elementu pomocí metody `AddNestableToIndex` poskytované rozhraním `INest`. Samotná metoda `AddNestableToIndex` přijímá dva argumenty. První z nich je typu `integer` a udává na jaké místo v programu se má zařadit objekt přijímaný v druhém argumentu, tedy typu `INestable`. U elementu „podmínka“, tedy třídy `IfThenElse`, jsou ale dva seznamy, do kterých se může element zařadit. Správný seznam se tak určuje podle prvního argumentu, kdy pokud se jedná o číslo $n \geq 0$, tak je zařazen do seznamu `thenContent` na pozici n. Pokud se jedná o číslo $n < 0$, tak je zařazen do `elseContent` na pozici $|n| - 1$.

Dále jsou důležité šablony reprezentující herní objekty. Základní třídou, jejíž jsou ostatní třídy potomky, je `BaseItemTemplate`. Z ní tedy dědí třída `VehicleTemplate`, která reprezentuje hlavní vlastnosti vozidel. Dále pak třída `HarvestableTemplate`, která reprezentuje hlavní vlastnosti ostatních objektů, jako jsou krystaly, vzorky a překážky. Tyto třídy obsahují nezbytné parametry těchto objektů, které stačí k rozlišení těchto objektů při načtení z XML souboru. V editoru se pak o nastavení hodnot v těchto třídách starají objekty třídy `EditorItem`. Ve hře už jsou ale jednotlivé herní objekty reprezentovány třídami `Vehicle` a `HarvestableObject`.

2.5.5 Možnost rozšíření

Hru jsem se snažil navrhnout tak, aby byla rozšiřitelná o další části uživatelského programu. Pokud by tedy mělo dojít k vytvoření nového elementu, musí být potomkem třídy `BaseProgramElement`. Dále musí implementovat rozhraní `INestable` a pokud by se mělo jednat o element, do kterého mohou být vkládány další elementy, pak musí implementovat rozhraní `INest`.

Pokud by se ale mělo jednat o rozšíření o nějaký element, který může obsahovat více, než dvě těla (například „switch“), tak by bylo nutné změnit i samotné rozhraní `INest`. Jak jsem zmínil v kapitole 2.5.4, tak pro zařazení je používán celočíselný typ. Je tedy možné využít jeho kladný a záporný rozsah, což pokryje jen rozdělení do dvou těl. Jako vhodné řešení se mi jeví přidání jednoho parametru typu `integer`, nebo výčtového typu, do metody `AddNestableToIndex` v rozhraní `INest`.

Dále je v abstraktní třídě `BaseProgramElement` výčtový typ dostupných instrukcí. Jakákoliv nová instrukce zde musí být zařazena a musí jí být přiřazena nová hodnota, dvojnásobek předchozí.

Je nutné také vytvořit k ní objekt, jako potomka třídy `GraphicScript`, a implementovat rozhraní `IDockable`. V případě, že by se mělo jednat o element, do kterého má být možné vkládat další elementy, pak musí implementovat i rozhraní `IDock`.

V poslední řadě je třeba vytvořit pro novou instrukci ikonu pro GUI, zajistit její načtení ve statické třídě `ResourceManager`, umístění ikony v GUI hry (obdobně, jako je na obrázku č. 15) a formuláři nastavení vozidla ve scéně editoru (jak je vidět na obrázku č. 13).

2.5.6 Návrh rozšíření

Nyní umožňuje hra jednoduché řízení toku programu, pomocí jednoduché smyčky a podmínky. Ve smyčce lze nastavit počet opakování. V podmínce větvení pak lze nastavit výraz „Je/Není“ před vozidlem „Volno/Krystaly/Vzorek/Vozidlo/Překážka“. Hru by bylo možné rozšířit o nový element „výraz“, který by byl použit jako argument pro smyčku, nebo podmínku. Takto by bylo například možné poskládat složitější výraz ve smyslu „pokud je před vozidlem překážka a je možné ji zvednout“. S tím by souviselo i vytvoření nové podmínky a nové smyčky, které by takový argument přijímaly, nebo úprava stávajících objektů.

Další možností, jak hru rozšířit, je přidat omezení na počet pokusů pro některé úrovně. Třeba jen režim hry „na jeden pokus“. V tomto režimu hry by hráč musel vymyslet program tak, aby všechny cíle mise byly splněny na jedno stisknutí tlačítka „Start“. V opačném případě by došlo k selhání mise.

Jako další rozšíření mě napadá ozvučení hry. V současné verzi nejsou použity žádné zvuky. Přehrávání nějaké příjemné melodie v pozadí, nebo zapracování zvuků jako odezvu na některé akce uživatele, případně reakci na vykonávání některých částí programu, by jistě zvýšilo uživatelský zážitek ze hry.

Tabulka 3: Specifikace počítače, na kterém byla hra testována

Operační systém	OS Windows 10 64 bit., verze 21H2
CPU	Intel® Core™ i5-6400
Operační paměť	16 GB
Graficka karta	AMD Radeon™ R7 360 Series
Rozlišení obrazovky	1920x1080 px

3 Uživatelská dokumentace

Hra byla vytvářena a testována na počítači se specifikací uvedenou v tabulce č. 3. V následujících podkapitolách popíšu aplikaci z pohledu uživatele. Popíšu důležité adresáře a soubory v aplikaci, dále pak hlavní menu, základní ovládání, editor map a vlastní hru.

3.1 Adresáře a soubory

Ve adresáři s aplikací se nachází spustitelný soubor `B-Code.exe`. Tímto souborem se hra spouští a jako první je načteno hlavní menu.

Soubory jednotlivých herních úrovní se nachází v adresáři map, konkrétní umístění je v `.\B-Code_Data\MapsData\`. Cesta je uvedena jako relativní k adresáři, ve kterém se hra nachází. Soubory, ve kterých jsou mapy uloženy, mají koncovku `.xml`.

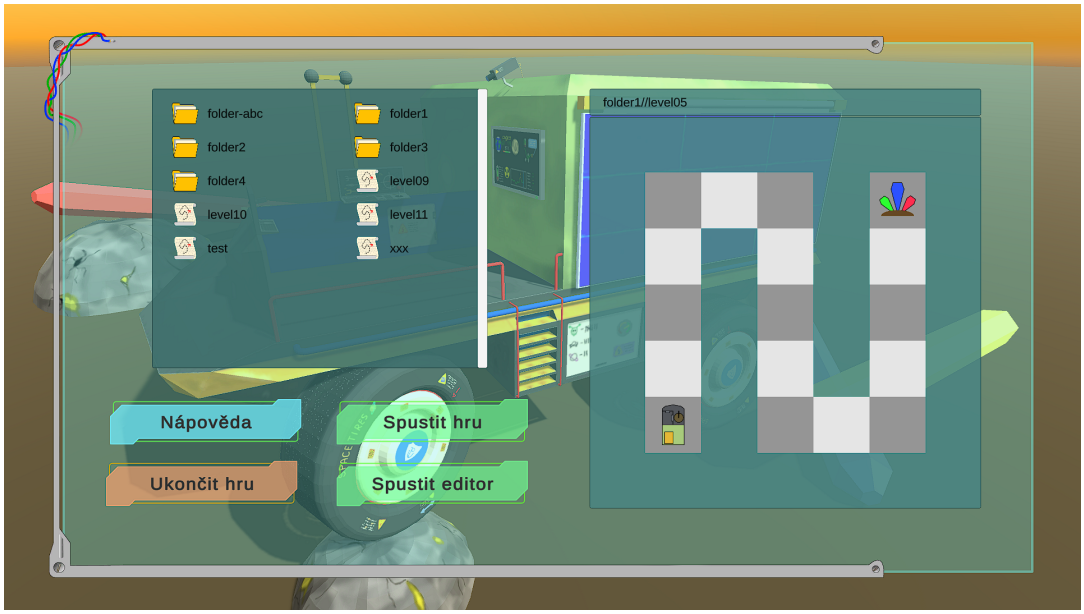
3.2 Popis hlavního menu

Při spuštění aplikace je jako první načteno hlavní menu, které je zobrazeno na obrázku č. 10. V levé části je možné vybrat soubor s mapou který, pokud je v pořádku, je následně zobrazen v pravé části. V pravé horní části je pak zobrazena cesta k souboru s mapou, která je zvolena. Název souboru je zde od adresáře oddělen dvěma lomítky.

Soubory jsou seřazeny podle abecedy. Pro spuštění hry tlačítkem „Spustit hru“ je nutné nejdřív nějaký soubor vybrat. Všechny soubory, které jsou abecedně zařazeny za vybraným souborem, se následně zařadí do „playlistu“ a ve hře budou postupně, po dokončení úrovně, načítány dokud nebude „playlist“ prázdný. Pak hra skončí a hráč bude přesměrován do hlavního menu.

Pokud je v hlavním menu vybrán nějaký soubor, tak kliknutím na tlačítko „Spustit editor“ dojde k načtení scény editoru map a do něj je načtena tato mapa. Pokud soubor vybrán není, otevře se v editoru prázdná mapa 10x10 polí.

Tlačítko „Nápověda“ pak otevře dialog nápovědy na stránce „O programu“ a tlačítko „Ukončit hru“ aplikaci ukončí.



Obrázek 10: Ukázka ze hry – hlavní menu

3.3 Základní ovládání

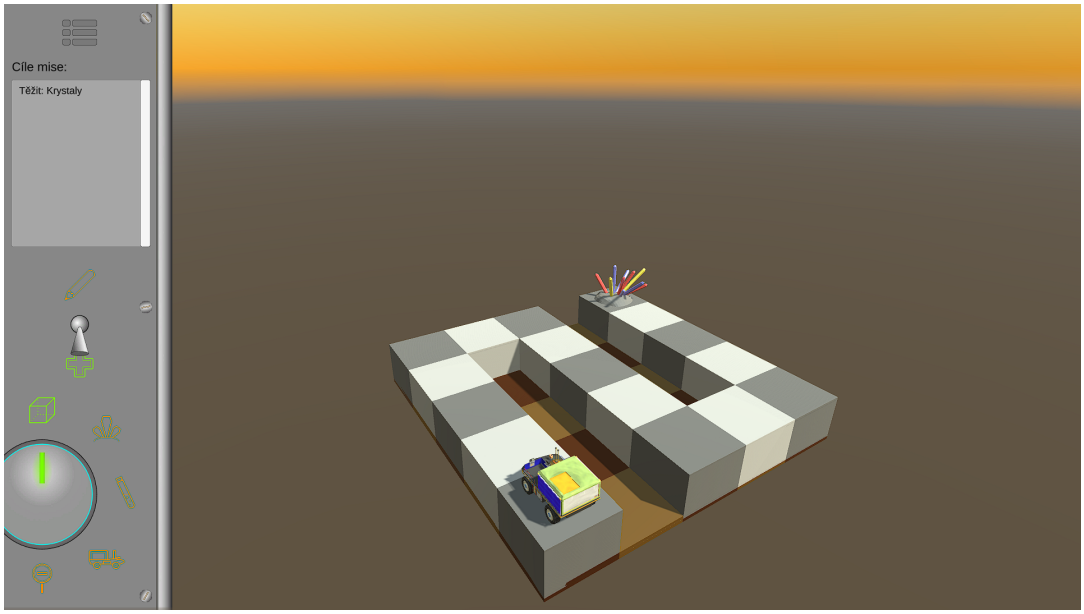
Pohled hráče se v editoru i ve vlastní hře ovládá pomocí myši. Přidržením pravého tlačítka a následným pohybem myši se ovládá otáčení pohledu do stran. Přidržením prostředního tlačítka a následným pohybem myši se pak pohled posouvá po vodorovné ose do stran. Pohyb po vodorovné ose je také možné provádět pomocí kurzorových kláves. Pomocí kolečka myši je pak ovládáno přiblížení kamery.

3.4 Editor map

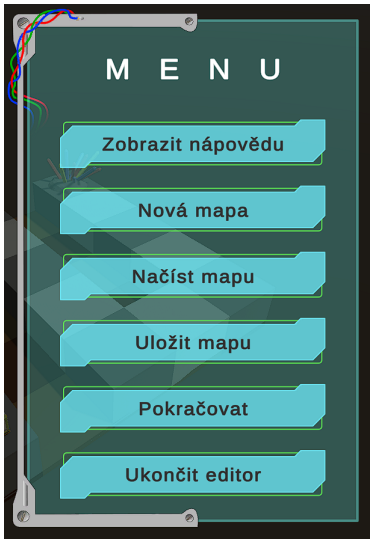
Na obrázku č. 11 je zobrazen náhled na editor s načtenou mapou. V levé části je ovládací panel. Na ovládacím panelu se nahoře nachází tlačítko menu. Jeho stisknutím se zobrazí dialog „MENU“, který je znázorněn na obrázku č. 12. Pod tímto tlačítkem se nachází seznam cílů mise, kam jsou zařazovány úkoly pro hráče.

Dále je zde přepínač pro režim vkládání a editace. Pokud je přepínač v režimu editace (*symbol tužky*), pak kliknutím na objekty ve výhledu hráče bude zobrazen dialog, kde je možné definovat vlastnosti těchto objektů. Například při kliknutí na vozidlo je zobrazen dialog „Nastavení vozidla“, který je k vidění na obrázku č. 13. Obdobně funguje kliknutí na krystaly, vzorky a překážky.

Pokud je přepínač v režimu vkládání (*symbol plus*), pak je aktivní i přepínač objektů, který je níže. V tomto režimu se ještě rozlišuje vkládání podlahy, krystalů, vzorků, vozidel a překážek. Pokud je aktivní vkládání podlahy, tak kliknutím levým tlačítkem myši do mapy se vytvoří podlaha. Opětovným klikáním



Obrázek 11: Ukázka ze hry – editor



Obrázek 12: Ukázka ze hry – menu editoru

Obrázek 13: Ukázka ze hry – nastavení vlastností vozidla

dojde k jejímu rozpůlení a následně rotaci kolem svislé osy. Pravým tlačítkem je možné podlahu odstranit. V tomto případě se při kliknutí na jiný objekt, než na podlahu, nic nestane. Pokud je zvoleno vkládání krystalů, vzorků, vozidel, nebo překážek, tak kliknutím levým tlačítkem do mapy dojde k vytvoření tohoto objektu. Opětovným kliknutím levého tlačítka lze objekt otáčet kolem svislé osy po směru hodinových ručiček. Kliknutím pravého tlačítka dojde k odstranění objektu.

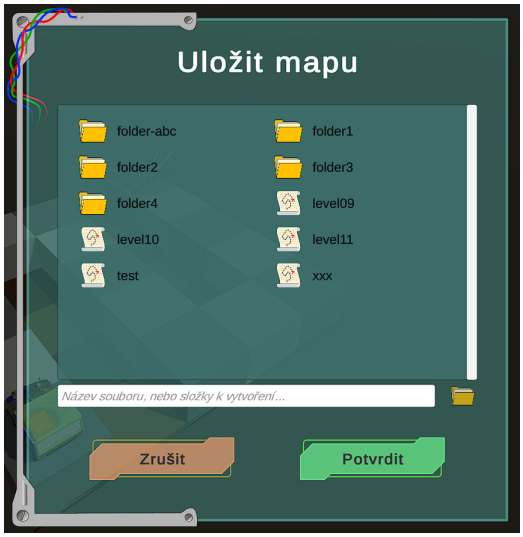
Po vytvoření objektů je tedy nutné ještě provést jejich nastavení. Tedy horní přepínač přepnout do režimu editace (*symbol tužky*) a jednotlivým objektům nastavit jejich vlastnosti.

U krystalů je možné nastavit jim jméno, zda lze objekt prozkoumat a zda je prozkoumání tohoto objektu cílem mise. Také je možné nastavit, jak se bude objekt jmenovat po prozkoumání. Dále je možné nastavit, zda je možné objekt těžit a zda je těžba cílem mise. Pokud jsou nastaveny cíle mise, pak se zobrazí v seznamu v ovládacím panelu.

U vzorků je možné nastavit jim, stejně jako u krystalů, jméno, možnost prozkoumání, jméno po prozkoumání a zda je prozkoumání cílem mise. Dále je zde možnost, jestli lze objekt zvednout a jestli je zvednutí objektu cílem mise.

U překážek lze nastavit jen jejich jméno, zda je možné překážku zvednout a zda je zvednutí překážky cílem mise. Ve hře pak, pokud lze překážku zvednout, je možné překážku i položit. Položit překážku je pak nutné, pokud má vozidlo zvednout něco jiného.

Nastavení vozidla zahrnuje změnu jeho názvu a čísla. Dále je možné aktivovat dostupné instrukce pro vytváření uživatelského programu. Je možné omezit maximální počet funkcí. V takovém případě nebude moci paměť vozidla obsahovat více funkcí, než je dovoleno. Obdobně je možné omezit počet instrukcí



Obrázek 14: Ukázka ze hry – uložení mapy

vkládaných do těchto funkcí. Dále je možné nastavit, zda se vozidlo při chybě ve vykonávání programu poškodí. V takovém případě, kdy došlo k poruše vozidla, je vozidlo nadále nepoužitelné. Dokud není opraveno jiným vozidlem. Současně může být vozidlo nastaveno jako poškozené již na začátku hry a jeho oprava může být jeden z cílů mise. V poslední řadě je možné nastavit spotřebu energie jednotlivých instrukcí a tím ztížit hráči průchod danou úrovní. Pokud například spotřebovává energii a nemá dostupnou instrukci nabíjení, musí hráč pečlivě vybírat, které instrukce použije. Pokud vozidlu dojde energie, tak dojde k chybě programu a vozidlo se zastaví. Pokud je k dispozici instrukce nabíjení, tak nedojde k poškození vozidla. Pokud ale instrukce nabíjení dostupná není, pak dojde k poškození vozidla i v případě, že je vozidlo nastaveno jako nepoškoditelné. To z důvodu nemožnosti vozidlo dobít a misi dokončit.

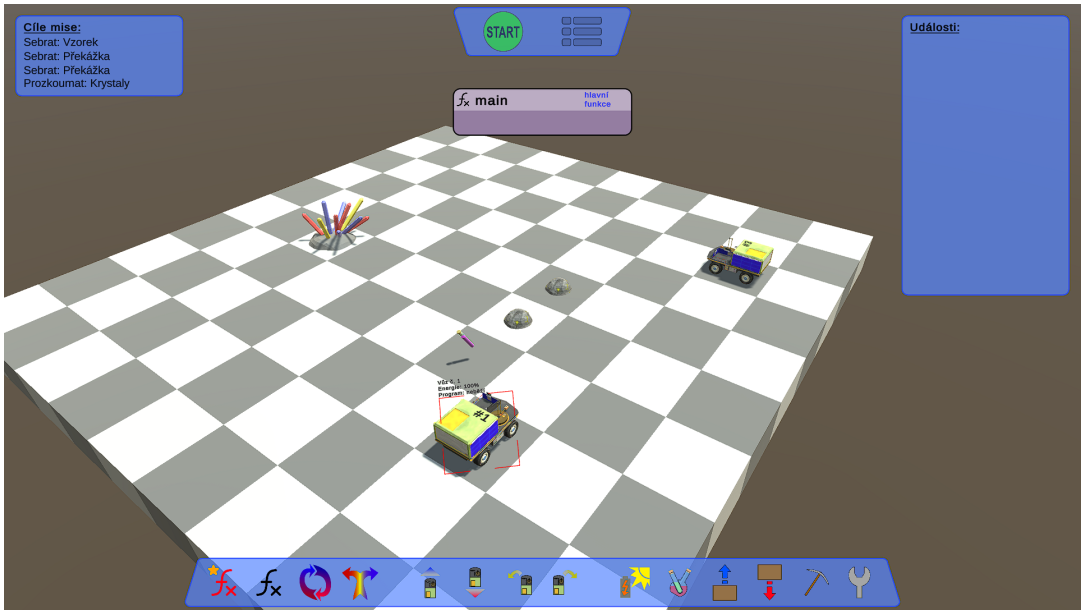
Ve chvíli, kdy je mapa vytvořena a objekty jsou nastaveny, je možné hru uložit pomocí volby „Uložit hru“ v menu. Je zobrazen dialog k vybrání cesty a názvu souboru, který je vyobrazen na obrázku č. 14.

V dialogu pro uložení souboru je také možné vytvářet adresáře zadáním názvu do vstupního pole a kliknutím na ikonu složky vpravo od tohoto vstupního pole.

Do uloženého souboru je zaznamenána i aktuální poloha kamery a při spuštění této úrovně je pak kamera nastavena do stejné polohy v jaké byla uložena. Tím je možné změnit prvotní pohled na mapu při spuštění hry.

3.5 Vlastní hra

Při spuštění vlastní hry se hráči načte mapa, do které jsou vykresleny veškeré interaktivní objekty, jak je vidět na obrázku č. 15. Při kliknutí levým tlačítkem myši na tyto objekty je u nich zobrazena krátká informace, co reprezentují. Při kliknutí na vozidlo je zobrazen i program tohoto vozidla. V případě nové hry pak



Obrázek 15: Ukázka ze hry – spuštění hry

tedy pouze prázdná funkce „main“. Kliknutím pravým tlačítkem myši na vozidlo je možné zobrazit dialog, kde je popsáno nastavení tohoto vozidla. Tedy jestli se může poškodit, jeho případné paměťové omezení, nebo kolik spotřebovává energie.

V levé horní části hráč vidí panel „Cíle mise“, kde jsou vypsány veškeré úkoly, které musí splnit pro úspěšné dokončení úrovně. V pravé části je pak panel „Události“, kam se zapisují události v průběhu hry. Tedy například, že došlo ke splnění úkolu, nebo chybě nějakého vozidla. Kliknutím na tuto událost je pak možné označit vozidlo, ve kterém k chybě došlo. Ve spodní části je pak panel dostupných instrukcí. Při najetí myši na jejich ikony je zobrazen krátký popisek, co daná instrukce znamená. Tažením myši z těchto ikon se dané elementy vytvářejí, což bude popsáno v kapitole 3.5.1.

Do programu se zapisují pouze ty instrukce, jejichž elementy jsou vloženy do nějaké funkce. Ostatní se nevykonají a při označení jiného objektu, dojde k jejich úplnému odstranění. Mazat vytvářené instrukce a funkce je možné i kliknutím pravým tlačítkem myši na daný element. Jediný element, který nelze smazat je funkce „main“.

V horní části obrazovky se pak nachází tlačítko „Start/Stop“, které spustí programy ve všech vozidlech současně. Vedle něj je tlačítko, které otevře dialog „MENU“. Tam je pak možné hru ukončit, restartovat, nebo zobrazit nápovědu ke hře.

Každý stisk tlačítka „Start“ se započítá a po splnění úrovně je celkový počet zobrazen. Hráč tak má možnost vidět, kolik potřeboval pokusů ke splnění dané úrovně. Úkolem hráče je vytvořit program, jehož spuštěním dojde ke splnění

všech cílů mise na nejmenší možný počet pokusů.

3.5.1 Tvorba programů

Aby hráč mohl dokončit jakoukoliv úroveň, musí nejdříve vytvořit a spustit program. Vozítka ve hře tento program vykonají a cílem je, aby během toho splnily pokud možno všechny cíle mise. Každé vozítko má svou paměť pro program. Tato paměť na začátku vždy obsahuje pouze prázdnou funkci „main“. Může ale obsahovat i uživatelské funkce. Na obrázku č. 15 je pak vidět prázdná hlavní funkce „main“.

Program hráč vytváří tím, že do těchto funkcí vkládá jednotlivé instrukce. Ať jsou to instrukce k pohybu, nebo k řízení toku programu, kde hra nabízí volání funkce, smyčku a větvení pomocí podmínky.

Instrukce se vytváří přidržením levého tlačítka myši na ikoně žádaného elementu. Mažou se pak kliknutím pravým tlačítkem myši na mazaný element. Tažením myši se stisknutým levým tlačítkem myši přes jednotlivé elementy se pak elementy přesouvají a vkládají do sebe. Pokud element, přes který táhneme jiný, může obsahovat tažený element, tak pro něj graficky vytvoří místo a je možné uvolnit levé tlačítko myši a tažený element se vnoří na uvolněné místo. Pokud by v cílové funkci již nebylo dost místa kvůli omezení velikosti paměti, tak o tom hráč bude informován zprávou v panelu „Události“. Na obrázku č. 16 je pak k vidění již vytvořený a spuštěný program.

K odpojení vnořeného elementu pak dojde opět jeho tažením myší, nebo při jeho smazání pravým tlačítkem myši. Při smazání nějakého elementu pak vždy dojde i ke smazání všech elementů, které jsou do něj vnořeny.

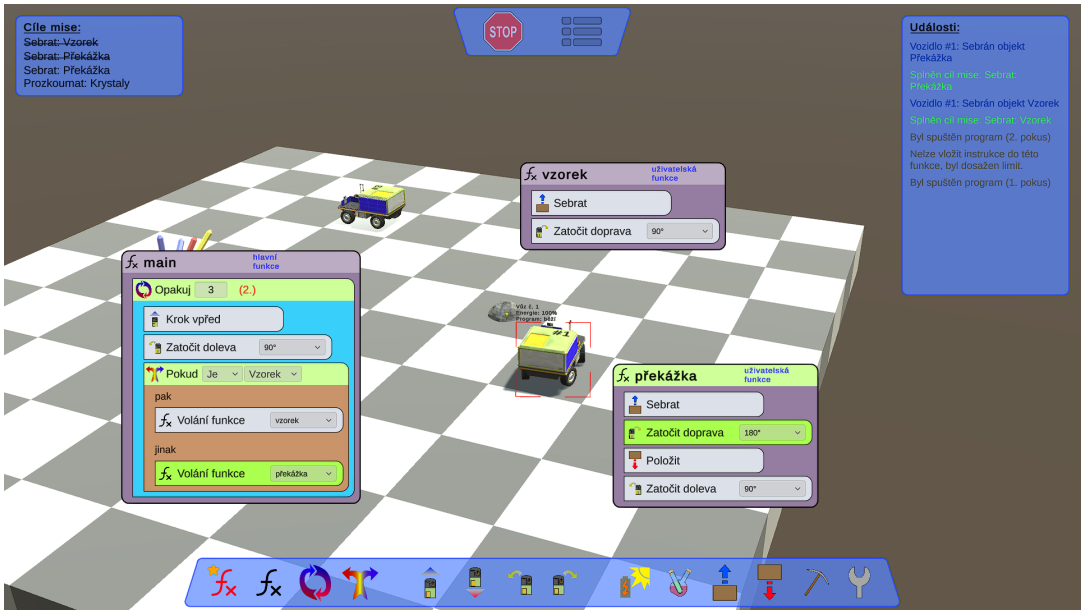
Elementy, které jsou vytvořeny, ale nejsou vnořeny do žádné funkce, se při spuštění programu nevykonají. Při označení nějakého objektu ve hře pak dojde k jejich trvalému smazání. Zachovány budou pouze funkce a jejich obsah.

3.5.2 Vykonání programu

Pokud hráč tedy vytvoří program pro jednotlivá vozidla ve hře, kliknutím na tlačítko „Start“ dojde ke spuštění programů všech těchto vozidel. Výjimkou jsou vozidla, která jsou poškozená. U těch lze spustit program jen jejich opravou, kterou může provést jiné vozidlo.

Jak je vidět na obrázku č. 16, tak průběh programu je graficky znázorněn zelenou barvou. Hráč tak vidí, která instrukce se aktuálně vykonává a odkud se k této instrukci program dostal. V případě, že dojde k chybě programu, tak jsou tyto části programu označeny červenou barvu.

Vozidlo nesmí během vykonávání programu nabourat do jiného předmětu, sjet z cesty, ani použít instrukci na objekt, který na to není nastaven. Tedy například se nesmí pokusit zvednout překážku, která je nastavena, že nelze zvednout. V těchto případech dojde k chybě programu a program se zastaví. Pokud je vozidlo nastaveno tak, že se má poškodit při chybě programu, tak se poškodí a další program nebude možné spustit, dokud nedojde k opravě vozidla.



Obrázek 16: Ukázka ze hry – vykonání programu

Pokud je vozidlo nastaveno tak, aby spotřebovávalo energii a energie mu dojde, poškodí se pouze v případě, že nemá povoleno použít instrukci nabíjení. Jak bylo uvedeno v kapitole 3.4, tak k poškození dojde i když je vozidlo nastaveno jako nepoškoditelné.

Při vykonávání jednotlivých instrukcí je ověřováno plnění cílů mise. Splněním všech cílů míse je úroveň úspěšně dokončena. V případě, že se poškodí poslední vozidlo ve hře chybou v programu, pak mise skončí neúspěchem a je nutné začít od začátku.

Závěr

V rámci této bakalářské práce byla vytvořena počítačová hra, která hráče provede tvorbou jednoduchých algoritmů a motivuje jej k rozvoji algoritmického myšlení. Hra splňuje všechny požadavky zadání. Navíc byl implementován editor map, který je nejlepší možností pro snadné doplnění obsahu hry.

Cílem textové části této práce bylo seznámení čtenáře s editorem Unity a základy tvorby her v tomto prostředí. Dále byl čtenář seznámen s programátorskou dokumentací a možnostmi rozšíření aplikace. Nakonec pak s uživatelskou dokumentací, kde bylo vysvětleno ovládání celé aplikace.

Conclusions

The aim of this bachelor's thesis was to create a computer game that guides the player through the creation of simple algorithms and motivates him to develop algorithmic thinking. The game meets all the requirements of the assignment. In addition, a map editor, which is the best option how to easily add extra content to the game, has been implemented.

The text part of this work acquaints the reader with the Unity editor and the basics of creating games in this environment. Furthermore, the reader was acquainted with the software documentation and the possibilities of extending the application. Finally, this part contains the user documentation that explains how to control the entire application.

A Obsah přiloženého DVD

bin/
: Obsahuje sestavenou hru pro OS Windows.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně ZIP archivu se všemi přílohami a všemi soubory potřebnými pro bezproblémové vygenerování PDF dokumentu textu.

src/
: Kompletní projekt Unity včetně všech zdrojových kódů a grafických prvků vytvořené aplikace.

readme.txt
: Instrukce pro spuštění aplikace.

Navíc DVD obsahuje:

install/
: Instalační soubory Unity verze 2020.3.12f1 a Unity Hub.

graphics/
: Grafické prvky vytvořené v aplikaci Affinity Designer.

models/
: 3D modely vytvořené v aplikaci Blender.

Seznam zkratek

API Application programming interface

GUI Graphical User Interface

PNG Portable Network Graphics

SSE2 Streaming SIMD Extension ver. 2

XML Extensible Markup Language

Literatura

[1] UMÍME INFORMATIKU. *Webová stránka hry Robotanik* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://www.umimeinformatiku.cz/robotanik⟩.

[2] UMÍME TO. *Webová stránka projektu Umíme to* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://www.umimeto.org/⟩.

[3] UMÍME TO. *Rodinná licence projektu Umíme to* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://www.umimeto.org/licence-personal-info⟩.

[4] CODECOMBAT. *Webová stránka hry Codecombat* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://codecombat.com/⟩.

[5] SPRITEBOX LLC. *Webová stránka hry Lightbot* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://lightbot.com/⟩.

[6] UNITY TECHNOLOGIES. *Webová stránka Unity* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://unity.com/⟩.

[7] BLENDER FOUNDATION. *Webová stránka programu Blender* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://www.blender.org/⟩.

[8] SERIF EUROPE LTD. *Webová stránka grafického editoru Affinity Designer* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://affinity.serif.com/en-gb/designer/⟩.

[9] UNITY TECHNOLOGIES. *Podporované programovací jazyky* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html⟩.

[10] UNITY TECHNOLOGIES. *Podmínky užití osobní licence* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://unity3d.com/unity/activation/personal⟩.

[11] UNITY TECHNOLOGIES. *Poplatky licence pro komerční užití* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://unity3d.com/unity/activation/personal⟩.

[12] UNITY TECHNOLOGIES. *Systémové požadavky editoru Unity* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://docs.unity3d.com/Manual/system-requirements.html⟩.

[13] MICROSOFT CORPORATION. *Ukončení podpory systému Windows 7* [online]. 2020 [cit. 2022-4-27]. Dostupný z: ⟨https://www.microsoft.com/en-us/windows/windows-7-end-of-life-support-information⟩.

[14] UNITY TECHNOLOGIES. *Speciální složky v projektu* [online]. [cit. 2022-4-30]. Dostupný z: ⟨https://docs.unity3d.com/560/Documentation/Manual/SpecialFolders.html⟩.

[15] UNITY TECHNOLOGIES. *Podporované externí editory* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://docs.unity3d.com/Manual/Preferences.html#External-Tools⟩.

[16] UNITY TECHNOLOGIES. *Pořadí volání jednotlivých metod ve sctiptu* [online]. [cit. 2022-4-27]. Dostupný z: ⟨https://docs.unity3d.com/Manual/ExecutionOrder.html⟩.