Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Herní engine pro plošinové hry

2019

Jiří Hausner

Vedoucí práce: Mgr. Petr Krajča, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Jiří Hausner
Název práce:	Herní engine pro plošinové hry
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2019
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Petr Krajča, Ph.D.
Počet stran:	76
Přílohy:	1 CD
Jazyk práce:	český

Bibliograhic info

Author:	Jiří Hausner
Title:	Game engine for platform games
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2019
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Petr Krajča, Ph.D.
Page count:	76
Supplements:	1 CD
Thesis language:	Czech

Anotace

Výsledkem práce je herní engine pro plošinové hry, jeho editor a dvě jednoduché hry. Editor umožňuje pohodlné vytvoření nových her. Uživatel může přidávat animace, zvuky, objekty nebo úrovně, které se navrhují v editoru úrovní.

Synopsis

The result of this work is game engine for platform games, its editor and two simple games. The editor allows comfortable creation of new games. User can add animations, sounds, objects or levels, which can be designed in level editor.

Klíčová slova: herní engine; plošinové hry; herní editor; editor úrovní; C++; OpenGL

Keywords: game engine; platform games; game editor; level editor; C++; OpenGL

Děkuji vedoucímu práce Mgr. Petru Krajčovi, PhD. za vedení práce a cenné rady, které mi pomohly při návrhu a vývoji aplikace.

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 vět

Seznam zdrojových kódů

1 Úvod

Cílem práce bylo naprogramovat herní engine pro plošinové hry spolu s jeho editorem a dvěmi jednoduchými hrami. Engine měl podporovat vykreslování postavené na rozhraní OpenGL, základní herní logiku, ovládání, správce zvuků a hudby a základní uživatelské rozhraní. Herní editor měl sloužit k pohodlnému vytváření nových her, kde by uživatel mohl přidávat nové animace, zvuky, objekty, nebo úrovně, které se měly navrhovat v editoru úrovní.

Práce je rozdělena do čtyř kapitol. V první kapitole se zabývám popisem herního enginu, editoru her, popisuji hry v něm vytvořené a nakonec jej porovnávám s konkurenčním software. V další kapitole se věnuji použitým technologiím. Třetí kapitola obsahuje programátorskou příručku nástinující popis klíčových tříd herního enginu a editoru her, omezení aplikace ale i možná rozšíření. Poslední kapitola obsahuje průvodce tvorbou hry v editoru a detailně popisuje rozšíření enginu pomocí skriptovacího jazyka Lua.

1.1 Herní engine

Herní engine je navržen jako spustitelná aplikace, která vykonává herní kód v podobě vstupního souboru `.gamedata`, který je zároveň výstupem z editoru. Tento herní soubor obsahuje výčet zdrojových souborů jako jsou textury, animace, dlaždice, zvuky, fonty, návrhy úrovní, definice herních objektů a objektů uživatelského rozhraní a nakonec nastavení hry.

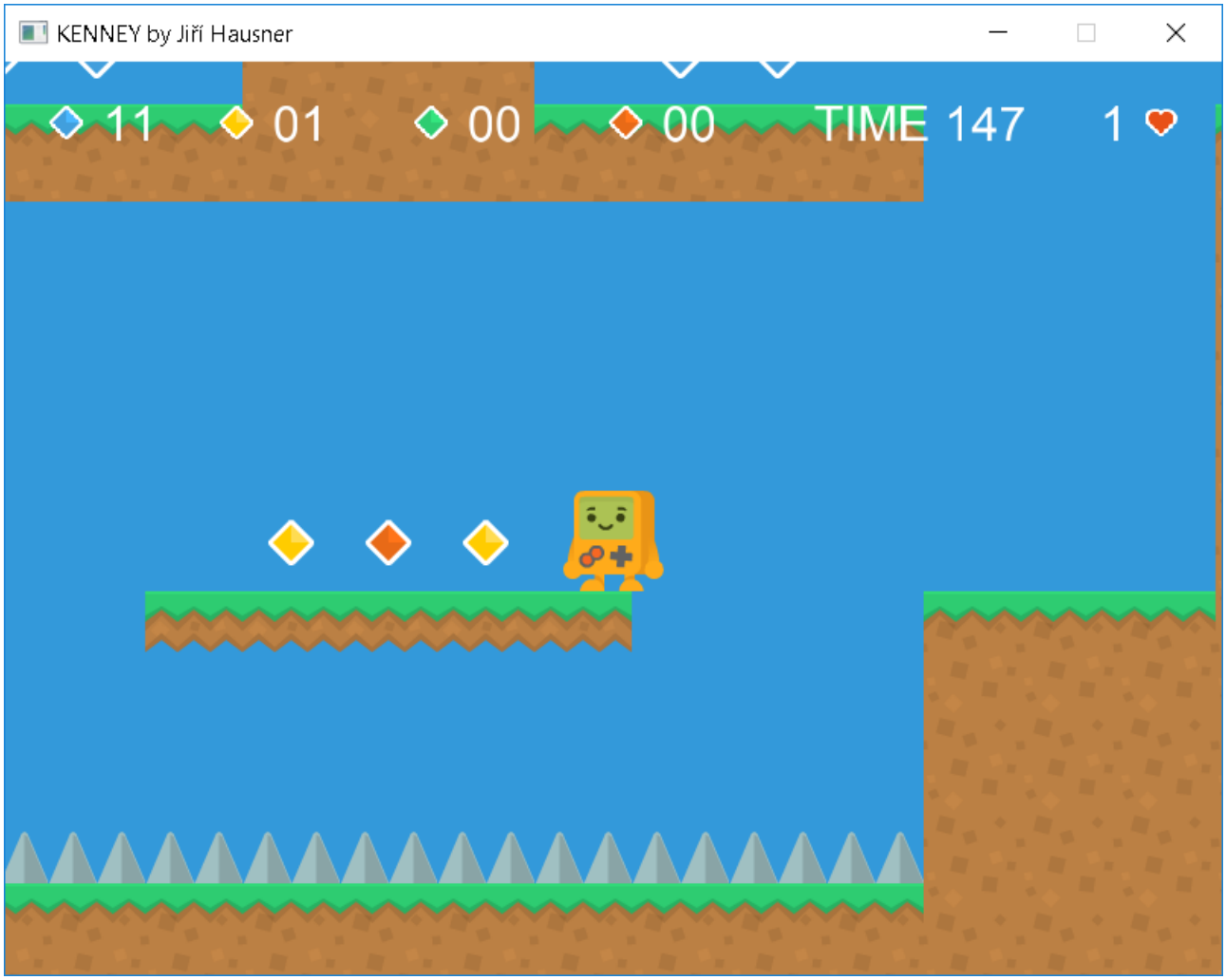
Engine nabízí kontrolu nad načtenými herními soubory spolu se zdrojovými soubory, práci s úrovněmi, fyziku a systém pro detekci kolizí, logiku základních herních objektů očekávaných v platformních hrách, přehrávání zvuků a hudby, vykreslování obsahu pomocí rozhraní OpenGL, ovládání pomocí myši, klávesnice a joysticků, implementaci jednoduchého uživatelského rozhraní spolu se základními prvky, nebo možností definovat si prvky vlastní. Engine je navržen tak, aby se dal jednoduše rozšířit o další možnosti. Avšak o tom více v kapitole 3.

1.2 Editor

Editor je aplikace uzpůsobená pro jednoduchou tvorbu nových her, umožňuje načtení zdrojových souborů jako jsou textury různých typů, zvuky a True Type fonty. Dále může uživatel vytvářet a definovat animace jakožto sled textur s určitou rychlostí, dlaždice, jež jsou využity k tvorbě prostředí úrovní.

1.3 Ukázka her

Obrázek 1: Hra MARIO-LIKE

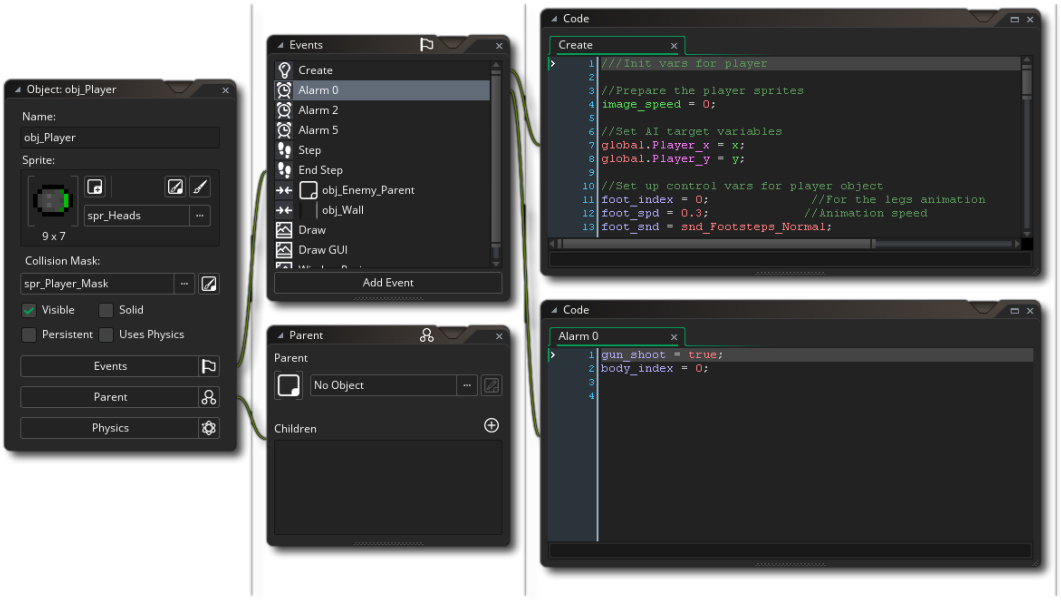


Obrázek 2: Hra KENNEY

1.4 Existující software

1.4.1 GameMaker Studio 2

GameMaker Studio 2 [1] je cross-platformní herní engine vyvíjený firmou YoYo-Games. Jeho jádro je naprogramované v jazyce C++ a editor v C#. Jádro enginu používá pro vykreslování her rozhraní Direct3D na platformě Windows a Xbox One, OpenGL na MacOS a Linux, a OpenGL ES na platformě Android a iOS.

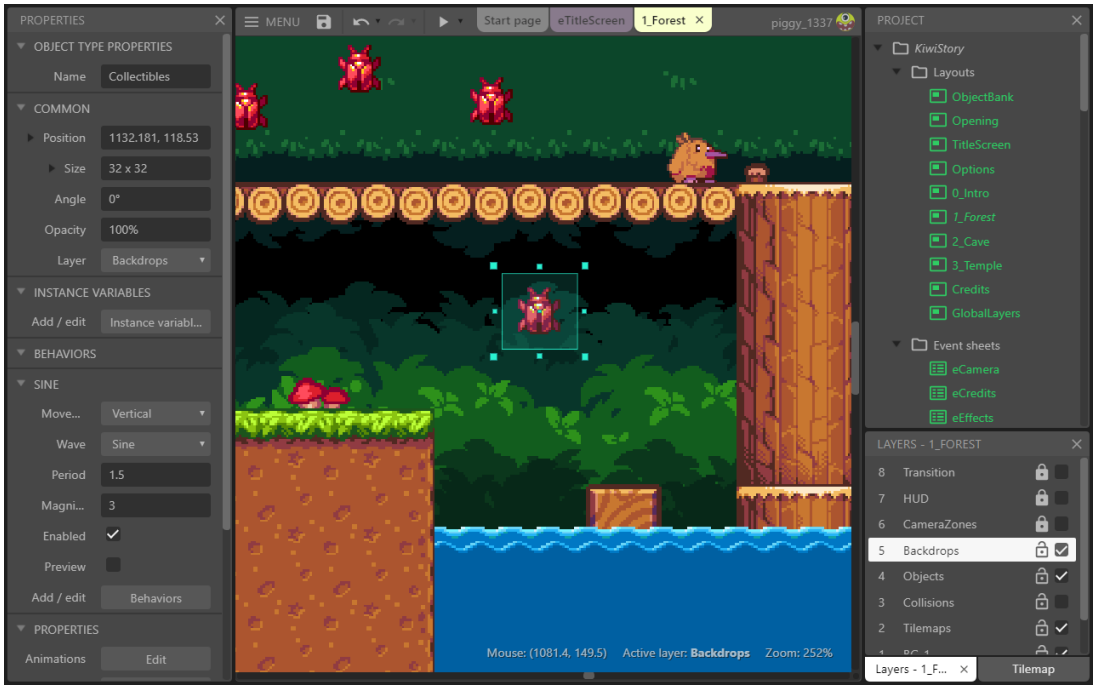


Obrázek 3: GameMaker Studio 2

Základním elementem enginu jsou zabudované editory rastrové grafiky, úrovní, skriptování, cest a shaderů (program sloužicí k řízení jednotlivých části programovatelného grafického řetězce grafické karty). Skriptování je zde v podobě jazyka GameMaker Language. Ten je imperativní, dynamicky typovaný, běžně spojovaný s jazykem JavaScript. Mimo skriptovací jazyk GameMaker Studio 2 disponuje také Drag and Drop vizuálním skriptovacím jazykem, který uživateli dovolí provést běžné akce a reagovat na běžné události. Viz obrázek 3.

1.4.2 Construct 3

Construct 3 [2] herní editor, založený na technologii HTML5, je mířený primárně na neprogramátory. Umožňuje rychlou tvorbu her pomocí systému událostí a Drag and Drop akcí, které slouží jako vizuální skriptovací jazyk.

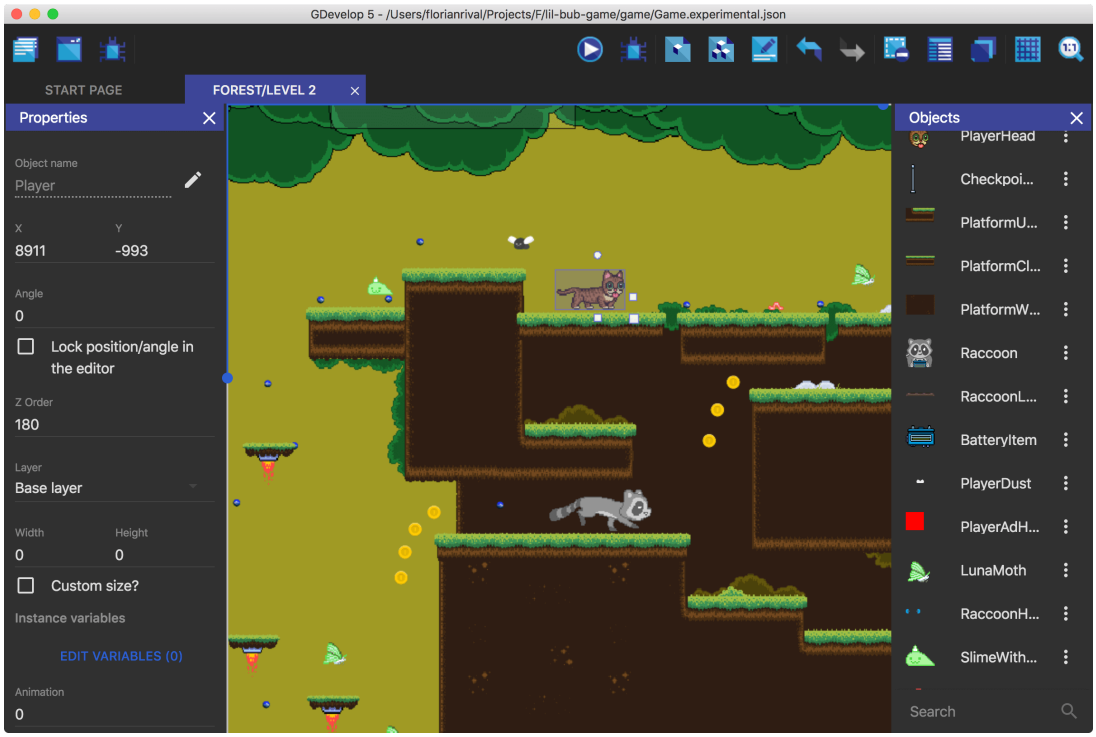


Obrázek 4: Construct 3

Construct 3 pokládá za hlavní cílové platformy prohlížeče schopné technologie HTML5 a mobilní zařízení běžící na operačním systému Android a iOS (pomocí technologie Cordova). Dále umožňuje rychlý export na internetové tržiště jako Facebook Marketplace, Amazon Appstore nebo Chrome Web Store.

1.4.3 GDevelop

GDevelop [3] je open-source, cross-platform herní editor navržený pro rychlou tvorbu her bez potřeby znalostí programování. Umožňuje tvorbu her typu jako jsou plošinové hry, puzzle hry, strategie apod.



Obrázek 5: GDevelop

Pro správu herní logiky používá GDevelop systém událostí. Události zde představují silný nástroj pro vyjádření herní logiky bez potřeby znalosti programování. Mimo jiné také implementuje možnost rozšíření skriptovacím jazykem JavaScript.

1.4.4 Porovnání

Herní engine a hlavně editor her byl inspirován dřívější verzí GameMakeru. Editor používá podobné rozdělení typů zdrojů (animace, zvuky, úrovně, objekty). Systém událostí v herním enginu je také podobný tomu v GameMakeru nebo v GDevelopu s rozdílem, že můj engine nepoužívá vizuálního skriptování, události jsou rovnou řešeny v programovacím jazyce Lua. Další rozdíl je v možnosti jednodušší tvorby uživatelských rozhraní, které si v případě GameMakeru musí řešit každý uživatel sám. Poslední obrovský rozdíl spočívá v tom, že GameMaker a ostatní jsou daleko více obecnější a umožňují tvorbu i her jiných žánrů.

2 Použité technologie

2.1 Visual Studio

Microsoft Visual Studio [4] je vývojové prostředí (IDE) od Microsoftu. Může být použito pro vývoj konzolových aplikací a aplikací s grafickým rozhraním spolu s aplikacemi Windows Forms, webovými stránkami, webovými aplikacemi

a webovými službami jak ve strojovém kódu, tak v řízeném kódu na platformách Microsoft Windows, Windows Mobile, Windows CE, .NET, .NET Compact Framework a Microsoft Silverlight.

2.2 C++

C++ [5] je multiparadigmatický programovací jazyk, který vyvinul Bjarne Stroustrup a další v Bellových laboratořích AT&T jako rozšíření jazyka C. C++ podporuje několik programovacích stylů (paradigmat) jako je procedurální programování, objektově orientované programování a generické programování, není tedy jazykem čistě objektovým. V mé práci jsem využil standardu C++17.

2.3 OpenGL

OpenGL [6] je prostředí pro vývoj přenosných, interaktivních 2D a 3D grafických aplikací. Od svého uvedení v roce 1992, OpenGL se stalo průmyslem nejpoužívanější a podporovalo 2D a 3D programovací rozhraní grafických aplikací (API), přinášející tisíce aplikací do široké škály počitačových platform.

2.4 GLFW

GLFW [7] je multi-platformní open-source knihovna pro vývoj OpenGL, OpenGL ES (verze pro přenositelná zařízení) a Vulkan aplikací na desktopových zařízeních. Nabízí jednoduché rozhraní pro tvorbu okna, kontextu (objekt držící veškerou funkcionalitu) a povrchů (vykreslovacích ploch), příjem vstupů a událostí.

2.5 FMOD

FMOD studio [8] nabízí jednoduchou integraci správy a přehrávání zvuků do jakékoliv aplikace. Využívají ji systémy jako Unity nebo Unreal Engine 4. FMOD nabízí rozhraní v jazycích C, C++ nebo C#.

2.6 Qt

Qt [9] je mnohem více než jen cross-platformní SDK – je to technologie, jež umožnuje rychle a cenově přívětivě navrhovat, vyvíjet a udržovat software při zaručení jednotného uživatelského rozhraní napříč všemi zařízeními. Tuto knihovnu jsem ve své práci využil k tvorbě uživatelského rozhraní aplikace editoru her.

2.7 JSON, Nlohmann

Ve své práci jsem využil knihovny json [12] od Nielse Lohmanna pro parsování json [11] souborů v jazyce C++. Knihovna nabízí intuitivní syntaxi a je jednoduše implementovatelná do jakékoliv aplikace, neboť je tvořena jediným hlavičkovým souborem (.h) a za jediný požadavek pokládá C++11.

2.8 Lua, sol2

Lua [10] je silný, efektivní, lehce zabudovatelný skriptovací jazyk. Podporuje procedurální, objektový, funkcionální a data-driven přístup k programování. V práci jsem ho využil pro rozšíření funkcionality herních objektů a objektů uživatelského rozhraní.

Pro zabudování jazyka Lua do enginu jsem využil knihovny sol2 [13], což je knihovna určená pro lehkou integraci rozhraní jazyka Lua (od verze 5.1+) do C++ aplikací. Knihovnu sol2 jsem si vybral, jelikož nabízí pokročilé funkce a také kvůli jejímu výkonu.

2.9 popl

Popl [14] je analyzátor argumentů C++ obsažený v jednom hlavičkovém souboru. Podporuje jednoduchou implementaci, až na C++11 nemá žádné požadavky, je nezavislý na platformě a je jednoduchý na použití.

2.10 CMake

CMake [15] je open-source, cross-platformní rodina nástrojů navržena pro kompilaci, testování a zabalení projektů do balíčků. CMake ovládá proces kompilace pomocí jednoduchých konfiguračních souborů jež generují nativní makefile soubory a připravují prostředí vlastní volby. V práci jsem ho použil ke kompilaci projektu engine a editoru na platformě Linux.

3 Programátorská příručka

3.1 Koncept herního enginu

Herní engine je primárně určený pro tvorbu tiled-based plošinových her (her, kde je vše umístěné do mřížky pevné velikosti). Zároveň je herní engine navržený dostatečně obecně, aby v něm šli vyvýjet i hry jiného typu, ale také poskytuje natolik jednoduché rozhraní, které umožní vytvořit jednoduchou hru i během deseti minut.

Každý herní projekt se skládá z dvou typů objektů – ze **zdrojových souborů**, které reprezentují vizuální stránku hry, a z **logických objektů**, které zdrojovým souborům přidávají význam.

Mezi zdrojové soubory patří textury (obrázky), True Type písma a soubory zvuku. Kromě zdrojových souborů, které do projektu přidá uživatel, každý projekt obsahuje základní soubory, které se použijí v případě chybějícího zdrojového souboru. Logické objekty jsou tvořeny animacemi, dlaždicemi, uživatelskými rozhraními, kolizemi, úrovněmi a herními objekty.

Prvním takovým objektem je objekt třídy `Sprite` – **animace**. Animace se skládá z uspořádaného seznamu textur, které reprezentují dílčí snímky animace. Dalším objektem jsou **dlaždice** (třída `Tileset`), jehož prvky slouží k návrhu vizuální stránky úrovně. Tento objekt využívá jedné textury reprezentující dvourozměrné pole, z něhož každý prvek je o stejné velikosti (ta je nastavena při vytváření projektu) a může být dosazen do mřížky v úrovních. Pro představu je ukázka textury dlaždice uvnitř dialogu úpravy dlaždic je na obrázku 9.

Každý herní projekt může definovat vícero **uživatelských rozhraní** – v daný okamžik je vždy maximálně jedno rozhraní aktivní. Uživatel sám určuje přechody mezi různými rozhranimi. Každé uživatelské rozhraní se skládá z jeho prvků, s kterými může uživatel interagovat.

Úrovně (třída `Level`) obsahují seznam herních objektů spolu s jejich pozicí a dvě dvourozměrná pole – jedno pro dlaždice, jež reprezentují vykreslení dané úrovně, a druhé pro kolize, které dlaždicím přiřazují význam. To vede k rychlému návrhu úrovně skládáním prvků dlaždic a kolizí jako puzzle. Poslední logickým objektem je **herní objekt** (třída `GameObject`). Tyto objekty se chovají jako aktéři v úrovních, využívají animací pro vykreslení, zvuků pro ozvučení. Každý herní objekt implementuje pouze základní logiku a uživatel je nucen naprogramovat jejich chování pomocí programovacího jazyka Lua a jeho rozhraní s enginem.

Hlavním aktérem herního enginu je systém událostí a jejich reakci pomocí skriptů programovacího jazyka Lua. Pro každý herní projekt je nutno definovat vlastní herní logiku a tedy se žádný herní projekt neobejde bez základních znalostí programování.

3.2 Popis tříd herní enginu

Tato podkapitola se věnuje popisu významných tříd projektu herního enginu.

3.2.1 Třída Application

Třída `Application` je třída, jež se stará o vytvoření kontextu a okna aplikace na dané platformě. Stará se o příjem a předání vstupních/výstupních událostí objektům tříd jako `Renderer` nebo `InputSystem`.

3.2.2 Třída Game

Třída `Game` se stará o správu samotné hry načtené ze souboru projektu. Aktualizuje současnou úroveň spolu s uživatelským rozhraním. Kromě toho propojuje veškerou funkcionalitu herního enginu a dodává ji do rozhraní skriptovacího jazyka Lua.

Zároveň tato třída slouží jako globální objekt, který je všudepřítomný od spuštění hry až po její ukončení. Dále obsahuje globální prostředí vazeb a definuje globální reakce na události, které se netýkají ani uživatelských rozhraní, jejich prvků ani herních objektů.

3.2.3 Třída Renderer

Abstraktní třída `Renderer` představuje rozhraní pro vykreslovaní herního enginu. Tato třída umožňuje nastavit aktuálně vykreslovanou úrovneň, uživatelské rozhraní, ohraničení pozice kamery, ztmavení obrazovky a rozměr okna. Dále definuje rozhraní pro předání zdrojových souborů textur a true type písma. V nynejší podobě jej implementuje pouze třída `GLRenderer`, jež reprezentuje vykreslování pomocí rozhraní OpenGL.

3.2.4 Třída SoundSystem

`SoundSystem` je třída, která hernímu enginu umožňuje spravovat a přehrávat zvuky a hudbu za pomoci knihovny FMOD. Definuje jednoduché rozhraní pro předání zdrojových souborů zvuků, nastavení hlasitostí a samotné přehrávání.

3.2.5 Třída InputSystem

Tato třída se stará o správu vstupů uživatele z klávesnice, myši a nebo joysticku. Umožňuje zjistit informace o právě používaném vstupním zařízení. Dále umožňuje zaznamenat, jaký text uživatel na klávesnici píše. K dosažení této funkcionality je využita knihovna GLFW.

3.2.6 Třídy herních objektů

Prototypy objektů, které uživatel definuje v editoru, jsou reprezentovány pomocí třídy `GameObjectPrototype`. Ta slouží k uložení společných dat objektů daného typu, uchování bytekódu reakcí na události a jednoduché vytvoření herního objektu podle daného prototypu.

Samotné herní objekty jsou reprezentovány třídou `GameObject`, která definuje pouze základní vlastnosti, jež se dají od herního objektu očekávat (název, rozměry, masku kolize, rychlost a směr, výchozí animaci, jeho lokální prostředí, kolizní systém, systém událostí a dědičnosti).

Hlavní roli zde hraje systém události, kterým může uživatel pomocí programovacího jazyka Lua nadefinovat reakci na jejich vyvolání. Každá událost má svůj vlastní význam, např. událost `PŘI VYTVOŘENÍ` je využita pro nastavení lokálního prostředí a událost `PŘI ZNIČENÍ` slouží k jejímu vyčištění. Významnými událostmi pak jsou `PŘI AKTUALIZACI`, ve které se definuje veškerá logika, `PŘI VYKRESLENÍ`, kde může uživatel dokreslit pro daný snímek co potřebuje a událost `KOLIZE`. Těch může definovat uživatel právě tolik, kolik prototypů herních objektů má definovaných, každý jedenkrát. Poslední události jsou `PŘI POČÁTKU ÚROVNĚ` a `PŘI KONCI ÚROVNĚ`, které reagují na změnu úrovní.

Při vytvoření nového objektu je vytvořené nové prostředí vazeb, které dědí z globálního, nastaví se pozice objektu a dojde k zavolání události `PŘI VYTVOŘENÍ`. Naopak při zničení je zavolána událost `PŘI ZNIČENÍ`.

Každý prototyp herního objektu může definovat jednoho předka (tzn. objekty tvoří strom dědičnosti, který je využitý při volání událostí kolize). Tedy při kolizi nějakého herního objektu s jiným nedojde pouze k vyvolání reakce na událost kolize s objektem daného typu, ale také se všemi typy jeho předků (postupně od listů stromu dědičnosti ke kořenu). Toto chování je možné v jakémkoliv místě přerušit pomocí příkazu `this.event_dispatch()` (o tom více v kapitole B.2).

3.2.7 Třída Layout a její prvky

Třída `Layout` představuje uživatelské rozhraní, jeho rozložení a jeho prvky. Vždy v daném okamžiku je aktivní pouze jedno uživatelské rozhraní. Navíc každý objekt třídy Layout implementuje základní ovládání jeho interaktivních prvků (tlačítka, vstupní pole) pomocí klávesnici a myši (případně joysticku).

Všechny prvky uživatelského rozhraní vychází ze třídy `UIObject`. Ta definuje základní logiku a spravuje události spojené s uživatelským rozhraním a jeho ovládání. Další třídy jsou `UILabel`, `UIImage`, `UIButton` a `UITextBox`. Každá z těchto tříd je spjatá s určitým typem prvku.

Stejně jako herní objekty i uživatelská rozhraní spolu s jejími prvky spravují systém událostí velmi podobný tomu u herních objektů. Místo události vytvoření a zničení jsou dostupné události `PŘI ZOBRAZENÍ` a `PŘI SCHOVÁNÍ`, které se vyvolají při změně uživatelských rozhraní. Prvky typu tlačítka a textového pole navíc definuje události `KLIKNUTÍ NA PRVEK`, `VÝBĚR PRVKU` a `ZRUŠENÍ VÝBĚRU PRVKU`, které jsou spojené se vstupem uživatele.

3.2.8 Třída Level

Jednotlivé úrovně reprezentuje třída `Level`. Ta definuje dvourozměrné pole o rozměrech úrovně pro mřížku dlaždic, které tvoří pozadí. Objekty i mřížku dlaždic je

v úrovni možné destruktivně měnit (i během průběhu hry), avšak při restartování úrovně se vždy vrátí do jejich původní podoby.

3.2.9 Třídy Scripting a EventHandle

Třída `Scripting` reprezentuje rozhraní mezi hrou, herními objekty a objekty uživatelských rozhraní s knihovnou sol2. Implementuje logiku nutnou pro vytvoření a dědičnost prostředí všech logických objektů a stará se o kompilaci skriptů, které tvoří reakce na události.

Reakce na události jsou reprezentovány třídou `EventHandle`, která kromě bytekódu skriptu také uchovává základní informace o vlastníku reakce

3.3 Editor

Tato podkapitola se věnuje popisu významných tříd projektu editoru her.

3.3.1 Třída Editor

Třída Editor představuje hlavní dialogové okno aplikace editoru. Více o rozložení prostředí editoru v kapitole 4.3.

3.3.2 Třída LevelEditor

Třída LevelEditor, která dědí z třídy QDialog, představuje dialog návrhu úrovní. Na levé straně obsahuje panel s třemi lištami, každou pro typ objektu, jež lze dosadit do mřížky, a na pravé straně samotný panel vykreslující danou úroveň. Více o rozložení návrháři úrovní v kapitole 4.3.8.

3.3.3 Třída Project

Třída Project se stará o vytvoření, načtení a správu herního projektu. Dále zprostředkovává jednotlivým editorům odkazy na objekty, maže nebo nové vytváří.

3.4 Sdílené soubory

Tato podkapitola se věnuje popisu souborů a tříd sdílených napříč projektu enginu a editoru.

3.4.1 Třída Gamefile

Tato třída slouží k načítání a ukládání herního projektu do souboru `.gamefile`. Hlavničkový soubor obsahuje definice struktur všech zdrojových objektů, objektů uživatelského rozhraní, herních objektů a úrovní.

3.5 Kompilace

Na platformě Microsoft Windows je kompilace jednoduchá. První podmínkou je mít nainstalované Visual Studio 2017 s kompilátorem verze 141. Dále je nutné spustit instalátory knihovny Qt verze 5.9, plugin Qt pro Visual Studio a FMOD studio. Všechny instalátory jsou dostupné v adresáři /install/windows. Nakonec je potřeba po otevření souboru projektu na cestě /src/solution.sln stisknout v navigačním pruhu tlačítko *sestavit*. (popř. kombinací tlačítek klávesnice CTRL a F5).

U platformy Linux se nejdříve musí splnit veškeré závislosti – ty jsou popsány uvnitř souboru dependencies.txt, který se nachází ve složce /install. Dále je potřeba vygenerovat makefile soubor spuštěním příkazu `cmake .` uvnitř složky src, poté už jen spustit příkaz `make` uvnitř složky /src/engine a /src/editor, jež vytvoří spustitelné soubory.

Tyto soubory je potřeba přesunout do správné složky – spustitelný soubor editoru do složky /bin a spustitelný soubor engine do složky /bin/project-exports/`<cílená-platforma>` spolu s dodatečnými soubory knihoven (nutné u platformy Windows).

Nakonec upozorňuji, že práce byla vyvíjena na platformě Windows, možnost kompilace na platformu Linux je spíš experimentální.

3.6 Možná rozšíření

Herní engine má připravené rozhraní pro rozšíření o vykreslovaní pomocí jiné knihovny (např. Vulkan). Herní subsystémy jako správa zvuků jsou rovněž navrženy tak, aby se daly rozšířit nebo nahradit.

Co se týče samotného chodu hry, provedení úrovní by se dalo rozšířit i o způsob kolizí, jež nejsou v mřížce, nebo rozšířit herní engine, aby měl uživatel větší kontrolu nad všemi vnitřními procesy.

4 Uživatelská příručka

4.1 Systémové požadavky

U platformy Microsoft Windows jsou pouze dva požadavky na spuštění aplikace:

1. Windows 7 nebo novější
2. Balíček Visual C++ Redistributable for Visual Studio 2017, jež je dostupný v podobě instalátoru uvnitř složky /install (potřeba nainstalovat x86 i x64).

Platforma Linux vyžaduje splnění závislostí – postup je k dispozici ve složce /install v podobě souboru dependencies.txt.

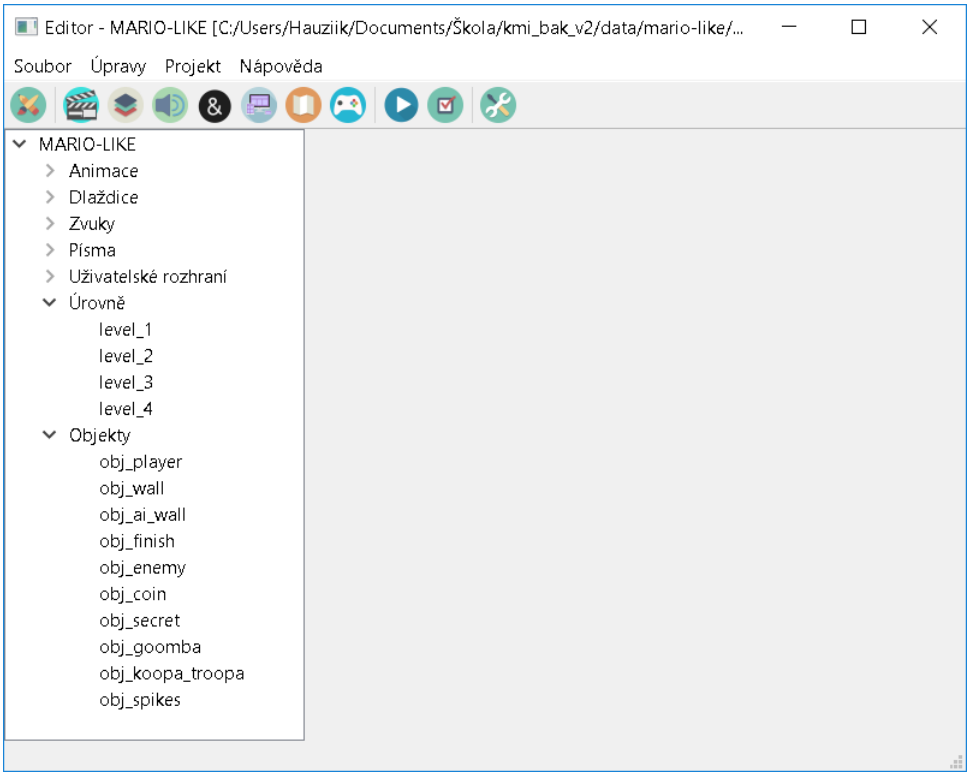
4.2 Instalace a odinstalace

Pro instalaci stačí přetáhnout nebo zkopírovat složku /bin na zvolené místo. Spuštění aplikace editoru nebo samotných her je podmíněné splněním systémových požadavků popsaných v kapitole 4.1. K odinstalaci stačí vytvořenou složku smazat, neboť kromě složek vytvořených pro projekt aplikace editoru žádný obsah nevytváří ani neukládá.

Pro správné použití programu editoru je nutno přesunou všechny soubory binárních souborů programu editoru ze složky /bin/editor na pevný disk, jinak není funkcionalita exportování projektu na všech systémech zaručena.

4.3 Práce s editorem

V následující podkapitole seznámím uživatele s postupy při práci v každém dialogovém okně herního editoru. Dále nastíním základní orientaci v rozhraní editoru.



Obrázek 6: Uživatelské rozhraní editoru

V horní části okna editoru se nachází panel, který v horní části obsahuje navigační pruh a ve spodní části panel rychlých akcí. Pod položkou `Soubor` se nachází základní operace se souborem projektu jako je vytvoření nového projektu, jeho uložení a načtení, export projektu do spustitelné podoby a ukončení aplikace editoru. Další položkou jsou úpravy. Ty umožnují spravovat textury a také přidávat animace, dlaždice, zvuky, písma, vytvářet objekty uživatelského rozhraní, herní objekty a úrovně. Poslední položkou navigačního pruhu je projekt, který nabízí spuštění aktuálního projektu a odkaz na dialog nastavení projektu.

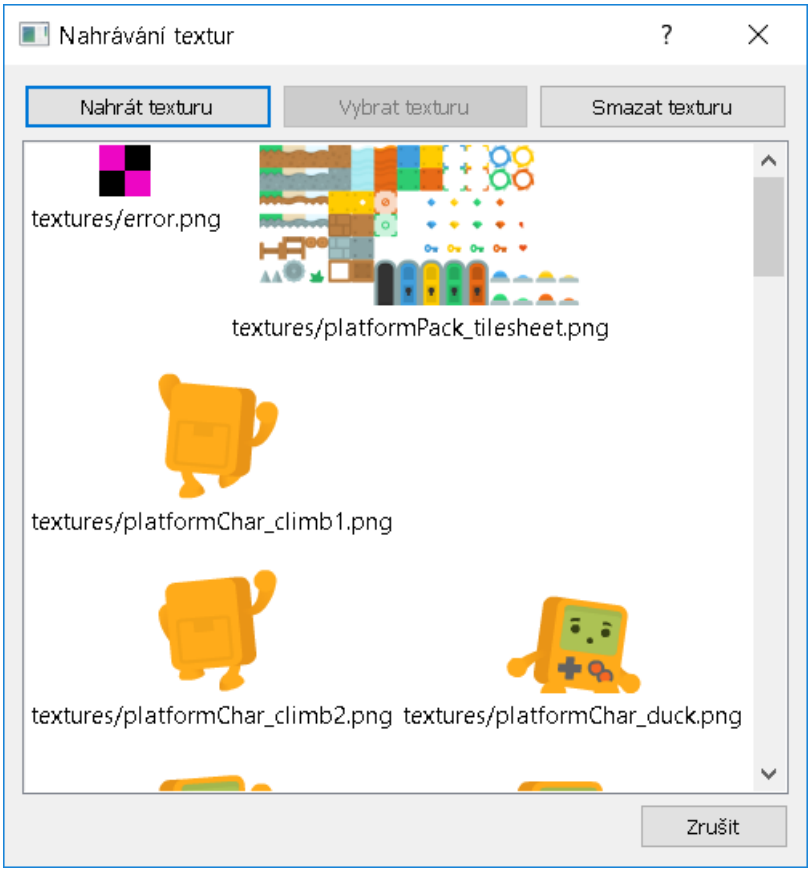
Levou stranu editoru zabírá projektové okno, které obsahuje stromovou strukturu zobrazující seznam všech dostupných zdrojových souborů a objektů.

Titulek aplikace vždy obsahuje cestu k právě otevřenému projektu spolu s indikátorem, zda byly změny v projektu uloženy.

Každé dialogové okno určené k úpravě některé entity vždy obsahuje tlačítka pro uložení objektu, jeho smazání a tlačítko zrušení určené k zahození změn. Pokud uživatel upraví daný objekt a neuloží jej, tak se vždy objeví varování.

4.3.1 Správa textur

Prvním důležitým dialogovým oknem je správce textur. Ten můžeme otevřít buď přes první položku (zleva) v hlavním navigačním pruhu, nebo pomocí ikony v panelu rychlého přístupu. V tomto okně nalezeneme seznam textur, jež jsou zahrnuty v daném projektu.

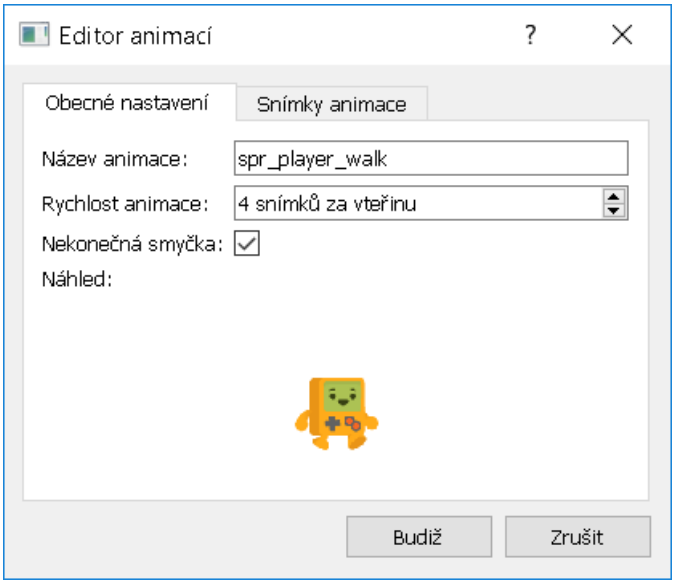


Obrázek 7: Správce textur

Dále můžeme načíst texturu novou nebo smazat texturu, jež byla označena kliknutím levého tlačítka myši. Seznam obsahuje i texturu *error.png*, jež je výchozí texturou pro objekty, jimž nebyla žádná textura přiřazena, a kterou nelze smazat.

4.3.2 Tvorba animací

Pro vytvoření nové animace musí uživatel stisknout tlačítko *přidat animaci*, jež se nachází pod položkou upravit v hlavním navigačním pruhu. To otevře *dialog úpravy zvolené (případně nové) animace*, které obsahuje dvě lišty – *obecné nastavení* a *snímky animace*. Rozložení dialogu je k dispozici na obrázku 8.



Obrázek 8: Dialog úpravy animace

Pod lištou obecného nastavení může uživatel nastavit název animace, rychlost přehrávání animace a jestli se má animace opakovat v nekonečné smyčce. Dále je zde zobrazen náhled prvního snímku pro rychlou orientaci.

Druhá lišta obsahuje seznam textur jak jdou po sobě, tlačítka pro přidání nové textury a smazání textury vybrané a tlačítka pro přesun textury doleva, příp. doprava.

4.3.3 Nastavení dlaždic

V dialogu nastavení dlaždic (obrázek 9) mimo název dlaždice pouze specifikujeme, kterou texturu daná dlaždice bude využívat. Tak učiníme kliknutím na tlačítko `nastavit texturu`, kde buď vybereme již načtenou texturu, nebo načteme novou texturu.

Obrázek 9: Nastavení dlaždic

4.3.4 Přidání hudby a zvuků

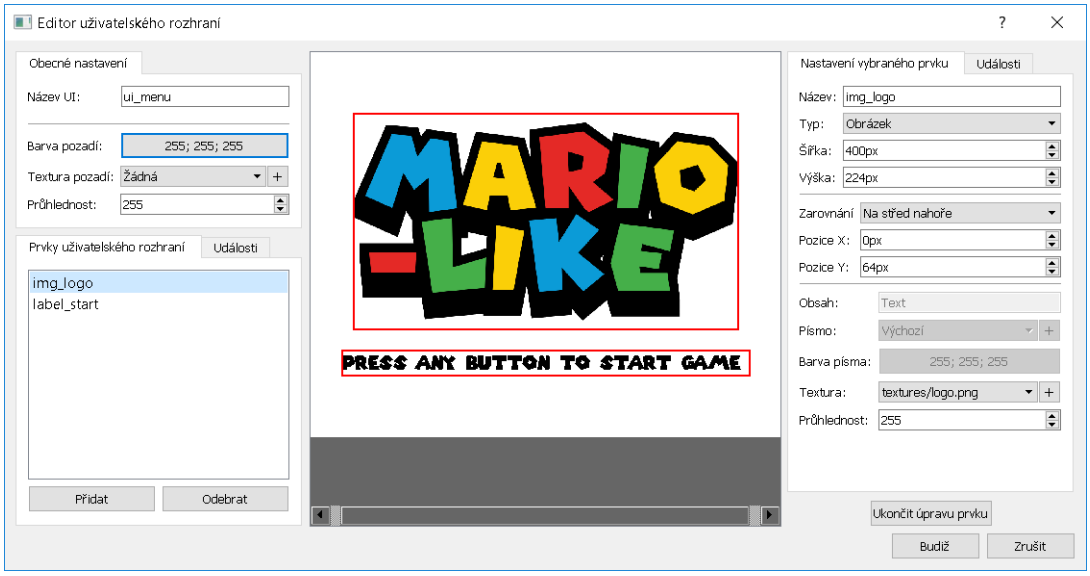
Editor hudby a zvuků funguje podobně jak editor dlaždic. Uvnitř dialogu úpravy zvuku můžeme nastavit název zvuku/hudby, jeho zdroj a poté typ, zda to je zvuk nebo hudba. Jediný rozdíl mezi zvukem a hudbou je ten, že hudba se spustí v nekonečné smyčce a zastaví se pouze při přechodu mezi úrovněmi, uživatelskými rozhraními, nebo, pokud tak určíme, pomocí funkcí ve skriptu rozšíření některého objektu.

4.3.5 Přidání písma

Herní engine umožňuje tvorbu uživatelských rozhraní spolu s vykreslováním textů pomocí true type fontů. Tyto fonty může uživatel načítat a spravovat v nabídce úpravy písma nebo přidávat přes odkaz nového písma v navigačním pruhu.

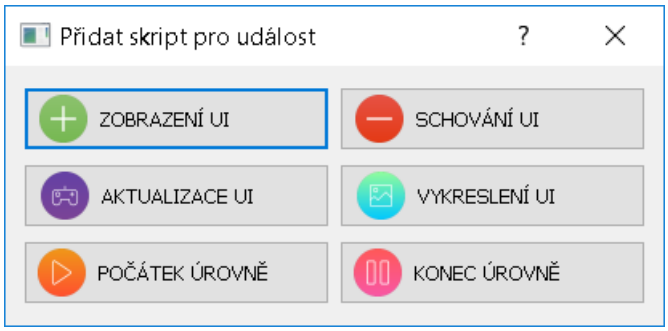
4.3.6 Tvorba uživatelského rozhraní

Přidání nového uživatelského rozhraní se dá provést přes tlačítko `Přidat uživatelské rozhraní` pod položkou upravit v hlavním navigačním pruhu. To otevře nabídku úpravy uživatelského rozhraní, jež obsahuje tři lišty.



Obrázek 10: Úprava uživatelského rozhraní

Okno úpravy uživatelského rozhraní je rozděleno na tři části. První část zleva se věnuje nastavení uživatelského rozhraní jako celku. Zde se zadávají parametry jako název uživatelského rozhraní, dále barva, textura a průhlednost pozadí. Pod formulářem se nachází dva seznamy, každý ve vlastní liště. První seznam obsahuje výčet prvků daného uživatelského rozhraní. Úprava jednotlivých prvků je dostupná pomocí dvojkliku. Druhá lišta obsahuje seznam událostí daného uživatelského rozhraní (ukázka možností přidání událostí na obrázku 11).

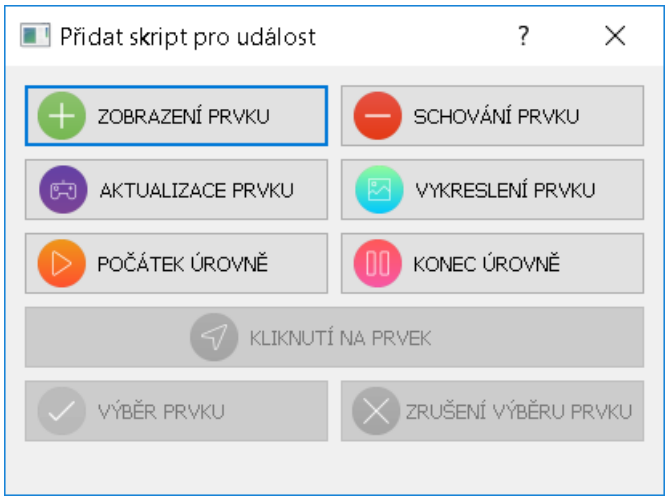


Obrázek 11: Přidání události pro uživatelské rozhraní

Druhá část okna je věnována rychlému náhledu daného uživatelského rozhraní. Toto plátno slouží pouze k zobrazení aktuálního rozložení uživatelského rozhraní a jeho prvků a aktualizuje se při každé úpravě ať už prvku, nebo celého uživatelského rozhraní. Plátno není interaktivní.

Poslední část okna je zobrazena pouze tehdy, pokud je právě upravován některý z prvků daného uživatelského rozhraní. Pokud je tato sekce aktivní, pak se v ní nachází dvě lišty – první obsahuje formulář parametrů zvoleného prvku daného uživatelského rozhraní a druhá lišta obsahuje seznam událostí zvoleného

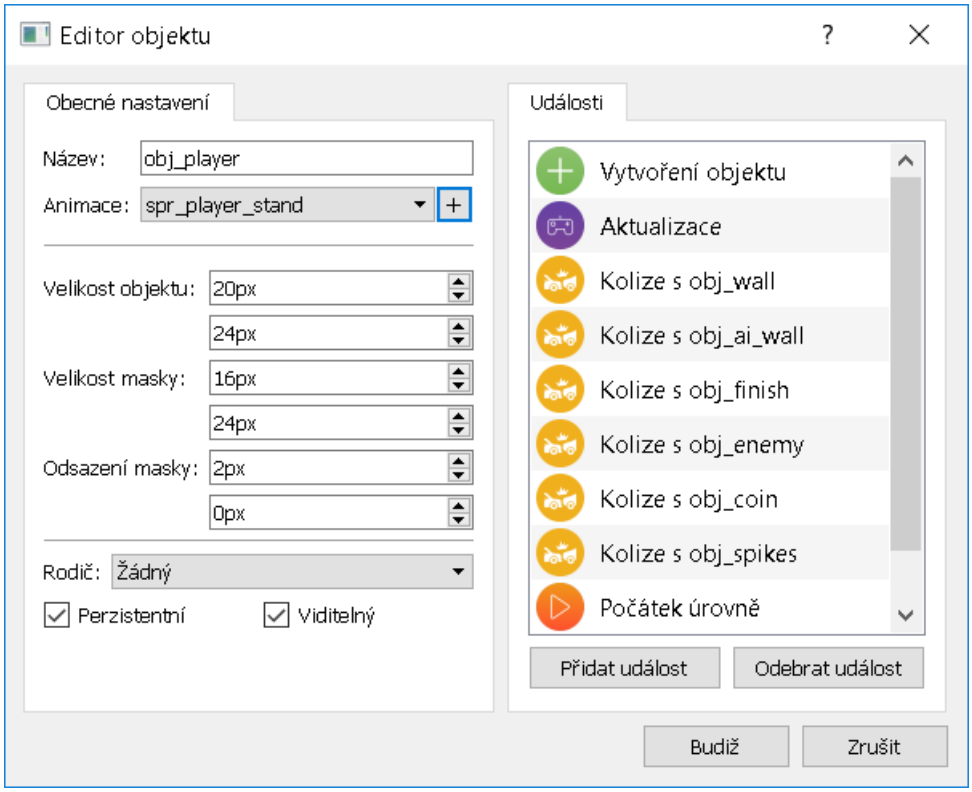
prvku. Mezi parametry prvku patří jeho název, typ, velikost a jeho relativní pozice vůči zarovnání. Další parametry jsou proměnlivé podle typu prvku – např. pro typ `text`, `tlačítko` a `textové pole` je dostupný parametr textový obsah, pro jiné parametr textura pozadí. Ukázka přidání události prvku uživatelského rozhraní je k dispozici na obrázku 12.



Obrázek 12: Přidání události pro prvek uživatelského rozhraní

4.3.7 Přidání a úprava objektů

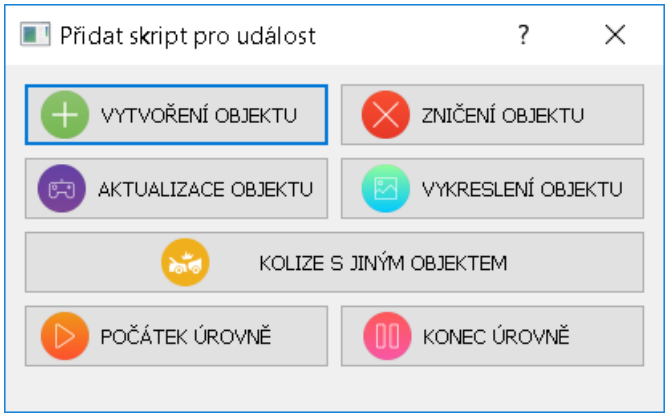
Přidání nového herního objektu se dá provést přes tlačítko přidat objekt pod položkou upravit v hlavním navigačním pruhu, které otevře dialog úpravy objektu, jež je rozdělené na dvě části.



Obrázek 13: Úprava herního objektu

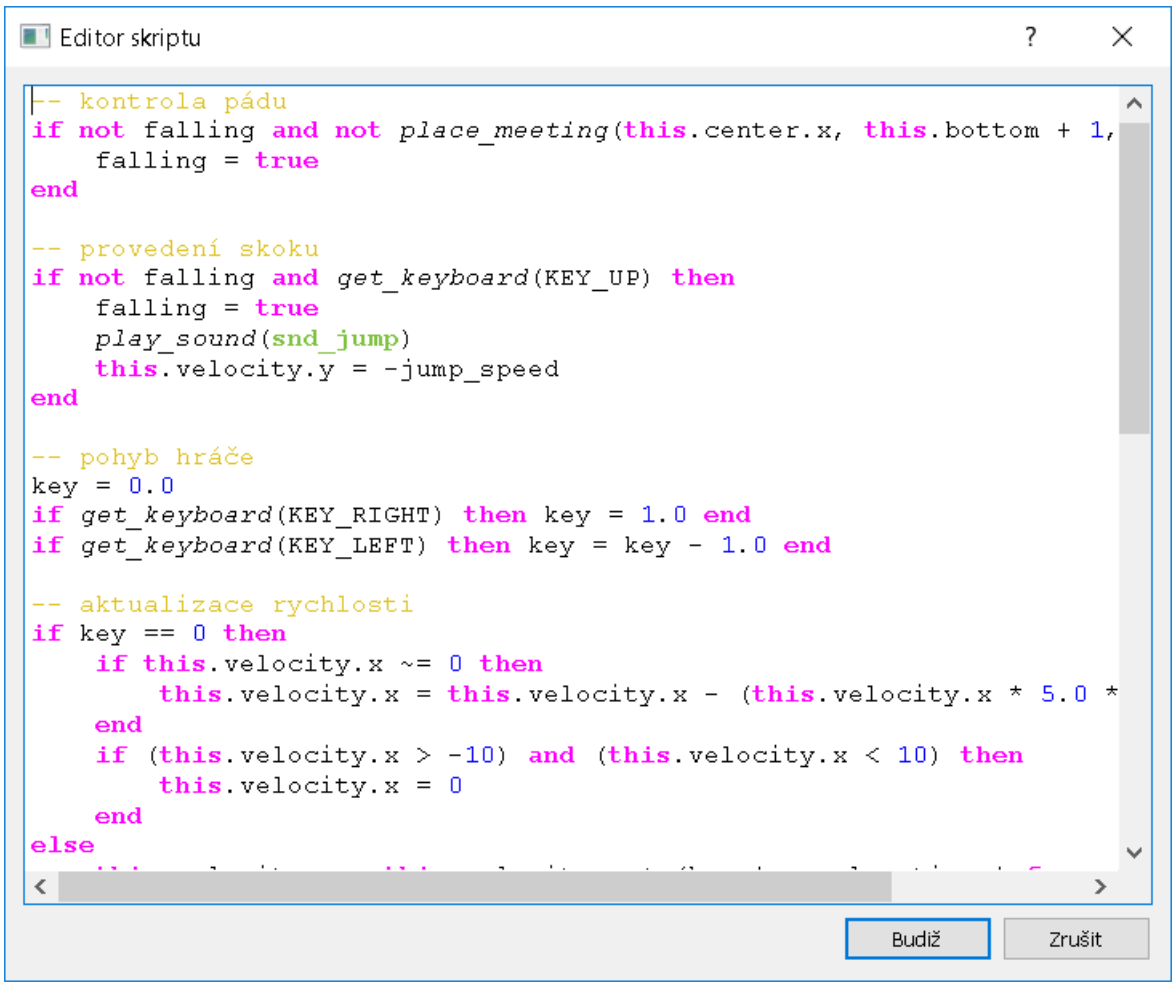
První část zleva obsahuje jednoduchý formulář, pomocí kterého se nastaví základní vlastnosti daného objektu jako jeho název, jeho výchozí animace, velikost objektu a jeho masky (případně odsazení). Dále zde je možnost učinit objekt perzistentním a nastavit jeho viditelnost. Poslední možností je určit jeho rodiče.

Druhá část je věnována seznamu událostí daného objektu, které je možné přidat pomocí tlačítka přidat, smazat vybráním události a stlačením tlačítka smazat, nebo upravit pomocí dvojkliku na zvolenou událost. Seznam událostí herních objektů je k dispozici na obrázku 14.



Obrázek 14: Úprava objektu hráče

Příklad skriptu události je dostupný na obrázku 15.

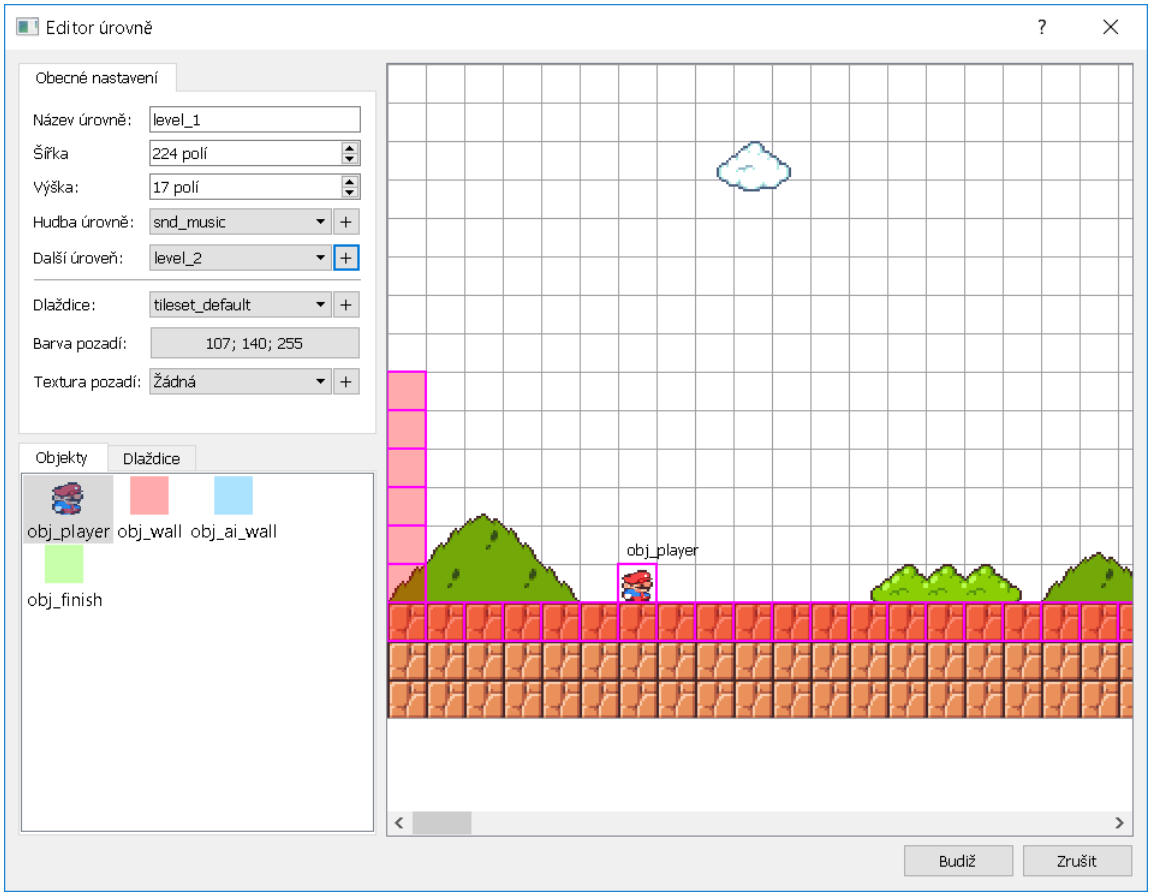


Obrázek 15: Rozšíření objektu hráče

4.3.8 Vytvoření a návrh úrovní

Odkaz na přidání úrovně se nachází pod položkou upravit v hlavním navigačním pruhu. Pokud chceme upravit stávající úroveň, musíme ji nalézt ve struktuře projektu a dvojklikem myši otevřít.

Dialog úpravy úrovní (obrázek 16) je rozdělen na dvě části. První umožnuje základní nastavení úrovně, tzn. její název, dlaždice, které bude využívat, počet políček v svislé, popř. vodorovné ose, hudba na pozadí a nastavení následující úrovně. Dále se zde nachází dvě tabulky. První obsahuje seznam herních objektů, které byly definovány v editoru a druhá zobrazí pole dlaždic, jež slouží k návrhu pozadí herní úrovně. Druhá část je věnována samotnému návrhu úrovně.



Obrázek 16: Návrhář úrovně

Ovládání v editoru úrovní je jednoduché a intuitivní. Pro přidání dlaždice, objektu nebo kolize stačí daný prvek vybrat kliknutím na něj levým tlačítkem myši a posléze kliknout znovu levým tlačítkem myši na místo v úrovni. Pro odebrání prvku stačí najet na místo prvku a kliknout na pravé tlačítko myši. V případě dlaždic a kolizí je možné takto přidávat a odebírat více prvků najednou držením vhodného tlačítka myši a posunem na dané místo v úrovni. Navíc pro ještě lehčí manipulaci je možné přidávat všechny prvky opakovaně podle svislé nebo vodorovné osy pomocí kombinace stisknutí vhodného tlačítka myši a klávesy `ctrl` pro svislou osu nebo klávesy `shift` pro osu vodorovnou. Jedinou podmínkou přidávání a odebírání prvků je být ve správné liště, když chcete pracovat s prvkem daného typu (např. pro přidání/odebrání objektu musí uživatel mít aktivní lištu s objekty).

4.4 Postupy tvorby hry

4.4.1 Vytvoření projektu

Nový projekt lze založit kliknutím na tlačítko nový projekt pod položkou soubor v navigačním pruhu. Poté vyskočí dialogové okno výběru složky, pomocí

které stanovíte, kam se má projekt spolu s daty ukládat. Pokud možno vytvořte složku novou nebo již existující vyprázdněte a ujistěte se, že máte do dané složky právo zápisu. Po výběru se do složky nakopírují elementární soubory potřebné k činnosti enginu spolu se souborem projektu.

Úspěšné vytvoření projektu je ukončeno otevřením nabídky nastavení projektu, kde jste vyzváni k jeho prvotnímu nastavení (viz kapitola 4.4.3)

4.4.2 Načítání a ukládání

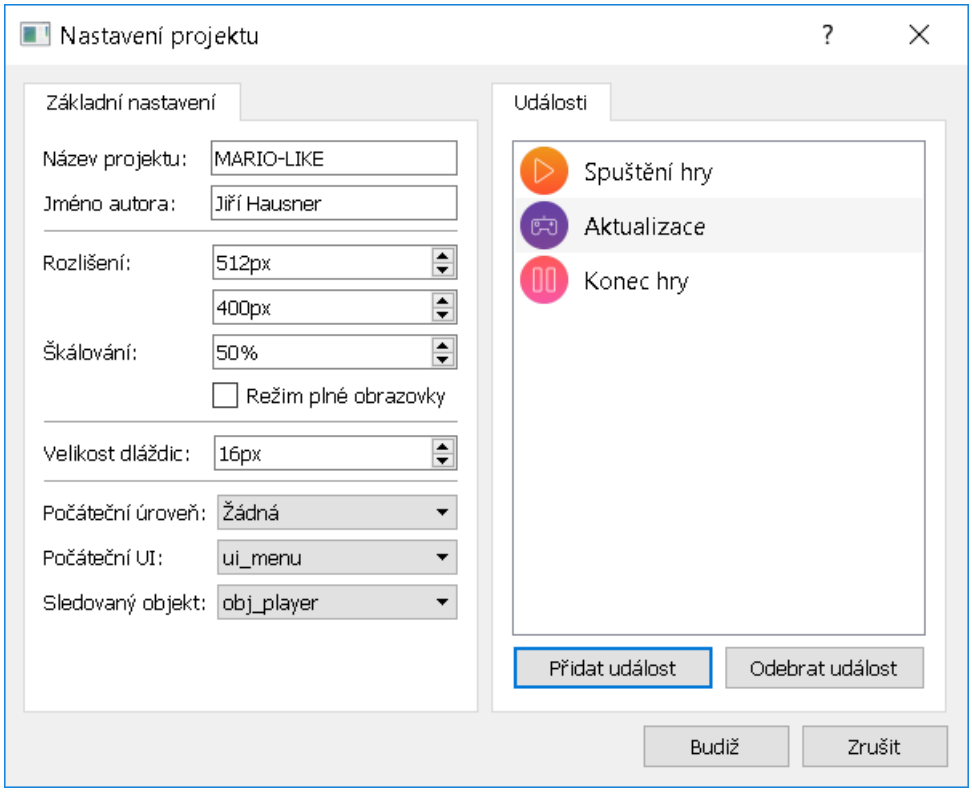
Načítání i ukládání herního projektu je možné uskutečnit zvolením vhodné položky pod oddělením souboru v hlavním navigačním pruhu. V případě načtení se otevře dialog, který vyžaduje výběr souboru typu .gamedata. Při ukládání má uživatel možnost projekt pouze uložit nebo jej uložit do jíného místa pomocí položky *uložit jako*.

4.4.3 Základní nastavení

Nabídka nastavení hry se vždy zobrazí při tvorbě nového projektu (omezená) nebo při kliknutí na tlačítko globální nastavení v hlavním nabídkovém pruhu. Tento dialog je rozdělen na dvě částí.

První obsahuje základní nastavení hry, jež je nutné vyplnit pro možnost spuštění nebo exportu projektu. Druhá část obsahuje seznam událostí, které jsou svázané s objektem hry. V tomto okně se nastavuje a pracuje s globálním prostředím skriptovacího jazyku Lua.

Počáteční úroveň, uživatelské rozhraní ani sledovaný objekt nemusí být zvolené, avšak ke správnému chodu hry by tak mělo být učiněno. Jejich zvolení je možné učinit zavoláním funkce z jakéhokoliv skriptu jakékoliv události.



Obrázek 17: Nastavení projektu

Následujícím krokem uživatele by mělo být nahrání všech potřebných zdrojů, které má v plánu v nové hře používat, přidání dlaždic, zvuků a souborů písma, přípravu objektů uživatelského rozhraní, herních objektů a úrovní.

4.4.4 Přehled událostí

Tabulka 1: Tabulka událostí

Název události	Popis
SPUŠTĚNÍ HRY	Událost spuštění hry (vyvolána i při restartování).
UKONČENÍ HRY	Událost ukončení hry.
VYTVOŘENÍ OBJEKTU	Událost herních objektů, která se vyvolá při jejich vytvoření.
ODSTRANĚNÍ OBJEKTU	Událost herních objektů, která se vyvolá při jejich odstranění.
ZOBRAZENÍ	Událost uživatelského rozhraní a jeho prvků, která se vyvolá při zobrazení uživatelského rozhraní.
SCHOVÁNÍ	Událost uživatelského rozhraní a jeho prvků, která se vyvolá při schování uživatelského rozhraní.

Název události	Popis
AKTUALIZACE	Událost, která se vyvolá pro každý objekt každý při počátku nového snímku aplikace.
VYKRESLENÍ	Událost určená pro vykreslování. Vyvolá se po vykreslení daného objektu. Vykreslení objektu a hry využívá absolutní pozice a vykreslení uživatelského rozhraní a jeho prvků využívá relativní pozice vůči pohledu.
KOLIZE	Událost herních objektů, která se vyvolá při kolizi s danným objektem, který je s událostí spjatý (označený jako **other**).
POČÁTEK ÚROVNĚ	Událost, která se vyvolá při zobrazení nové úrovně.
KONEC ÚROVNĚ	Událost, která se vyvolá na konci úrovně před změnou na další.
VÝBĚR PRVKU	Událost prvku uživatelského rozhraní, která se vyvolá při výběru danného prvku (prvek musí být typu tlačítka nebo textového pole).
ZRUŠENÍ VÝBĚRU PRVKU	Událost prvku uživatelského rozhraní, která se vyvolá při zrušení výběru danného prvku (prvek musí být typu tlačítka nebo textového pole).
KLIKNUTÍ NA PRVEK	Událost prvku uživatelského rozhraní, která se vyvolá při kliknutí na daný prvek (prvek musí být typu tlačítka nebo textového pole).

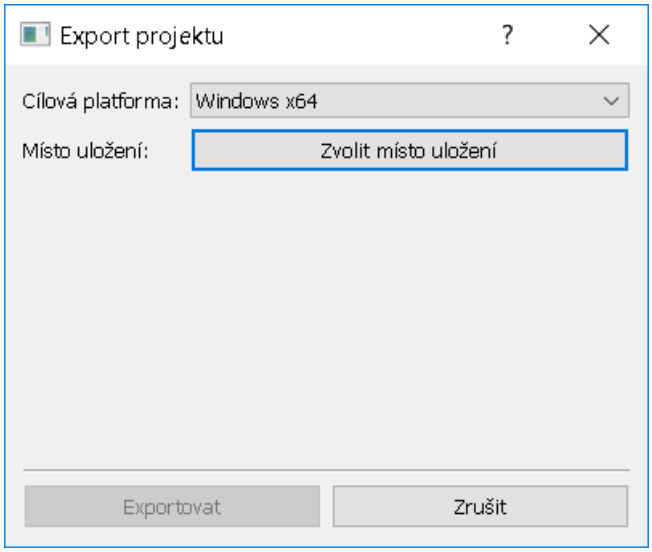
4.4.5 Vlastní testování hry

V případě, že herní projekt už má připravenou grafiku, ozvučení, vytvořené herní objekty a uživatelské rozhraní, může uživatel otestovat hru pomocí tlačítka spustit, které se nachází v hlavním nabídkovém pruhu pod položkou projekt. Pokud projekt nebyl uložen od posledních změn, editor se otáže, jestli chcete spustit neuloženou verzi, nebo projekt uložit a spustit verzi aktuální.

Při úspěšném spuštění enginu nad daným projektem se zároveň spustí vývojářská konzole obsahující výstup logu a chybových hlášek aplikace engine spolu s hláškami vyvolanými z prostředí skriptovacího jazyka Lua použitého v rozšíření objektů.

4.4.6 Dokončení projektu a jeho export

Jakmile projekt splňuje dvě základní podmínky, a to nastavení hry a navrhnutí počáteční úrovně, může uživatel vygenerovat výstupní adresář obsahující soubor herního enginu spolu s potřebnými knihovnami, složku s kopií souboru projektu a všemi daty, na kterých je projekt závislý.



Obrázek 18: Export projektu

Dialogové okno exportu hry je dostupné přes tlačítko export v sekci projekt v hlavním nabídkovém pruhu. Zde si uživatel vybere místo uložení cílového adresáře a cílovou platformu. V případě platformy Windows se zkopírují i instalační soubory standardních knihoven C++. U platformy Linux musí uživatel zaručit splnění závislostí sám, neboť se to liší u každé distribuci.

4.5 Programování základních prvků plošinových her

Předposlední podkapitola je věnována příkladům programování typických prvků plošinových her pomocí programovacího jazyka Lua. Kompletní rozhraní programovacího jazyka Lua je k dispozici jako příloha B.

4.5.1 Rozpohybování hráče

Nejprve vytvoříme objekt hráče. K vytvořenému objektu hráče nadefinujeme konstanty, které budou potřeba při provádění pohybu hráče, přidáním události `PŘI VYTVOŘENÍ`. Otevřeme skript reakce na danou událost a zapíšeme následující:

```
  -- maximální rychlost chůze objektu hráče bude 300 px/s
  rychlost_chuze = 300
```

Nyní přidáme další událost – `PŘI AKTUALIZACI`. Zde doplníme kód, který bude představovat využití vstupu uživatele pro pohyb hráče.

```
-- pohyb směrem doprava
if get_keyboard(KEY_RIGHT) then
  this.velocity.x = rychlost_chuze

-- pohyb směrem doleva
```

```
elseif get_keyboard(KEY_LEFT) then
  this.velocity.x = -rychlost_chuze

-- jinak zastavíme pohyb hráče po vodorovné ose
else
  this.velocity.x = 0.0
end

-- nastavíme animaci chůze pokud je nenulová rychlost
if this.velocity.x ~= 0 then
  this.animation = spr_player_walk
else
        this.animation = spr_player_stand
end
```

Nyní se postava hráče pohybuje vlevo a vpravo pomocí vstupu z klávesnice.

4.5.2 Gravitace a kolize se zdí

Využijeme pomocnou proměnnou *pada*, která definuje, jestli objekt hráče padá či nikoliv.

```
-- definujeme pomocnou proměnnou typu boolean
pada = false

-- nadefinujeme velikost gravitace
gravitace = 600
```

Je nutné testovat každý snímek, zda se nenachází objekt hráče nad objektem zdi. Pokud ano, musíme nastavit rychlost objektu tak, aby objekt padal.

```
-- test kolize s objektem zdi pod hráčem
local x = this.center.x
local y = this.bottom + 1
if not pada and not place_meeting(x, y, object_wall) then
  pada = true
end

-- * KÓD POHYBU HRÁČE POMOCÍ KLÁVESNICE *

-- aplikace gravitace, pokud hráč padá
if pada then
        this.velocity.y = this.velocity.y + gravitace * frame_time
end

-- doplnění stavů animace o animaci pádu
if this.position.y ~= this.last_position.y then
        this.animation = spr_player_fall
elseif this.velocity.x ~= 0 then
  this.animation = spr_player_walk
else
```

```
  this.animation = spr_player_stand
end
```

Nakonec je potřeba doplnit kód hráče tak, aby se při kolizi se zdí pád zastavil. Zároveň si můžeme kolizi připravit i pro zastavení o zeď při pohybu na strany. Testování kolize pro pohybující se objekty pracuje pixel po pixelu, tak nám stačí zjistit, jestli byl posun horizontalní nebo vertikální a pro danou souřadnici vynulovat rychlost a vrátit na předchozí pozici. Přidáme tedy událost kolize s objektem zdi a doplníme kód následovně:

```
-- pohyb na ose X
if this.step.x ~= 0.0 then
        -- vrácení objektu zpět a vynulování rychlosti
        this.velocity.x = 0
        this.position.x = this.last_position.x
end

-- pohyb na ose Y
if this.step.y ~= 0.0 then
        -- zastavení pádu
        if this.velocity.y > 0 then
                pada = false
        end

        -- vrácení objektu zpět a vynulování rychlosti
        this.velocity.y = 0
        this.position.y = this.last_position.y
end
```

4.5.3 Skok hráče

Pro skok nejprve dodefinujeme v události `PŘI VYTVOŘENÍ` konstantu rychlosti skoku:

```
-- * OSTATNÍ KONSTANTY *

-- rychlost skoku bude 600 px/s
rychlost_skoku = 600
```

Nyní už jen doplníme událost aktualizace o provedení skoku při stlačení tlačítka šipky nahoru.

```
-- zkontrolujeme, jestli nepadá a jestli drží tlačítko skoku
if not pada and get_keyboard(KEY_UP) then
  -- nastavíme pomocné hodnoty, rychlost a přehrajeme zvuk skoku
  pada = true
  play_sound(snd_jump)
  this.velocity.y = -rychlost_skoku
end
```

```
-- * ZBYTEK KÓDU UDÁLOSTI AKTUALIZACE *
```

4.5.4 Sběratelné předměty

Jelikož budeme potřebovat inicializovat globální proměnnou, která bude uchovávat například počet bodů (skóre), přidáme do globálního nastavení událost `PŘI SPUŠTĚNÍ` a vyplníme kód následovně:

```
-- celkový počet bodů (nulový při spuštění)
pocet_bodu = 0
```

Pro sběratelné předměty nejdříve vytvoříme nový objekt – obecný bonus. Další specifické bonusy musí mít předka obecný bonus a definovat podobné nastavení prostředí pomocí události `PŘI VYTVOŘENÍ`, například:

```
-- počet bodů za sebrání
pocet_bodu = 100
```

Nyní přidáme objektu hráče událost kolize s obecným bonusem. Ten naplníme kódem, který při vykonání přičte skóre a vymaže objekt bonusu.

```
-- přičtení bodu
global.pocet_bodu = global.pocet_bodu
  + other.env.pocet_bodu

-- spustíme zvuk sebrání bonusu
play_sound(snd_bonus)

-- smažeme objekt bonusu
destroy_object(other)
```

4.5.5 Reakce hráče na zranění

Různé hry řeší reakci hráče na zranění různě. V některých má hráč více životů a tedy je mu několik chyb odpuštěno. V jiné hře pro hráče končí hra při první chybě. První případ by vyžadoval deklaraci globální proměnné reprezentující počet životů, ale pro jednoduchost zůstaneme u druhého připadu. Stačí nám tedy definovat objekty, které budou hráči škodit, a objektu hráče doplnit událost kolize s nimi, která by mohla vypadat následovně:

```
-- došlo ke kolizi s objektem pasti můžeme odstranit objekt hráče
object_destroy()

-- nebo restartovat hru
restart()
```

```
-- nebo ukončit úroveň a nastavit uživatelské rozhraní prohry
set_level(nil)
set_layout(ui_lost)

-- záleží na volbě uživatele
```

4.5.6 Pohyb hráče po žebříku

Náš předchozí kód stačí doplnit o pomocnou proměnnou, která bude signalizovat stav, kdy je hráč na žebříku. Tedy událost `PŘI VYTVOŘENÍ` doplníme následovně:

```
-- * ZBYTEK UDÁLOSTI PŘI VYTVOŘENÍ *

-- pomocná proměnná
na_zebriku = false
```

Nyní je potřeba upravit kód události aktualizace, protože, pokud je hráč na žebříku, tak by neměl padat, a zároveň tlačítky nahoru a dolů by mu měl být umožněn pohyb.

```
-- upravený test kolize se zdí pod hráčem
local x = this.center.x
local y = this.bottom + 1
if not na_zebriku and not pada then
  if not place_meeting(x, y, object_wall) then
    pada = true
  end
end

-- * KÓD POHYBU HRÁČE POMOCÍ KLÁVESNICE *
-- * APLIKACE GRAVITACE PŘI PÁDU *

-- doplnění o pohyb na žebříku
if na_zebriku then
  -- pohyb nahoru
  if get_keyboard(KEY_UP) then
    this.velocity.y = -rychlost_chuze
  elseif get_keyboard(KEY_DOWN) then
    this.velocity.y = rychlost_chuze
  end

  -- zrušení stavu
  na_zebriku = false
end

-- * STAVY ANIMACÍ *
```

Nakonec je ještě potřeba přidat událost kolize s objektem žebříku, která budou pouze obsahovat nastavení pomocné proměnné, tedy:

```
-- hráč se dotýká žebříku
na_zebriku = true
```

4.5.7 Střelba munice

Nejprve vytvoříme objekt střely, kterému bude stačit doplnit události kolize se zdí a kolize s pastí/nepřítelem. Při kolizi se zdí pouze zavoláme zničení objektu, protože nechceme, aby střely proletěli zdí.

```
-- zničíme objekt střely
object_destroy(this)

-- přehrajeme zvuk minutí
play_sound(snd_miss)
```

Při kolizi s pastí/nepřítelem nejprve zničíme past/nepřítele, a teprve potom zničíme objekt střely.

```
-- zničíme objekty zdi a střely
object_destroy(other)
object_destroy(this)

-- přehrajeme zvuk zásahu
play_sound(snd_hit)
```

Nakonec musíme doplnit kód události aktualizace hráče o střelbu. K tomu se stačí zeptat, jestli uživatel stisknul tlačítko střelby (například mezerník) a poté vytvořit objekt střely, kterému se nastaví směrový vektor podle orientace objektu hráče.

```
-- střelba
if get_keyboard(KEY_SPACE) then
  -- vytvoříme objekt střely
  strela = create_object(
    this.center.x, this.center.y, obj_bullet)

  -- nastavíme rychlost 500 px/s
  strela.velocity.x = 500
  if this.direction == DIR_LEFT then
    strela.velocity.x = -strela.velocity.x
  end

  -- přehrajeme zvuk výstřelu
  play_sound(snd_shoot)
end

-- * ZBYTEK KÓDU UDÁLOSTI AKTUALIZACE *
```

Tento kód by se dal doplnit o globální proměnnou, která by indikovala maximální počet střel. Také by se mohl doplnit o časovač, který by hlídal, jak rychle po sobě může hráč střelu vystřelit.

4.6 Tutoriál návrhu jednoduché hry

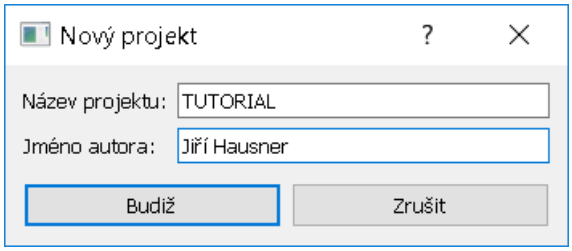
V poslední podkapitole se pokusím naznačit jednoduchý tutoriál pro vytvoření triviální hry v editoru her. Všechny soubory, které byly využity při tvorbě projektu tutoriálu jsou k dispozici v adresáři /data/tutorial-data.

Začneme spuštěním editoru – otevřeme soubor editor.exe (v případě platformy Windows).

4.6.1 Příprava projektu

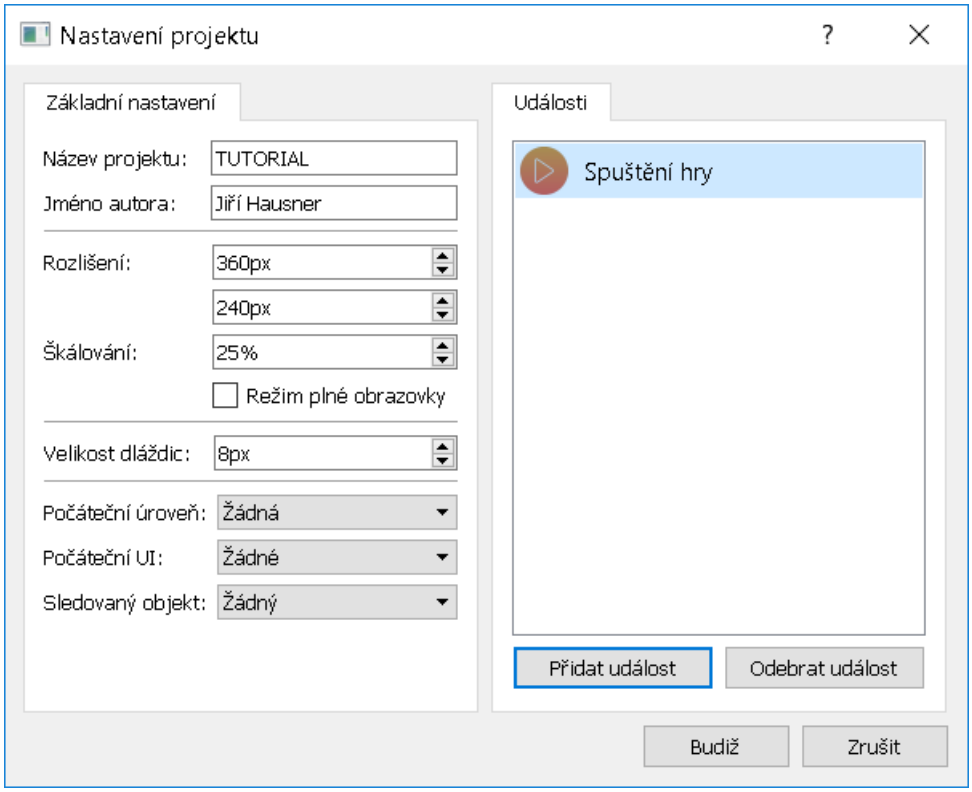
Nejdříve vytvoříme nový projekt pomocí tlačítka *Nový* pod položkou *Soubor* v hlavním navigačním pruhu. Následně se objeví nabídka volby adresáře, do kterého se nový projekt má umístit (v nejlepším případě takový adresář vytvoříme).

Po zvolení adresáře pro uložení projektu se nám otevře dialog *Nový projekt*, ve kterém vyplníme název projektu a jméno autora (obrázek 19).



Obrázek 19: Tutoriál – nový projekt

Nyní přejdeme k základnímu nastavení projektu pomocí otevření dialogu *Nastavení projektu*, který nastavíme dle vlastních potřeb – ukázka nastavení projektu je na obrázku 20. V projektu tutoriálu volíme dané rozlišení a nastavíme škálování na 25% – jelikož je velikost dlaždice 8px, bude ve výsledku 4x větší.



Obrázek 20: Tutoriál – nastavení projektu

Přidáme událost `SPUŠTĚNÍ HRY`, ve které budeme chtít inicializovat proměnnou, která bude reprezentovat skóre. Tedy událost vyplníme následujícím kódem:

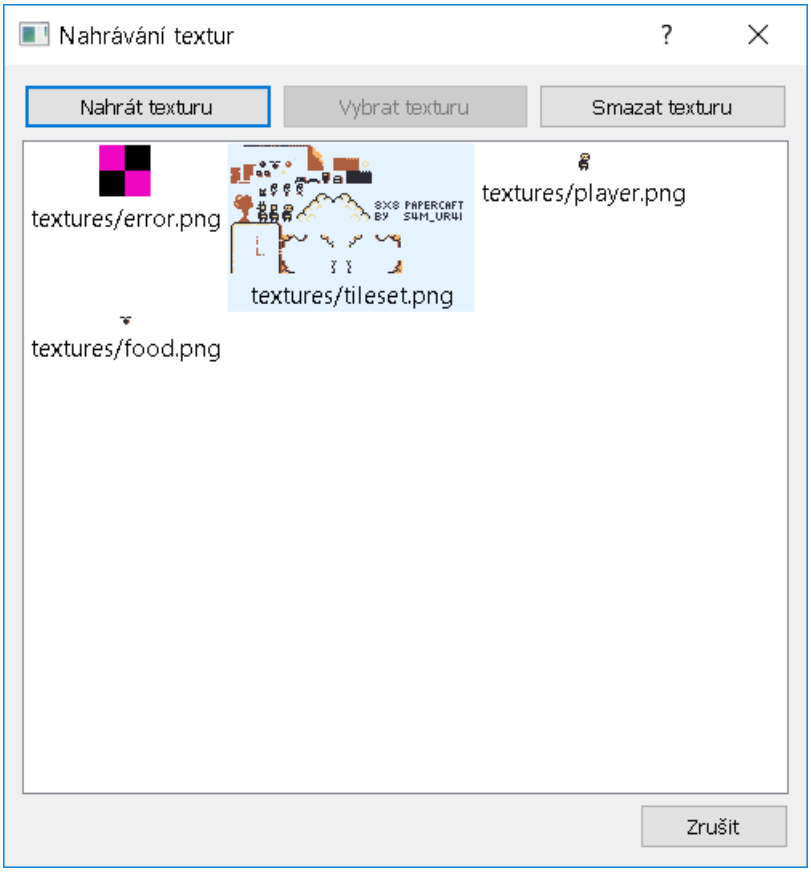
```
-- nastavení počtu bodů
points = 0
```

Počáteční úroveň, uživatelské rozhraní a sledovaný objekt vybereme až budou připraveny.

4.6.2 Nahrání zdrojových souborů

Následujícím krokem je načtení zdrojových souborů textur, písma a zvuků. Uživatel musí načíst pouze texturu hráče (minimálně jednu – avšak pouze jedna nebude stačit pro rozpohybování animace) a texturu dlaždice (bez ní nemůže navrhnout úroveň). Každý herní projekt zahrnuje výchozí písmo *Arial* velikosti 16px. Způsob přidání textur je popsán v kapitole 4.3.1, přidávání zvuků v kapitole 4.3.4 a písma v kapitole 4.3.5.

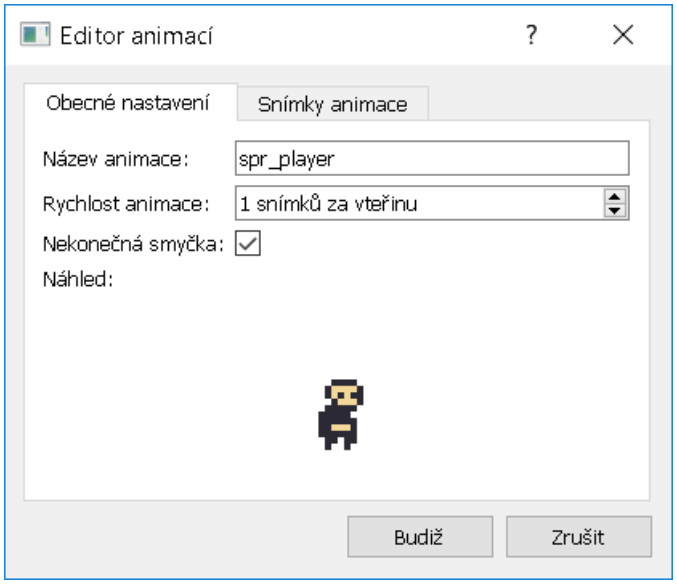
Jako první klikneme na tlačítko *Spravovat textury* pod položkou *Úpravy* v hlavním navigačním pruhu. Zobrazí se nám dialog *Nahrávání textur*, ve kterém opakovaným kliknutím na tlačítko *Nahrát texturu* nahrajeme všechny textury, jež budeme ve hře potřebovat – texturu pro hráče, bonus a texturu pro dlaždice. Po dokončení nahrávání by měl dialog vypadat jako na obrázku 21.



Obrázek 21: Tutoriál – načtení textur

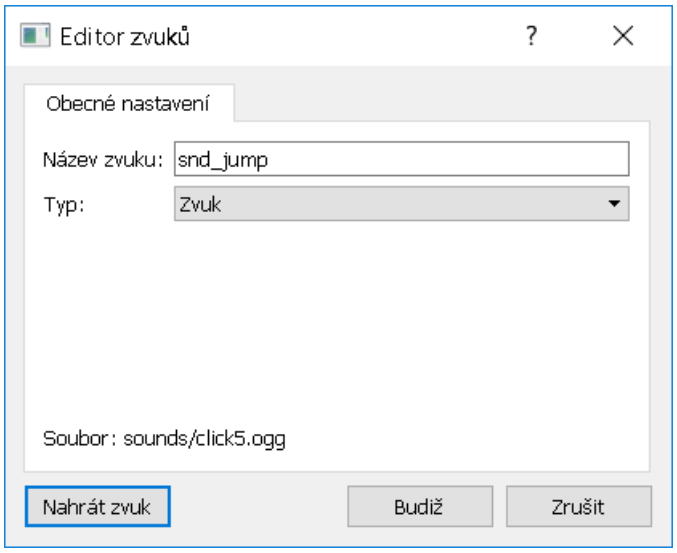
Jakmile máme soubory textur načteny, musíme ještě vytvořit animace, které použijeme pro vykreslení herních objektů. Klikneme na tlačítko *Přidat animaci* pod položkou *Úpravy* v hlavním navigačním pruhu, které nám vytvoří novou animaci a otevře *Editor animací*.

První animací bude animace hráče – nastavíme její název a v liště *Snímky animace* přiřadíme texturu hráče. Úvodní lišta dialogu by měla následně vypadat jako na obrázku 22. Proces zopakujeme ještě jednou pro animaci bonusu, zdi a cíle.



Obrázek 22: Tutoriál – tvorba animací

Nyní načteme zdrojové soubory zvuku – klikneme na tlačítko *Přidat zvuk* pod položkou *Úpravy* v hlavním navigačním pruhu. Otevře se nám dialog *Editor zvuku* (obrázek 23), kde vyplníme název nového zvuku, nastavíme jeho typ (v našem případě ponecháme zvuk) a kliknutím na tlačítko *Nahrát zvuk* vybereme a nahrajeme zdrojový soubor zvuku do struktury projektu.

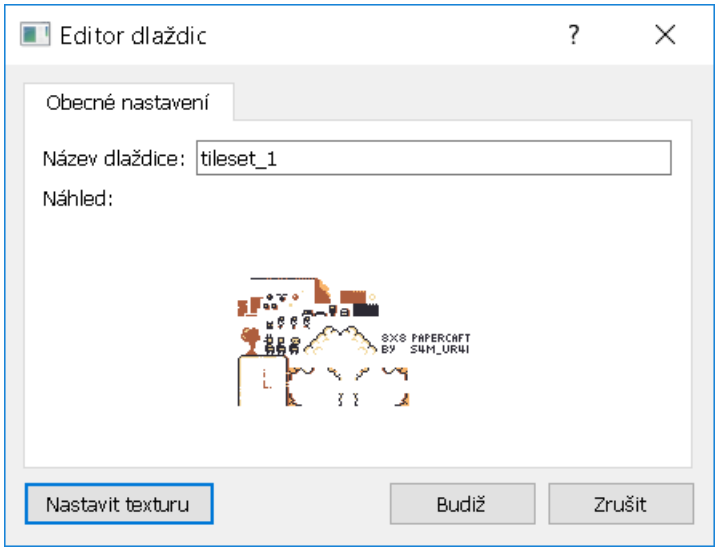


Obrázek 23: Tutoriál – tvorba zvuku

Pro náš projekt budeme potřebovat zvuk reprezentující skok hráče a zvuk, který se přehraje při sebrání bonusu.

Nakonec musíme připravit objekt dlaždice, který budeme využívat pro tvorbu vizuální stránky úrovní. Klikneme na tlačítko *Nová dlaždice* pod položkou *Úpravy*

v hlavním navigačním pruhu a vytvoříme ji. Při otevření dialogu *Editor dlaždic* pouze nastavíme název dlaždice a přiřadíme dlaždici texturu – viz obrázek 24.



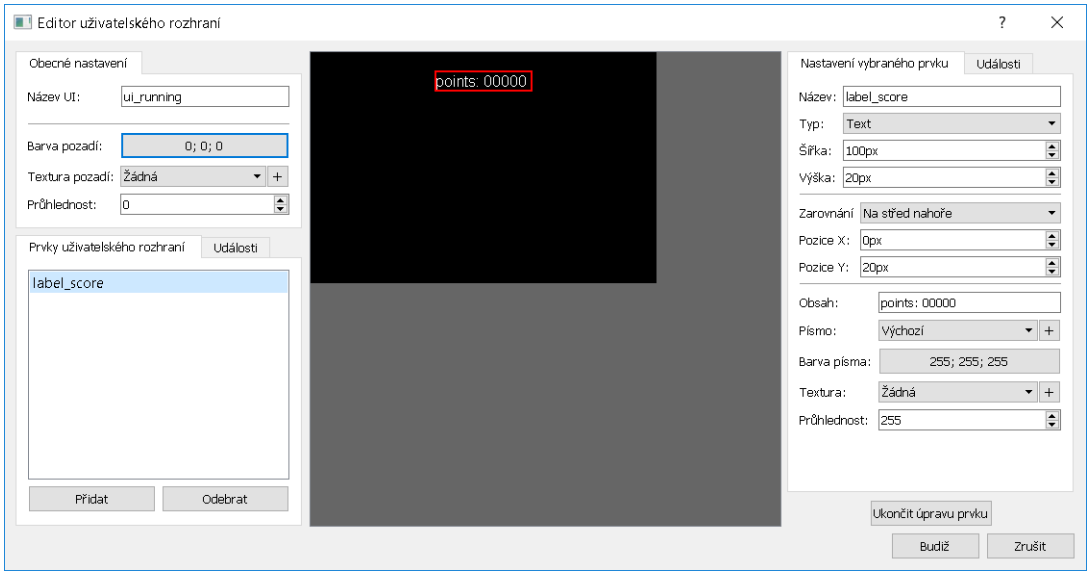
Obrázek 24: Tutoriál – tvorba dlaždice

Nyní máme všechny zdrojové soubory připravené a načtené do struktury projektu. Máme připravené animace i dlaždice, takže se můžeme vrhnout do návrhu objektů, úrovní a uživatelských prostředí.

4.6.3 Vytvoření jednoduchého uživatelského rozhraní

Pro náš tutoriálový projekt nám postačí dvě uživatelská rozhraní – jedno, které se bude využívat během hraní úrovní, a druhé pro vyjádření konce hry.

Klikneme na tlačítko *Nové uživatelské rozhraní* pod položkou *Úpravy* v hlavním navigačním pruhu a nazveme ho `ui_running`. Obecné nastavení můžeme ponechat ve výchozím nastavení až na položku průhlednost, kterou nastavíme na nulu (chceme aby šla vidět úroveň).



Obrázek 25: Tutoriál – uživatelského rozhraní průběhu hry

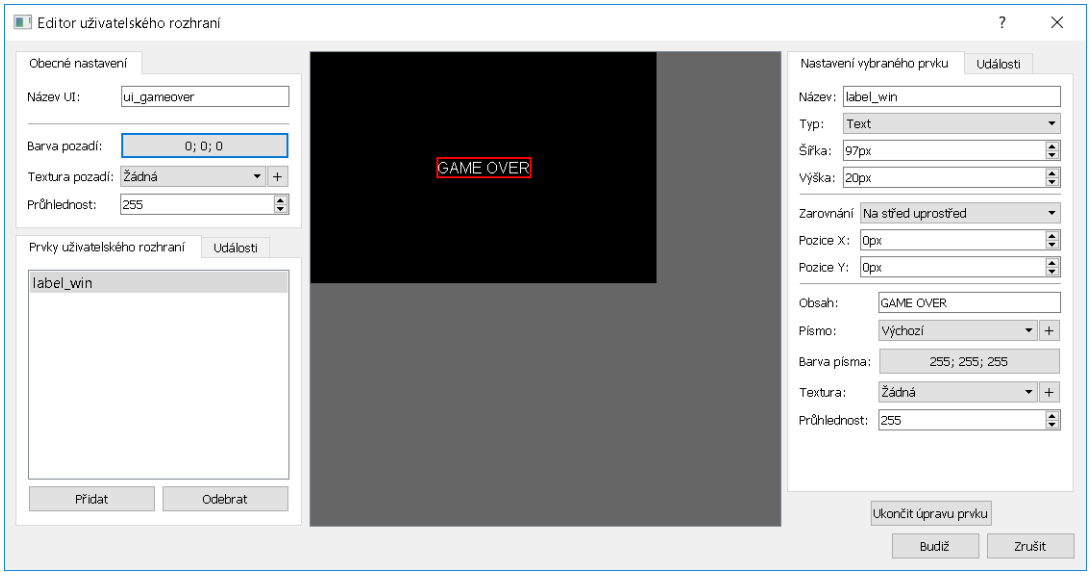
Jelikož jedinná ukládaná hodnota je skóre, postačí nám vytvořit jedinný prvek, který nazveme `label_score`. Tento prvek nastavíme podle obrázku 25. Jelikož požadujeme, aby se v každý okamžik zobrazilo přesné skóre, přidáme prvku událost `AKTUALIZACE`, ve které se budeme snažit docílit aktualizaci obsahu prvku `label_score`.

```
-- nastavení počtu bodů
this.content = string.format("points: %0.5d", math.floor(
        global.points
))
```

Nakonec budeme požadovat, aby při stisknutí tlačítka escape došlo k ukončení hry, tedy přidáme událost `AKTUALIZACE`, kterou vyplníme následujícím kódem:

```
-- ukončení tlačítkem ESC
if get_keyboard(KEY_ESCAPE) then
  exit()
end
```

Druhé uživatelské rozhraní bude sloužit k upozornění hráče, že dokončil hru. Vytvoříme tedy další nové uživatelské rozhraní, které pojmenujeme `ui_gameover`.



Obrázek 26: Tutoriál – uživatelské rozhraní konce hry

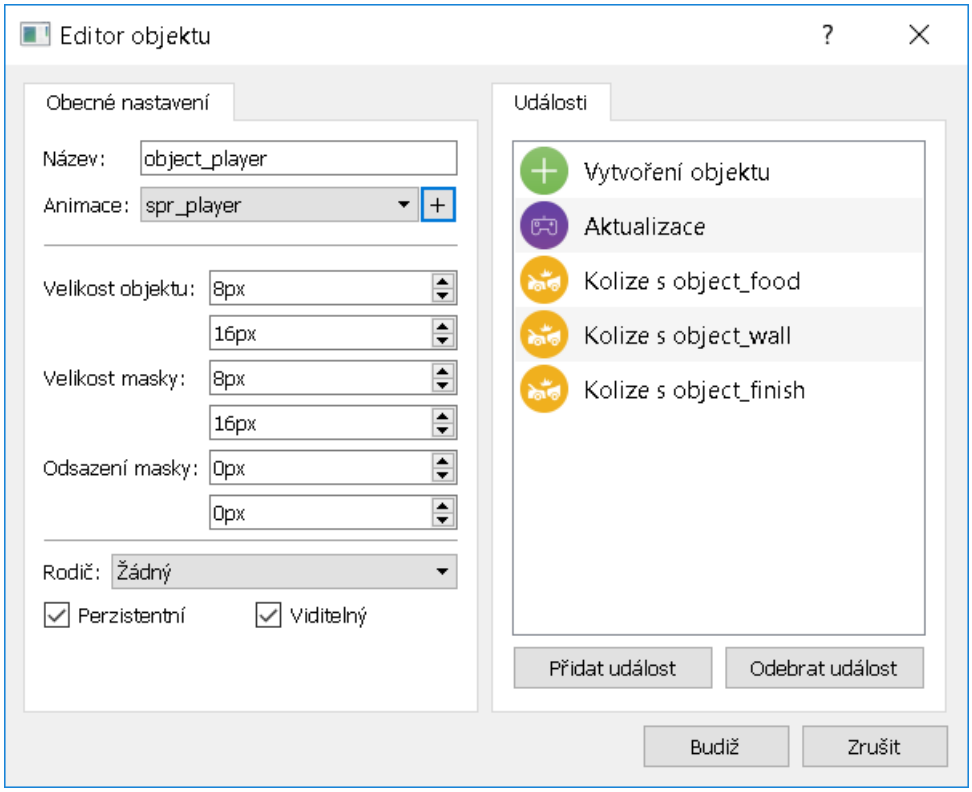
Zde nám postačí přidat prvek reprezentující text konce hry a naprogramovat, aby při stisknutí tlačítka escape došlo k restartování hry, což docílíme následujícím kódem:

```
-- restartování tlačítkem ESC
if get_keyboard(KEY_ESCAPE) then
        set_layout(ui_running)
        set_level(level_1)
        global.points = 0
end
```

Tím máme hotová uživatelská rozhraní a už nám pouze chybí nastavit herní objekty a vytvořit herní úroveň.

4.6.4 Nastavení herních objektů

V našem tutoriálu budeme pracovat se čtyřmi objekty – hráč, bonus, zeď a cíl. Začneme tedy kliknutím na na tlačítko *Přidat objekt* pod položkou *Úpravy* v hlavním navigačním pruhu. Tímto způsobem přidáme čtyři objekty, kterým nastavíme jejich názvy, animace a základní velikosti. Jedinný objekt, který bude potřebovat doprogramovat logiku bude objekt hráče. Jakmile máme vytvořené a připravené všechny objekty, vrhneme se na programování skriptů událostí objektu hráče.



Obrázek 27: Tutoriál – nastavení objektu hráče

První událostí bude `VYTVOŘENÍ OBJEKTU`. Zde budeme inicializovat konstanty rychlostí pohybu, skoku a gravitace – viz kód níže:

```
-- rychlost chůze, skoku a gravitace
rychlost_chuze = 50
rychlost_skoku = 150
gravitace = 300
```

Nyní přidáme událost `AKTUALIZACE`. Zde musíme vyřešit logiku hráče jako například pohyb a skok pomocí vstupu z klávesnice, aplikaci gravitace při pádu a zjištění, zda by neměl hráč padat. Tento skript by mohl vypadat následovně:

```
-- test kolize s objektem zdi pod hráčem
local x = this.center.x
local y = this.bottom + 1
if not pada and not place_meeting(x, y, object_wall) then
        pada = true
end

-- zkontrolujeme, jestli nepadá a jestli drží tlačítko skoku
if not pada and get_keyboard(KEY_UP) then
        -- nastavíme pomocné hodnoty, rychlost a přehrajeme zvuk skoku
        pada = true
        play_sound(snd_jump)
```

```
        this.velocity.y = -rychlost_skoku
end

-- pohyb směrem doprava
if get_keyboard(KEY_RIGHT) then
        this.velocity.x = rychlost_chuze

-- pohyb směrem doleva
elseif get_keyboard(KEY_LEFT) then
        this.velocity.x = -rychlost_chuze

-- jinak zastavíme pohyb hráče po vodorovné ose
else
        this.velocity.x = 0
end

-- aplikace gravitace pokud hráč padá
if pada then
        this.velocity.y = this.velocity.y + gravitace * frame_time
end
```

Nakonec je potřeba přidat událost `KOLIZE` pro všechny ostatní objekty. Prvním objektem bude objekt zdi. Zde se musíme postarat, aby se hráč při kolizi s objektem zdi zastavil a vrátil na předchozí pozici. To vystihuje kód níže.

```
-- pohyb na ose X
if this.step.x ~= 0 then
  -- vrácení objektu zpět a vynulování rychlosti
  this.velocity.x = 0
  this.position.x = this.last_position.x
end

-- pohyb na ose Y
if this.step.y ~= 0 then
  -- zastavení pádu
  if this.velocity.y > 0 then
    pada = false
  end

  -- vrácení objektu zpět a vynulování rychlosti
  this.velocity.y = 0
  this.position.y = this.last_position.y
end
```

Další kolizní událost bude s objektem bonusu, která se musí postarat o odstranění objektu bonusu, přehrání zvuku, který reprezentuje sebrání bonusu, a přičtení počtu budů.

```
-- odstraníme objekt bonusu
destroy_object(other)
```

```
-- přehrajeme zvuk
play_sound(snd_food)

-- přičteme body
global.points = global.points + 100
```

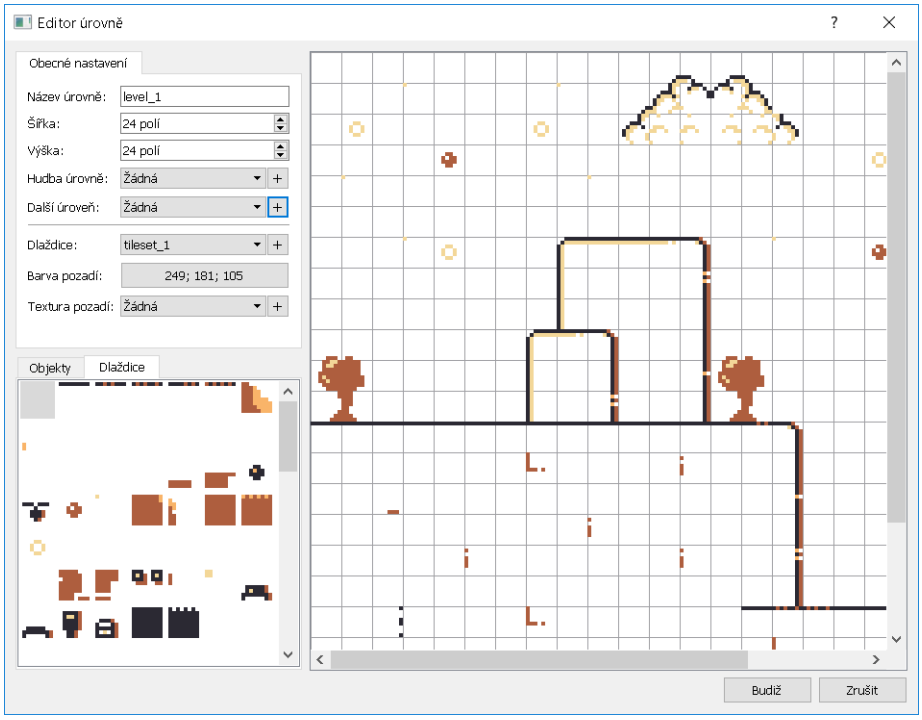
Poslední událost kolize je s objektem cíle. Zde nám postačí, aby při kolizi hráče s objektem cíle došlo k přechodu do uživatelského rozhraní konce hry a aby nebyla žádná úroveň aktivní.

```
-- nastavíme uživatelské rozhraní konce hry
set_layout(ui_gameover)
set_level(nil)
```

4.6.5 Tvorba úrovní

Nyní nastal čas doplnit hru o úrovně. Vytvoření úrovně je dostupné přes tlačítko *Nová úroveň* pod položkou *Úpravy* v hlavním navigačním pruhu. Editor úrovní a jeho návrhář je popsán v kapitole 4.3.8.

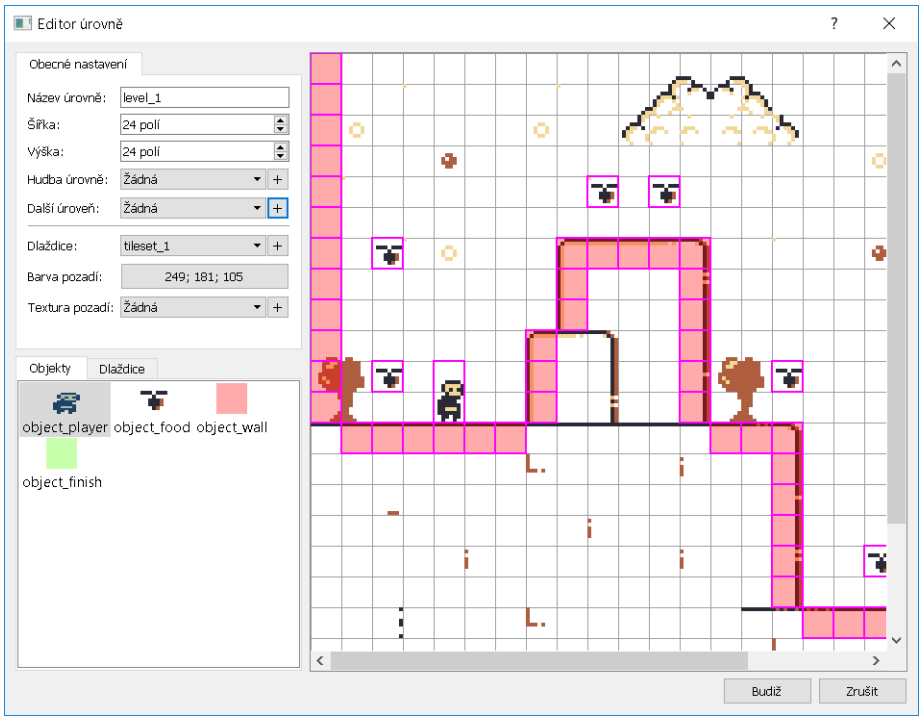
V dialogu *Editor úrovně* (obrázek 30) musíme přiřadit úrovni název (nejlépe jedinečný), nastavit jeho hlavní dlaždici (představující textury pozadí), určit velikost úrovně (počet políček v šířce a výšce) a určit následující úroveň (pokud nejde o úroveň poslední – v takovém případě volíme možnost *Konec hry*).



Obrázek 28: Tutoriál – návrh vizuální stránky úrovně

Jakmile máme úroveň nastavenou, můžeme přejít k jejímu návrhu. Popis prvků a ovládání dialogu *Editor úrovní* je k dispozici v kapitole 4.3.8.

Jako první volíme prvky vybrané dlaždice a kliknutím levým tlačítkem myši je zasadíme do mřížky úrovně podle vlastní volby. Navrhujeme tím vizuální stránku úrovně. Dokončení návrhu úrovně by mohlo vypadat jako na obrázku 28.



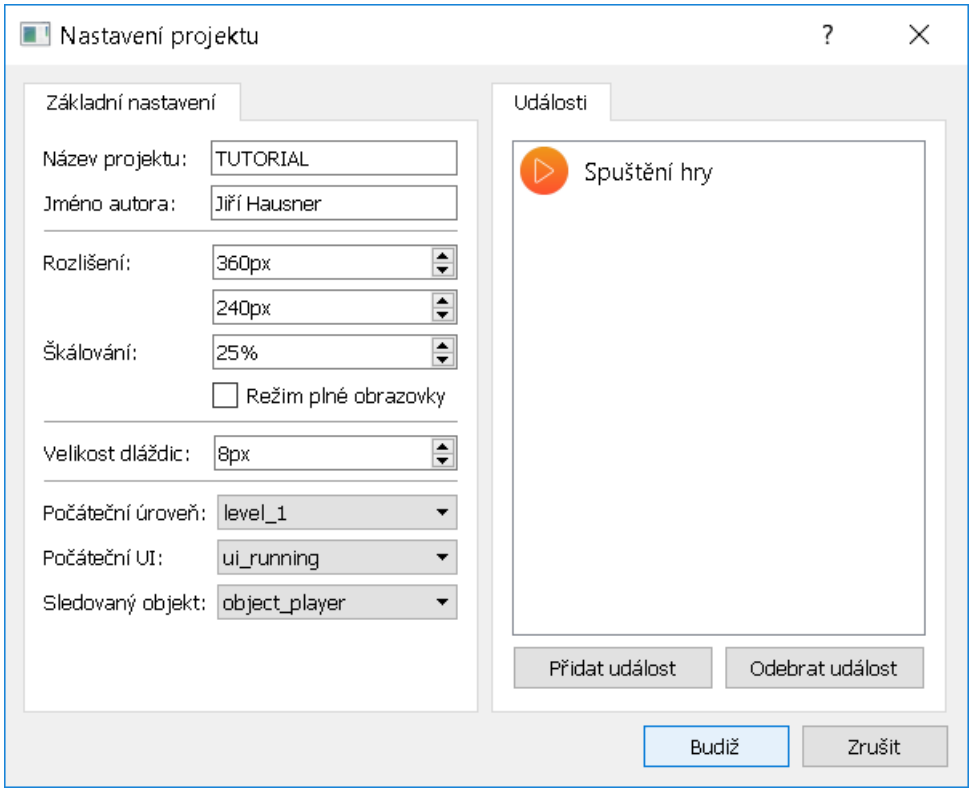
Obrázek 29: Tutoriál – přidání objektů do úrovně

Vybereme lištu objektů a nejdříve umístíme objekt hráče na správné místo. Následně vyplníme naší úroveň ostatními objekty – bonusy, zdmi a cílem. Například vybereme objekt `obj_bonus` a kliknutím levého tlačítka myši do mřížky úrovně jej můžeme opakovaně přidat. Návrh úrovně by měl nyní vypadat jako na obrázku 29.

Nyní je úroveň hotová. Editor můžeme ukončit stisknutím tlačítka `budiž`. a můžeme hru poprvé spustit (o tom více v následující kapitole).

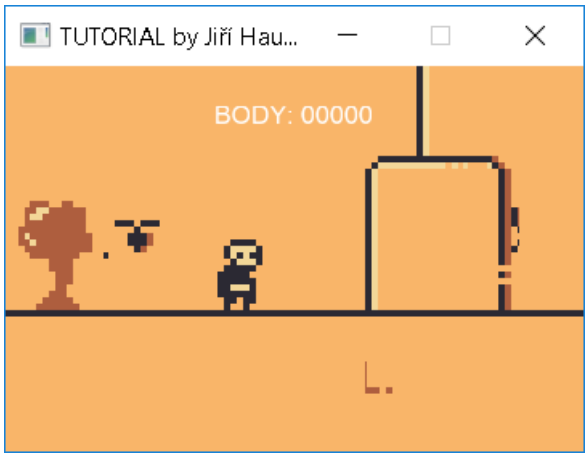
4.6.6 Testování hry

Jakmile byly přidány zdrojové soubory, vytvořené animace, připravená uživatelské rozhraní, nastavené herní objekty (hlavně objekt hráče) a vytvořena první úroveň, musí uživatel zobrazit dialog nastavení projektu pomocí tlačítka *Globální nastavení* pod položkou *Projekt* v hlavním navigačním pruhu. Zde uživatel nastaví počáteční úroveň, počáteční uživatelské rozhraní a sledovaný objekt podle obrázku 30 a hra bude připravena ke spuštění.



Obrázek 30: Tutoriál – konečné nastavení hry

Pro spuštění testování hry musí uživatel stisknout tlačítko *Spustit* pod položkou *Projekt* v hlavním navigačním pruhu. Ukázka spuštěného projektu je na obrázku 31.

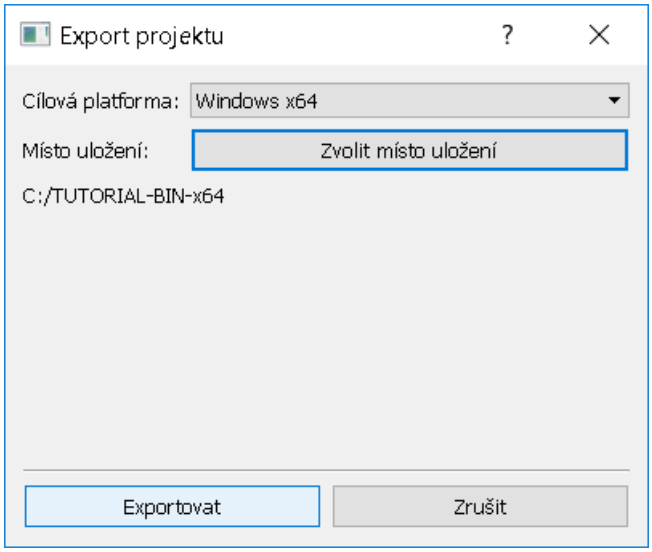


Obrázek 31: Tutoriál – testování hry

4.6.7 Dokončení hry

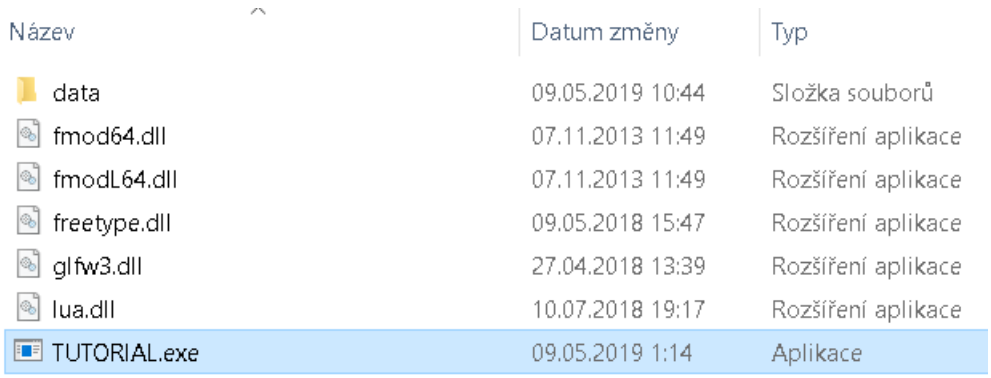
Nakonec převedeme projekt do spustitelné podoby přes dialog *Export projektu*, který zobrazíme stisknutím tlačítka *Export* pod položkou *Soubor* v hlavním navi-

gačním pruhu. Celkový popis exportu projektu je v kapitole 4.4.6. Zvolíme cílovou platformu a místo uložení (nejlépe nově vytvořená prázdná složka) a stiskneme tlačítko *Exportovat* (obrázek 32).



Obrázek 32: Tutoriál – převedení projektu do spustitelné podoby

Po převedení do spustitelné podoby se uživateli v případě operačního systému Windows zobrazí průzkumník souborů otevřený v adresáři exportovaného projektu, kde soubory projektu budou uloženy do složky /data a samotná hra bude spustitelná souborem `<název-projektu>.exe`. Struktura adresáře exportované hry v operačním systému Windows je k náhledu na obrázku 33.



Obrázek 33: Tutoriál – výstupní složka se spustitelným souborem

Nyní je tutoriálový projekt dokončen a připraven k distribuci.

Závěr

Cílem práce bylo vytvoření herního engine pro plošinové hry, vytvořit jeho editor her a s jeho pomocí navrhnout dvě jednoduché hry.

Herní engine poskytuje načítání a správu textur, zvuků a písma, správu vstupů z klávesnice, myši a případně joysticku, správu úrovní rozložených do mřížky, disponuje sadou prvků uživatelského rozhraní a implementuje základní funkcionalitu logiky a fyziky herních objektů, prvků uživatelských rozhraní a her samotných. Komunikace mezi herním enginem a editorem je založena na sdílení souboru herního projektu.

Editor her umožňuje pohodlné vytvoření nových her a jejich sestavení pro různé platformy. Poskytuje celou řadu editorů spravujících načítání, úpravu a mazání zdrojových souborů jako jsou textury, zvuky nebo písma. Dále disponuje návrhářem herních úrovní a uživatelských rozhraní. Uživatel může vytvářet vlastní animace, objekty kolizí a dokonce vlastní herní objekty. Testování herního projektu a jeho export je k dispozici stiknutím jediného tlačítka.

Veškerá herní logika je řešena pomocí systému událostí a rozhraní programovacího jazyku Lua, které umožňuje snadné rozšíření herních objektů, uživatelských rozhraní a jejich prvků.

Conclusions

The result of this work was to implement game engine for platform games, its editor and use it to design two simple games.

Game engine provides loading and managment of texture, sound and font files, managment of input/output with devices such as keyboard, mouse or even joystick, managment of tile-based game levels, has set of basic user interface elements and implements basic functionality of game logics and physics for game objects, user interfaces and for games itself. Communication between game engine and editor is based on sharing of project file.

The game editor allows comfortable creation of new games and its export on multiple platforms. It provides whole range of editors, which manage loading, editing and deletion of source files such as textures, sounds or fonts. It futher has designer for game levels and layouts of user interfaces. User can create his own animations, collision objects and is even allowed to create his own game objects. Testing of game project and its export is available with single button click.

All game logic is realized using event system and interface of programming language Lua, which is used for easy extension of game objects, layouts and its elements.

A Obsah přiloženého CD

bin/
: Program Editor ve složce /editor a složky s hrami. Adresář obsahuje i všechny runtime knihovny a další soubory potřebné pro bezproblémový běh programu z CD.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové texty programu Engine a Editor se všemi potřebnými (příp. převzatými) zdrojovými texty, knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření spustitelných verzí programu.

readme.txt
: Instrukce pro instalaci a spuštění programu Editor a her v editoru vytvořených, včetně všech požadavků pro jeho bezproblémový provoz.

Navíc CD obsahuje:

data/
: Složka obsahující testovací data použitá v práci a pro potřeby testování práce při tvorbě posudků a obhajoby práce.

install/
: Instalátory aplikací, runtime knihoven a jiných souborů potřebných pro provoz programu Engine a Editor, které nejsou standardní součástí operačního systému určeného pro běh programu.

literature/
: Vybrané položky bibliografie, příp. jiná užitečná literatura vztahující se k práci.

U veškerých cizích převzatých materiálů obsažených na CD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na CD, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `readme.txt`.

B Kompletní rozhraní rozšíření her pomocí jazyka Lua

B.1 Rozhraní systémových tříd

Tato podsekce je věnována popisu rozhraní systémových tříd, ke kterým má uživatel přístup.

První takovou třídou je třída **Vec2**, která představuje dvourozměrný vektor. Objekt této třídy lze vytvořit zavoláním konstruktu **Vec2()**, jež má dva nepovinné parametry odpovídající jeho souřadnicím.

x
: Vlastnost odkazující na x-ovou souřadnici vektoru.

y
: Vlastnost odkazující na y-ovou souřadnici vektoru.

width
: Synonymum pro vlastnost x.

height
: Synonymum pro vlastnost y.

Obdobně na tom je třída **Vec2**, která tentokrát představuje trojrozměrný vektor. Objekt této třídy lze vytvořit zavoláním konstruktu **Vec3()**, pro žádný parametr vytvoří nulový vektor, při jednom parametru vytvoří vektor se všemi složkami se stejnou velikostí nebo poslední možností je zadat všechny tři parametry.

x
: Vlastnost odkazující na x-ovou souřadnici vektoru.

y
: Vlastnost odkazující na y-ovou souřadnici vektoru.

z
: Vlastnost odkazující na z-ovou souřadnici vektoru.

r
: Synonymum pro vlastnost x.

g
: Synonymum pro vlastnost y.

b
: Synonymum pro vlastnost z.

Rozhraní třídy **Game**, které reprezentuje hru, je v této verzi herního enginu pro uživatele uzavřené a pro komunikaci s hrou je dostupné rozhraní funkcí definované v kapitole B.4.

Třída **Sprite**, jež reprezentuje objekt animace. Objekty této třídy nelze inicializovat. Implementuje následující vlastnosti a metodu:

id
: Vlastnost id dané animace. Určena pouze ke čtení.

name
: Vlastnost reprezentující název dané animace. Určena pouze ke čtení.

speed
: Vlastnost reprezentující počet snímků animace za sekundu.

loops
: Vlastnost odkazující na pravdivostní hodnotu, zda-li se daná animace opakuje v nekonečné smyčce.

textures_count
: Vlastnost odkazující na počet textur v dané animaci. Určena pouze ke čtení.

texture_at (index) – *number*
: Metoda, která vrátí index textury podle daného indexu v animaci.

Třída **Tileset**, jež reprezentuje objekt dlaždice. Objekty této třídy nelze inicializovat. Implementuje následující vlastnosti:

id
: Vlastnost id dané dlaždice. Určena pouze ke čtení.

name
: Vlastnost reprezentující název dané dlaždice. Určena pouze ke čtení.

texture
: Vlastnost odkazující index textury dané dlaždice. Určena pouze pro čtení.

Dále je k dispozici rozhraní třídy **Timer**, kterou je možné využít například jako stopky. Objekt této třídy lze vytvořit zavoláním konstruktoru **Timer()**, který má jeden volitelný parametr, pomocí kterého se nastavuje cílová hranice odpočtu času. Tato třídy implementuje vlastnosti a metody zmíněné níže.

elapsed_time
: Vlastnost, která vrátí délku času uběhlého od počátku počítání v sekundách.

coeficient
Vlastnost odkazující na zlomek vyjadřující vzdálenost časovače od cíle. Rozmezí [0, 1).

coeficient_no_overflow
Vrátí zlomek bez přetečení vyjadřující vzdálenost časovače od cíle. Rozmezí [0, 1].

finished
Vlastnost odkazujicí na pravdivostní hodnotu, zda-li časovač dosáhl cílové hranici.

start ()
Metoda, která spustí časovač.

start_with_length (length) – *number*
Synonymum pro metodu *start()*, avšak má jeden povinný parametr, pomocí kterého nastaví cílovou hranici časovače.

set_length (length) – *number*
Metoda, která nastaví cílovou hranici časovače. Od této hodnoty se počítá koeficient.

Herní úroveň je vyjádřena pomocí třídy **Level**. Rozhraní umožňuje nejen zjištění a nastavení vlastností úrovně, ale i práci s ní a jejími herními objekty.

id
Vlastnost odkazující na index dané úrovně. Určena pouze ke čtení.

name
Vlastnost reprezentující název dané úrovně. Určena pouze ke čtení.

tileset
Vlastnost vracející dlaždici dané úrovni. Pokud taková neexistuje, vrátí *nil*. Určena pouze ke čtení.

next_level
Vlastnost vracející následující úroveň. Pokud taková neexistuje, vrátí *nil*. Určena pouze ke čtení.

music
Vlastnost reprezentující index hudby přidělené k dané úrovni v editoru.

start ()
Metoda, která spustí danou úroveň.

restart ()
Synonymum metody *start()*.

reset_tiles ()
Metoda sloužící k obnovení rozložení dlaždic v dané úrovni.

get_tile_at (x, y) – *number*, *number*
Metoda, která vrátí index dlaždice na souřadnicích x a y.

set_tile_at (x, y, tile_index) – *number*, *number*, *number*
Metoda sloužící k nastavení dlaždice na souřadnici x a y.

Poslední hlavní třídou je třída **Layout** představující objekt uživatelského rozhraní. Ta definuje nejen přístup k práci s daným UI, ale i s jejími prvky.

id
Vlastnost odkazující na index daného uživatelského rozhraní. Určena pouze ke čtení.

name
Vlastnost reprezentující název daného uživatelského rozhraní. Určena pouze ke čtení.

background_color
Vlastnost reprezentující barvu pozadí daného uživatelského rozhraní. Vrací a nastavuje se pomocí hodnoty objektu *Vec3*.

background_texture
Vlastnost reprezentující texturu pozadí daného uživatelského rozhraní. Žádná textura je reprezentována hodnotou *nil*.

background_alpha
Vlastnost reprezentující hodnotu průhlednosti daného uživatelského rozhraní. Rozmezí hodnoty <0, 1>.

objects
Vlastnost odkazující pole prvků daného uživatelského rozhraní. Určena pouze ke čtení.

env
Vlastnost odkazující na tabulku prostředí daného uživatelského rozhraní.

select_next ()
Metoda, která vybere další vybíratelný objekt v daném uživatelském rozhraní.

select_previous ()
Metoda, která vybere předchozí vybíratelný objekt v daném uživatelském rozhraní.

click_selected ()
Metoda, která provede událost kliknutí nad aktuálně vybraným objektem v daném uživatelském rozhraní.

add_object (object) – *userdata*
Metoda, která je určena k přidání objektu pomocí jeho ukazatele do daného uživatelského rozhraní.

remove_object (object) – *userdata*
Metoda, která je určena k odebrání objektu pomocí jeho ukazadele z daného uživatelského rozhraní.

B.2 Rozhraní tříd herních objektů

Každý herní objekt je tvořen třídou **GameObject**, jež definuje základní vlastnosti a logiku.

id
Vlastnost odkazující na index daného herního objektu. Určená pouze ke čtení.

name
Vlastnost odkazující na název daného herního objektu.

size
Vlastnost reprezentující velikost daného herního objektu reprezentovanou vektorem.

mask_size
Vlastnost reprezentující velikost masky daného herního objektu reprezentovanou vektorem.

mask_offset
Vlastnost reprezentující odsazení masky daného herního objektu reprezentovanou vektorem.

animation
Vlastnost reprezentující aktuální index animace herního objektu.

animation_index
Vlastnost reprezentující aktuální index animace herního objektu.

is_animation_finished
Vlastnost reprezentující pravdivostní hodnotu, zda-li animace byla dokončena. Určena pouze ke čtení.

position
Vlastnost odkazující na vektor, který reprezentuje pozici daného herního objektu.

last_position
Vlastnost odkazující na předchozí pozici daného herního objektu reprezentovanou vektorem. Určená pouze ke čtení.

left
Vlastnost odkazující na nejlevější pozici daného objektu. Určena pouze ke čtení.

right ()
Vlastnost odkazující na nejpravější pozici daného objektu. Určena pouze ke čtení.

top ()
Vlastnost odkazující na nejvyšší pozici daného objektu. Určena pouze ke čtení.

bottom
Vlastnost odkazující na nejspodnější pozici daného objektu. Určená pouze ke čtení.

center
Vlastnost odkazující na dvourozměrný vektor reprezentující střed daného objektu. Určená pouze ke čtení.

direction
Vlastnost odkazující na orientaci daného herního objektu. Vhodné použít konstanty DIRECTION_LEFT a DIRECTION_RIGHT. Určená pouze ke čtení.

velocity
Vlastnost odkazující na vektor rychlosti daného herního objektu.

step
Vlastnost odkazující na vektor směru posledního pohybu daného herního objektu. Určena pouze ke čtení.

is_moving
Vlastnost, která vrátí pravdu, jestliže se daný objekt pohybuje (má nenulovou rychlost). Určena pouze ke čtení.

parents
Vlastnost odkazující pole rodičů daného herního objektu. Jsou reprezentovány indexy prototypů herních objektů.

persistent
Vlastnost reprezentující perzistentnost daného herního objektu.

visible
Vlastnost reprezentující viditelnost daného herního objektu.

env
: Vlastnost odkazující na tabulku prostředí daného herního objektu.

distance_from (object) – *userdata*
: Metoda, která vrátí vzálenost v pixelech od daného objektu k objektu reprezentovaným jeho ukazatelem *object*.

discard_event ()
: Metoda, která přeruší dědičné chování kolizního systému (přestane vykonávat události kolize nad prototypy rodičů daného objektu).

B.3 Rozhraní tříd objektů uživatelského rozhraní

Zakladní třídou, ze které dědí všechny ostatní třídy reprezentující prkvy uživatelského rozhraní, je třída **UIObject**, která definuje vlastnosti a metody popsané níže.

id
: Vlastnost odkazujicí na index daného objektu uživatelského rozhraní. Určená pouze ke čtení.

name
: Vlastnost odkazujicí na jméno daného objektu uživatelského rozhraní. Určená pouze ke čtení.

type
: Vlastnost odkazujicí na typ daného objektu uživatelského rozhraní. Určená pouze ke čtení.

position
: Vlastnost odkazujicí na vektor pozice daného objektu uživatelského rozhraní.

size
: Vlastnost odkazujicí na vektor velikosti daného objektu uživatelského rozhraní.

align
: Vlastnost odkazujicí na typ zarovnání daného objektu uživatelského rozhraní. Vhodné využít konstant ALIGN_TOP_LEFT, ALIGN_TOP_CENTER, atd.

alpha
: Vlastnost odkazujicí na hodnotu nastavení průhlednosti daného objektu uživatelského rozhraní. Rozmezí hodnoty <0, 1>.

left
: Vlastnost odkazující na nejlevější pozici daného objektu uživatelského rozhraní. Určená pouze ke čtení.

right
: Vlastnost odkazující na nejpravější pozici daného objektu uživatelského rozhraní. Určená pouze ke čtení.

top
: Vlastnost odkazující na nejvyšší pozici daného objektu uživatelského rozhraní. Určená pouze ke čtení.

bottom
: Vlastnost odkazující na nejspodnější pozici daného objektu uživatelského rozhraní. Určená pouze ke čtení.

horizontal_center
: Vlastnost odkazující na horizontální prostředek pozice daného objektu uživatelského rozhraní. Určená pouze ke čtení.

vertical_center
: Vlastnost odkazující na vertikální prostředek pozice daného objektu uživatelského rozhraní. Určená pouze ke čtení.

env
: Vlastnost odkazující na tabulku prostředí daného objektu.

Třída **UILabel** slouží k vykreslení textu v okně daného uživatelského rozhraní. Definuje následující vlastnosti:

content
: Vlastnost odkazující na řetězec obsahu daného objektu uživatelského rozhraní.

font
: Vlastnost odkazující na index fontu daného objektu uživatelského rozhraní.

UIImage

texture
: Vrátí index textury daného objektu uživatelského rozhraní.

UISelectable

select ()
: Metoda představující událost výběru daného objektu uživatelského rozhraní.

unselect ()
Metoda představující událost výběru jiného objektu uživatelského rozhraní.

click ()
Metoda představující událost kliknutí na daný objekt uživatelského rozhraní.

UIButton

texture
Vlastnost odkazující na index textury daného objektu uživatelského rozhraní.

UITextBox dědí vše od třídy UIButton, navíc pouze definuje logiku aktualizace obsahu v případě, že je textbox aktivní a je psán text pomocí klávesnice.

B.4 Seznam funkcí

Skripty událostí hry, herních objektů, objektů uživatelských rozhraní a jejich prvků mají přístup k rozhraní objektům hry, subsystémů a aplikace pomocí funkcí popsaných níže:

restart ()
: Funkce restartující hru. Při jejím spuštění se nejprve vyhodnotí skript události hry *PŘI SPUŠTĚNÍ* a dojde k nastavení výchozí úrovně, uživatelského rozhraní a sledovaného objektu.

quit ()
: Funkce, která slouží k ukončení chodu aplikace. Při jejím spuštění se vyhodnotí skript události *PŘI UKONČENÍ*.

exit ()
: Synonymum funkce *quit()*.

create_object (object, x, y) – *userdata*, *number*, *number*
: Funkce, pomocí které se do aktuální úrovně vytvoří a přidá nový objekt typu *object*.

destroy_object (object) – *userdata*
: Funkce, pomocí které se z aktuální úrovně odebere objekt odkazovaný parametrem *object*.

set_followed_object (object) – *userdata*
: Funkce, která nastaví právě sledovaný objekt pomocí jejího prvního parametru *object*.

find_object (uid) – *number*
: Funkce, která se pokusí vyhledat objekt podle zadaného *uid*.

find_objects (id) – *number*
: Funkce, která se pokusí vyhledat objekty daného typu *id*.

find_objects_by_name (name) – *string*
: Funkce, která se pokusí vyhledat objekty podle zadaného názvu *name*.

set_level (level) – *userdata*
: Funkce, která nastaví aktuální uroveň na *level*. Pro žádnou úroveň zadejte hodnotu *nil*.

set_level_fade_time (time) – *number*
: Funkce, která nastaví dobu v sekundách, která se používá pro přechod z jedné do druhé úrovně. Výchozí hodnota je 0,25.

restart_level ()
: Funkce, která restartuje nastavení objektů aktuální úrovně.

set_layout (layout) – *userdata*
Funkce, která nastaví aktuální uživatelské rozhraní na *level*. Pro žádné uživatelské rozhraní zadejte hodnotu *nil*.

set_layout_fade_time (time) – *number*
Funkce, která nastaví dobu v sekundách, která se používá pro přechod z jednoho uživatelského rozhraní do druhého. Výchozí hodnota je 0,25.

get_nearby_objects (object) – *userdata*
Funkce, která pro zadaný objekt *object* vrátí pole obsahující všechny objekty, které jsou v bezprostřední vzdálenosti.

collision_at (x, y, object) – *number*, *number*, *userdata*
Funkce, která vrátí pravdivostní hodnotu, zda-li se na dané pozici (x, y) nachází objekt typu *object*.

place_meeting (x, y, object) – *number*, *number*, *userdata*
Synonymum funkce *collision_at()*.

get_fullscreen ()
Vrátí pravdivostní hodnotu, zda-li je aplikace v režimu plné obrazovky.

set_fullscreen (value) – *number*
Nastaví pravdivostní hodnotou *value*, zda-li má být aplikace v režimu plné obrazovky nebo nikoliv.

get_resolution ()
Vrátí vektor reprezentující rozlišení okna aplikace.

set_resolution (x, y) – *number*, *number*
Nastaví velikost rozlišení pomocí parametrů x a y.

get_camera_position ()
Vrátí vektor reprezentující aktuální pozici kamery v úrovni.

set_camera_position (x, y) – *number*, *number*
Nastaví aktuální pozici kamery v úrovni pomocí parametrů x a y.

get_tile_size ()
Vrátí velikost dlaždice v pixelech.

set_tile_size (size) – *number*
Funkce, která nastaví velikost dlaždice v pixelech.

draw_text (x, y, text, font, color, alpha) – *nubmer*, *nubmer*, *string*, *nubmer*(, *userdata*, *number*)
Funkce, pomocí které může uživatel vykreslit text během skriptu události vykreslení.

draw_textured_rectangle (x, y, width, height, texture, alpha) – *nubmer*, *nubmer*, *number*, *nubmer*(, *number*, *number*)
Funkce, pomocí které může uživatel vykreslit text během skriptu události vykreslení.

set_window_fade (value) – *nubmer*
Funkce sloužící k nastavení hodnoty vykreslení zakrytí okna aplikace. Rozmezí hodnoty <0, 1>.

set_camera_boundaries (top, bottom, left, right) – *nubmer*, *nubmer*, *nubmer*, *nubmer*
Nastaví omezení pohybu kamery shora, zdola, zleva a zprava.
Výchozí nastavení je 0 pro všechny směry.

play_sound (sound) – *number*
Spustí přehrávání zvuku odkazovaného přes jeho id. Id zvuku je doplněno zápisem jeho názvu.

play_music (music) – *number*
Zastaví přehravání aktuálně hrané hudby a spustí přehrávání hudby odkazované přes její id. Id zvuku je doplněno zápisem jeho názvu.

get_master_volume ()
Vrátí hlavní hlasitost. Od tohoto čísla se odvíjí ostatní hlasitosti. Rozmezí hodnot je [0, 1].

get_music_volume ()
Vrátí hlavní nastavení hlasitosti hudby. Rozmezí hodnot je [0, 1].

get_sound_volume ()
Vrátí hlavní nastavení hlasitosti zvuků. Rozmezí hodnot je [0, 1].

is_music_playing ()
Vrátí pravdivostní hodnotu, zda-li hraje hudba.

get_sound_fade_in ()
Vrátí délku času přechodu zesilování nové hudby v sekundách.

get_sound_fade_out ()
Vrátí délku času přechodu zeslabování staré hudby v sekundách.

set_master_volume (volume) – *number*
Nastaví hlavní hlasitost. Od tohoto čísla se odvíjí ostatní hlasitosti. Rozmezí hodnot je [0, 1].

set_music_volume (volume) – *number*
Nastaví hlasitost hudby. Rozmezí hodnot je [0, 1].

set_sound_volume (volume) – *number*
Nastaví hlasitost zvuků. Rozmezí hodnot je [0, 1].

set_sound_fade_in (time) – *number*
Nastaví délku času přechodu zesilování nové hudby v sekundách.

set_sound_fade_out (time) – *number*
Nastaví délku času přechodu zeslabování staré hudby v sekundách.

stop_all_sounds ()
Zastaví přehrávání všech zvuků.

stop_music ()
Zataví přehrávání hudby.

get_mouse_click ()
Vrátí pravdivostní hodnotu, zda uživatel drží levé tlačítko myši.
Možnost použít konstanty KEY_UP a KEY_DOWN.

get_mouse_position ()
Vrátí pozici myši v okně jako ukazatel na hodnotu typu Vec2.

get_keyboard_typing ()
Vrátí řetězec, který je tvořen zadáváním kláves uživatelem.

get_keyboard_value (key)
Vrátí pravdivostní hodnotu, zda-li je klávesa *key* zmáčknutá.

get_joystick_value (button)
Vrátí pravdivostní hodnotu, zda-li je tlačítko *button* zmáčknutá.
V případě páček vrací číselnou hodnotu nastavení páčky.

get_any_button_pressed ()
Vrátí pravdivostní hodnotu, zda-li je alespoň jedna klávesa jakéhokoliv vstup aktivní.

is_using_keyboard ()
Vrátí pravdivostní hodnotu, zda-li je aktuálně používané vstupní zařízení klávesnice.

is_using_joystick ()
Vrátí pravdivostní hodnotu, zda-li je aktuálně používané vstupní zařízení joystick.

LOG (message) – *string*
Zapíše zprávu *message* do vývojářské konzole a do souboru log.txt, který se nachází v kořenové složce aktuálního projektu.

B.5 Speciální proměnné

Každý skript události má přistup k několika speciálních proměnných. První takovou je proměnná `this` – ta představuje odkaz na objekt, který právě vykonává daný skript. V případě herního objektu odkazuje `this` na strukturu herního objektu, podobně pak u uživatelských rozhraní a jejich prvcích. Avšak u objektu hry tato proměnná nemá žádný význam.

Další speciální proměnnou je `global`. Ta představuje odkaz na tabulku globálního prostředí, které je definované v objektu hry. Inicializuje se při zavolání události `POČÁTEK HRY` a přistupuje se k němu pomocí tečkové notace – tedy pokud je definovaná proměnná *variable*, pak se k ní dá přistoupit z jakéhokoliv skriptu zápisem *global.variable*.

Speciální proměnná `frame_time` je důležitá zejména při aktualizaci herních objektů, u kterých je důležité použít délku posledních snímku hry. Tato proměnná vrací hodnotu času v sekundách.

Dále proměnná `other` slouží jako odkaz na druhý objekt, který je součástí právě detekované kolize a je pouze dostupný v události kolize. Poslední speciální proměnné jsou `current_level`, `current_layout` a `followed_object`, přičemž každá odkazuje na aktuální úroveň, uživatelské rozhraní nebo sledovaný objekt.

Nakonec pro každou animaci, dlaždici, zvuk, písmo, uživatelské rozhraní, úroveň a objekt je definovaná proměnná odkazující na danou strukturu, která je pojmenovanná podle názvu daného objektu.

B.6 Seznam konstant

V poslední části následuje přehled konstant dostupných v skriptu každého herního objektu i objektu uživatelského rozhraní. Viz tabulka 2.

Tabulka 2: Tabulka konstant dostupných ve skriptu

Název konstanty	Popis
STATE_NONE	Žádný stav.
STATE_FADE_IN	Průběh zobrazení.
STATE_RUNNING	Aktivní stav.
STATE_FADE_OUT	Průběh schování.
STATE_CHANGING	Stav změny.
DIR_UP	Směr objektu nahorů.
DIR_DOWN	Směr objektu dolů.
DIR_LEFT	Směr objektu doleva.
DIR_RIGHT	Směr objektu doprava.
SOUND_CHANNEL	Číslo kanálu pro zvukové efekty.
MUSIC_CHANNEL	Číslo kanálu pro hudbu.
BUTTON_UP	Tlačítko klávesnice je puštěné.

Název konstanty	Popis
BUTTON_DOWN	Tlačítko klávesnice je zmáčknuté
BUTTON_FREE	Tlačítko je volné.
BUTTON_PRESSED	Tlačíto je právě stlačené.
BUTTON_HOLD	Tlačítko je držené.
BUTTON_RELEASED	Tlačítko bylo puštěno.
KEY_A	Číslo klávesy a.
KEY_B	Číslo klávesy b.
KEY_C	Číslo klávesy c.
KEY_D	Číslo klávesy d.
KEY_E	Číslo klávesy e.
KEY_F	Číslo klávesy f.
KEY_G	Číslo klávesy g.
KEY_H	Číslo klávesy h.
KEY_I	Číslo klávesy i.
KEY_J	Číslo klávesy j.
KEY_K	Číslo klávesy k.
KEY_L	Číslo klávesy l.
KEY_M	Číslo klávesy m.
KEY_N	Číslo klávesy n.
KEY_O	Číslo klávesy o.
KEY_P	Číslo klávesy p.
KEY_Q	Číslo klávesy q.
KEY_R	Číslo klávesy r.
KEY_S	Číslo klávesy s.
KEY_T	Číslo klávesy t.
KEY_U	Číslo klávesy u.
KEY_V	Číslo klávesy v.
KEY_W	Číslo klávesy w.
KEY_X	Číslo klávesy x.
KEY_Y	Číslo klávesy y.
KEY_z	Číslo klávesy z.
KEY_ESCAPE	Číslo klávesy escape.
KEY_ENTER	Číslo klávesy enter.
KEY_SPACE	Číslo klávesy mezerník.
KEY_TAB	Číslo klávesy tabulátor.
KEY_CAPSLOCK	Číslo klávesy capslock.
KEY_SHIFT	Číslo klávesy shift.
KEY_CTRL	Číslo klávesy ctrl.
KEY_ALT	Číslo klávesy alt.
KEY_RIGHT	Číslo klávesy šipky vpravo.
KEY_LEFT	Číslo klávesy šipky vlevo.

Název konstanty	Popis
KEY_DOWN	Číslo klávesy šipky dolů.
KEY_UP	Číslo klávesy šipky nahorů.
JOY_0	Číslo tláčítka 0
JOY_1	Číslo tláčítka 1
JOY_2	Číslo tláčítka 2
JOY_3	Číslo tláčítka 3
JOY_4	Číslo tláčítka 4
JOY_5	Číslo tláčítka 5
JOY_6	Číslo tláčítka 6
JOY_7	Číslo tláčítka 7
JOY_8	Číslo tláčítka 8
JOY_9	Číslo tláčítka 9
JOY_10	Číslo tláčítka 10
JOY_11	Číslo tláčítka 11
JOY_12	Číslo tláčítka 12
JOY_13	Číslo tláčítka 13
JOY_14	Číslo tláčítka 14
JOY_15	Číslo tláčítka 15
JOY_XBONE_A	Číslo tlačítka A.
JOY_XBONE_B	Číslo tlačítka B.
JOY_XBONE_X	Číslo tlačítka X.
JOY_XBONE_Y	Číslo tlačítka Y.
JOY_XBONE_LB	Číslo tlačítka LB.
JOY_XBONE_RB	Číslo tlačítka RB.
JOY_XBONE_SHARE	Číslo tlačítka SHARE.
JOY_XBONE_OPTIONS	Číslo tlačítka OPTIONS.
JOY_XBONE_LEFT_STICK	Číslo tlačítka levé páčky.
JOY_XBONE_RIGHT_STICK	Číslo tlačítka pravé páčky.
JOY_XBONE_LEFT_STICK_X	Číslo horizontálního nastavení levé páčky.
JOY_XBONE_LEFT_STICK_Y	Číslo vertikálního nastavení levé páčky.
JOY_XBONE_RIGHT_STICK_X	Číslo horizontálního nastavení pravé páčky.
JOY_XBONE_RIGHT_STICK_Y	Číslo vertikálního nastavení pravé páčky.
JOY_XBONE_ARROWS_X	Číslo horizontálního nastavení šipek.
JOY_XBONE_ARROWS_Y	Číslo vertikálního nastavení šipek.
JOY_XBONE_LEFT_TRIGGER	Číslo levého triggeru.
JOY_XBONE_RIGHT_TRIGGER	Číslo pravého triggeru.

Název konstanty	Popis
JOY_PS4_SQUARE	Číslo tlačítka čtverce.
JOY_PS4_CROSS	Číslo tlačítka křížku.
JOY_PS4_CIRCLE	Číslo tlačítka kolečka.
JOY_PS4_TRIANGLE	Číslo tlačítka trojúhelníku.
JOY_PS4_L1	Číslo tlačítka L1.
JOY_PS4_R1	Číslo tlačítka R1.
JOY_PS4_L2	Číslo tlačítka L2.
JOY_PS4_R2	Číslo tlačítka R2.
JOY_PS4_SHARE	Číslo tlačítka sdílení.
JOY_PS4_OPTIONS	Číslo tlačítka možností.
JOY_PS4_L3	Číslo tlačítka L3.
JOY_PS4_R3	Číslo tlačítka R3.
JOY_PS4_PS	Číslo tlačítka PS.
JOY_PS4_PAD_PRESS	Číslo tlačítka PADPRESS.
JOY_PS4_LEFT_STICK_X	Číslo horizontální hodnoty levé páčky.
JOY_PS4_LEFT_STICK_Y	Číslo vertikální hodnoty levé páčky.
JOY_PS4_RIGHT_STICK_X	Číslo horizontální hodnoty pravé páčky.
JOY_PS4_RIGHT_STICK_Y	Číslo vertikální hodnoty pravé páčky.
JOY_PS4_ARROWS_X	Číslo horizontální hodnoty šipek.
JOY_PS4_ARROWS_Y	Číslo vertikální hodnoty šipek.
JOY_PS4_DPAD_X	Číslo horizontální hodnoty DPADu.
JOY_PS4_DPAD_Y	Číslo vertikální hodnoty DPADu.
ORDER_NAME_ASC	Pořadí uložených dat podle jména vzestupně.
ORDER_NAME_DESC	Pořadí uložených dat podle jména sestupně.
ORDER_POINTS_ASC	Pořadí uložených dat podle bodů vzestupně.
ORDER_POINTS_DESC	Pořadí uložených dat podle bodů sestupně.
ORDER_TIME_ASC	Pořadí uložených dat podle času vzestupně.
ORDER_TIME_DESC	Pořadí uložených dat podle času sestupně.

Literatura

[1] GAMEMAKER STUDIO, YoYoGames: GameMaker je multiplatformní herní enginy vyvíjený firmou YoYoGames. Wikipedia [online]. 2019-3-28. [cit. 2019-04-30]. Dostupné z: https://en.wikipedia.org/wiki/GameMaker_Studio

[2] CONSTRUCT, Scirra. Construct jde herní editor založený na technologii HTML5 [online]. 2019. [cit. 2019-04-30]. Dostupné z: https://www.construct.net/en

[3] GDEVELOP: Open-source, cross-platform game creator designed to be used by everyone. [online]. 2019. [cit. 2019-04-30]. Dostupné z: https://gdevelop-app.com/

[4] VISUAL STUDIO, Microsoft: Microsoft Visual Studio je vývojové prostředí (IDE) od Microsoftu. Wikipedia [online]. 2018-12-18. [cit. 2019-04-20]. Dostupné z: https://cs.wikipedia.org/wiki/Microsoft_Visual_Studio

[5] C++: C++ je multiparadigmatický programovací jazyk. Wikipedia [online]. 2018-11-12. [cit. 2019-04-20]. Dostupné z: https://cs.wikipedia.org/wiki/C%2B%2B

[6] OPENGL, the Khronos Group: The Industry Standard For High Performance Graphics. [online]. ©2019. [cit. 2019-04-20]. Dostupné z: https://www.opengl.org/about/

[7] GLFW A multi-platform library for OpenGL, OpenGL ES, Vulkan, window and input. GitHub [online]. 2019-04-16. [cit. 2019-04-20]. Dostupné z: https://github.com/glfw/glfw

[8] FMOD STUDIO, Firelight Technologies Pty Ltd: The adaptive audio solution for games [online]. ©2019. [cit. 2019-04-20]. Dostupné z: https://www.fmod.com/studio

[9] QT, Cross-platform software development for embedded & desktop [online]. ©2019. [cit. 2019-04-20]. Dostupné z: https://www.qt.io/

[10] LUA: about. [online]. 2018-09-07. [cit. 2019-04-20]. Dostupné z: https://www.lua.org/about.html

[11] JSON, JavaScript Object Notation: je způsob zápisu dat nezávislý na počítačové platformě. Wikipedia [online]. 2018-05-06. [cit. 2019-05-07]. Dostupné z: https://cs.wikipedia.org/wiki/JavaScript_Object_Notation

[12] JSON, Nlohmann: JSON for modern C++. GitHub [online]. 2019-04-16. [cit. 2019-04-20]. Dostupné z: https://github.com/nlohmann/json

[13] SOL2, ThePhD: a C++ <-> Lua API wrapper with advanced features and top notch performance. GitHub [online]. 2019-03-29. [cit. 2019-04-20]. Dostupné z: https://github.com/ThePhD/sol2

[14] POPL: Header-only C++ program options parser library. GitHub [online]. 2018-06-24. [cit. 2019-04-30]. Dostupné z: https://github.com/badaix/popl

[15] CMAKE: Build with CMake, build with confidence. [online]. 2019. [cit. 2019-04-20]. Dostupné z: https://cmake.org/