VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PLOŠINOVÁ HRA VE 2D

2D PLATFORMER GAME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **LUKÁŠ VINCENC**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ MILET, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

144143

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Vincenc Lukáš**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Plošinová hra ve 2D**
Kategorie:	Počítačová grafika
Akademický rok:	2022/23

Zadání:

1. Nastudujte herní vývoj, herní enginy. Seznamte se s herními mechanikami plošinových her.
2. Navrhněte hru, která bude inovativním způsobem využívat herní mechaniky. Bude je různým způsobem kombinovat. Navrhněte editor úrovní.
3. Implementujte navrženou hru a herní editor.
4. Vytvořte sadu úrovní (deset a více), na kterých budou prezentovány herní mechaniky se vzrůstající obtížností.
5. Hru a editor otestujte na uživatelích a zhodnoťte kvalitu. Sepište závěry, hru zveřejněte a vytvořte video upoutávku.

Literatura:

- Gregory, Jason. *Game engine architecture*. crc Press, 2018. ISBN 1351974289, 9781351974288
- Bishop, Lars, et al. "Designing a PC game engine." IEEE Computer Graphics and Applications 18.1 (1998): 46-53.
- Adams, Ernest, and Joris Dormans. Game mechanics: advanced game design. New Riders, 2012. ISBN 0321820274, 9780321820273

Při obhajobě semestrální části projektu je požadováno:
Body jedna a dva a kostra aplikace.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Milet Tomáš, Ing., Ph.D.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	31.10.2022

Abstrakt

Hlavním úkolem této práce je vytvořit plošinovou hru ve 2D s možností vytváření vlastních úrovní. Nejdříve jsou vysvětleny základní pojmy spojené s herním vývojem. Následně je popsán konkrétní návrh výsledné hry a jeho implementace, ke které byl využit herní engine Unity kombinovaný se skripty v jazyce C#. Vlastní úrovně, jež vytvořil uživatel, jsou reprezentovány pomocí obrázků ve formátu TIFF. Hotový obrázek stačí pouze nahrát do hry a vygeneruje se z něj úroveň. Při generování je využito vlastnosti vrstvení formátu TIFF. V různých vrstvách jsou předávána jiná vstupní data pro generování.

Abstract

The main objective of this thesis is to create a 2D platformer game, that allows the user to create his own levels. Firstly, some basic terminology concerning game development is explained. Then the design and implementation of the game is described. The implementation is carried out using the Unity game engine combined with scripts in the C# programming language. User created levels are represented with the use of images in the TIFF format. These images can be simply uploaded to the game and the corresponding level is generated. The ability to create layers in the image is utilized. Each layer contains different information to be generated.

Klíčová slova

Unity, hra, 2D, C#, plošinová hra, editor úrovní, herní vývoj, formát TIFF

Keywords

Unity, game, 2D, C#, platformer game, level editor, game development, TIFF format

Citace

VINCENC, Lukáš. *Plošinová hra ve 2D.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Milet, Ph.D.

Plošinová hra ve 2D

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Mileta, Ph.D Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Lukáš Vincenc
6. května 2023

Poděkování

Tímto bych chtěl poděkovat mému vedoucímu, Ing. Tomášovi Miletovi, Ph.D, za jeho vedení, návrhy pro vylepšení a podporu při tvorbě této práce. Také děkuji Filipovi Žáčkovi, který poskytl pomoc v podobě vlastnoručně vytvořených textur pro výslednou hru.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Počítačové hry existují již desítky let a jejich počet se neustále zvyšuje. Díky každodennímu pokroku a vývoji má dnes téměř každý přístup k elektronickému zařízení, na kterém je může hrát. Výpočetní síla těchto zařízení je nesmírná a ještě před několika lety by byla nemyslitelná. Lidstvo již není tak limitováno, jak bylo dříve. S příchodem snadno přístupných nástrojů se v dnešní době může kdokoliv na svém počítači pustit do vývoje a převést svojí vizi do svého herního světa.

Tato práce má dva hlavní cíle. Prvním je navrhnout a následně implementovat počítačovou hru ve 2D prostředí, která hráče zaujme. Konkrétně se jedná o plošinovou hru. Tento žánr spočívá, jak již název napovídá, v pohybování se po mapě skládající se primárně z plošin. Tento typ her může být obohacen o různé typy překážek a přidávání speciálních schopností ústřední postavě.

Druhým cílem je připravit způsob, kterým může hráč navrhovat vlastní úrovně do vytvořené hry. Tento aspekt je další cestou, jak navnadit uživatele k navrácení se do hry. Hry tohoto typu jsou mezi komunitou hráčů velmi oblíbené, jelikož mají možnost mezi sebou své úrovně sdílet a vytvářet tím pro ostatní vlastní výzvy. Způsob stavění vlastních úrovní ovšem musí být jednoduchý na pochopení a uživatelsky přívětivý.

V této práci je nejdříve představena vytvořená hra a jsou uvedeny hry, které posloužily jako inspirace. Následně jsou vysvětleny základní pojmy, které je potřeba znát pro pochopení zbytku práce. Dále je dopodrobna popsán návrh. Ten se zabývá jednotlivými aspekty hry a jejich výslednou funkcionalitou. Nakonec je popsáno, jak byl popsaný návrh realizován a implementován pomocí herního enginu Unity.

Kapitola 2

Představení hry

Produktem této práce je hra s názvem *Skinwalker*. Jedná se o 2D plošinovou hru zasazenou do sci-fi světa, ve které hráč přebírá roli postavy se speciálními schopnostmi a vyhýbá se nástrahám okolního světa.

Obrázek 2.1: Pohled na hru *Skinwalker*

2.1 Cíl hry

Hráč překonavá mapy a snaží se dostat do jejich cíle. Po cestě naráží na různé nástrahy, od zpomalovacích bloků, přes obrácení gravitace celého světa, až po smrtící ostny, které jej vrátí na samotný začátek úrovně. Tyto překážky mu mohou buďto uškodit, nebo značně pomoct. Všechny pasti se dají překonat, a to například pomocí speciálních schopností postavy. Hráč musí správně zvolit jakou schopnost má v danou chvíli využít. Hlavním cílem je úspěšně dokončit všechny mapy.

2.2 Editor úrovní

Hra dále nabízí uživateli možnost vytvoření vlastní úrovně. K tomu stačí v počítači nakreslit obrázek pomocí libovolného grafického editoru a ve hře jej otevřít. Jednotlivé pixely tohoto souboru se podle jejich barev následně vygenerují jako různé typy bloků. Po načtení

mapy ve hře má uživatel možnost svojí úroveň proběhnout. Zároveň má k dispozici tlačítko „Reload", pomocí kterého jednodušše načte poslední změny v obrázku a propíše je do mapy. Pouhým sdílením tohoto souboru může úroveň vyzkoušet kdokoliv. Tato možnost je cílena i na uživatele bez větších technických schopností.

2.3 Hry s podobnými koncepty

Tato sekce uvádí několik her, které posloužily jako inspirace k vytvoření hry *Skinwalker*.

Mekazoo[1]

Mekazoo je 2D hra, ve které hráč překonává překážky pomocí několika speciálních schopností jeho postavy. Tyto schopnosti je možné kdykoliv přepnout, a díky tomu si hráč může vybrat tu nejvhodnější pro daný scénář. Dané schopnosti jsou vyobrazeny zvířaty. Například klokan reprezentuje skok do výšky a pelikán poskytuje schopnost létání.

Tato hra posloužila jako primární inspirace pro vytvoření produktu této práce. Hlavní herní mechanika, spočívající v přepínání speciálních schopností, je přítomna i v této hře.

Obrázek 2.2: Promo snímek ze hry Mekazoo. Na obrázku jsou vyobrazeny všechna zvířata, jejichž schopnosti může hráč nabrat. Snímek převzat z internetu.[2]

Celeste[3]

Jedná se o hru pro jednoho hráče. Cílem je ústřední postavu Madeline probojovat přes mnoho úrovní a dostat jí na vrchol hory Celeste. Po své trase naráží na všelijaké překážky a speciální bloky, které musí hráč překonávat. K dispozici má herní mechaniky jako lezení po stěnách, nebo sprint.

Pro hru *Skinwalker* posloužila jako inspirace primárně svým uměleckým stylem, kterým je pixel art. Způsob osvětlení map poskytl značné podněty pro tuto hru, která využívá podobný systém světel. Zároveň některé speciální bloky v této práci vychází z *Celeste*.

[1] https://store.steampowered.com/app/390330/Mekazoo/
[2] https://twitter.com/Simply_Xbox/status/808939621367369728/photo/1
[3] https://store.steampowered.com/app/504230/Celeste/

Obrázek 2.3: Snímek ze hry *Celeste*. Hra využívá různě barevná osvětlení a každé z nich vychází z jednoho bodu. Na obrázku jsou zároveň vidět pohyblivé platformy, které byly inspirací pro vytvoření podobných platforem pro hru *Skinwalker*. Obrázek převzat z internetu.[4]

Dead Cells[5]

Tato hra se také řadí mezi plošinovky, ovšem je více zaměřená na akci. Hráč prozkoumává úrovně a bojuje s nepřáteli, které po cestě potkává. *Dead Cells* přidávala při implementaci produktu této práce drobné podněty po stránce pohybu postav a plošin. Nejvíce ovšem posloužila pro inspiraci z vizuální stránky a *Skinwalker* se lehce snaží reprodukovat její vzhled.

Obrázek 2.4: Snímek ze hry Dead Cells. Hra využívá stylu pixel art, který se tato práce snaží do jisté míry reprodukovat. Obrázek převzat z internetu.[6]

[4] https://www.fakaheda.eu/images/product_medias/6150-celeste-10.jpg
[5] https://store.steampowered.com/app/588650/Dead_Cells/
[6] https://pbs.twimg.com/media/FqOEovIWIAI0x3b?format=jpg&name=4096x4096

Kapitola 3

Teorie

3.1 Hra

„Hra“ je velmi široký pojem. Raph Koster ji ve své knize *A Theory of Fun for Game Design* [5] popisuje jako interaktivní zážitek, který hráči poskytuje výzvy s postupně zvyšující se náročností, jež se postupně naučí a ovládne. Lze mluvit o hrách deskových, sportovních, či videohrách, na které se tato práce zaměřuje.

Videohra je taková hra, která se vyskytuje na elektronických zařízeních, například stolních počítačích, konzolích, nebo mobilních telefonech. Cambridge Dictionary jí označuje jako hru, kde uživatel ovládá pohybující se obraz mačkáním tlačítek [3]. Nemusí se ovšem jednat jen o kratochvíli, ale například i o formu umění.

3.2 2D plošinové hry

Obrázek 3.1: Pravděpodobně nejznámnější série plošinových her je o Mariovi, instalatéru s cílem zachránit unešenou princeznu. Na své cestě překonává mapy s rozestavenými plošinami a při tom se vyhýbá nepřátelům. Obrázek převzat z internetu.[1]

[1] https://www.datasciencecentral.com/wp-content/uploads/2021/10/super-mario-bros-1-1-i20783.jpg

Videohry mohou být mnoha různých žánrů. Jelikož produktem této práce je 2D plošinová hra, je následující výklad zaměřen právě na tento žánr. Jak již její název napovídá, odehrává se plně ve dvoudimenzionálním prostředí. To znamená, že hráč může postavou pohybovat pouze horizontálně a vertikálně, nikoliv však dále, či blíže ke kameře.

Pojem „plošinová hra" vychází ze vzhledu herního prostředí. Hráč se ve většině případů naviguje světem, ve kterém jsou rozmístěny plošiny. Tento typ her může být dělen na další podžánry. Některé mohou být zaměřeny více na akci a jejich hlavní herní mechanikou je souboj s nepřáteli. V jiných je hráčovým úkolem vyhýbat se překážkám rozmístěným po mapě a dostat se do cíle.

3.3 Herní vývoj

Proces vývoje počítačových her není pevně nadefinovaný a každý tým si jej může uzpůsobit podle svých potřeb. I přesto se dá shrnout do několika bodů, které lze vidět v procesu vytváření většiny her. Různé zdroje uvádí lehce odlišné kroky, hlavní myšlenka ovšem zůstává stejná. Tato podkapitola vychází primárně ze dvou zdrojů. Jedním je internetový článek Rosse Brambla [2] uvádějící 7 hlavních fází vývoje, které jsou popsány i zde. Druhým je kniha *Blood, Sweat and Pixels* [7] Jasona Schreiera popisující proces a úskalí, která potkala vývojáře již existujících známých her.

Většina vývojových týmů má velmi podobnou strukturu a jeho velikost je závislá na rozpočtu. Tento tým mívá producenta, který se stará o hladký průběh vývoje. Jsou potřeba lidé starající se o příběh hry, využité zvuky, vizuální styl hry a návrhy map. Dále se v týmu nachází programátoři, kteří veškeré návrhy implementují do samotné hry a testeři snažící se odhalit co nejvíce chyb ještě před vydáním hry. V neposlední řadě je nutné mít i někoho, kdo se bude starat o marketing a bude dostávat výslednou hru do povědomí veřejnosti. [7]

Výše popsaná struktura se vztahuje primárně na týmy pracující pod větším herním studiem. Ovšem existují i tzv. *indie vývojáři*, což mohou být buďto jedinci pracující o samotě, nebo velmi malé skupiny lidí. V takovém případě musí samozřejmě jedna osoba nabírat více rolí a úkolů. Takovým vývojářům totiž chybí finanční podpora velkého herního studia a v důsledku toho nemají zdroje k zaplacení dalších lidí. Nicméně níže popsaný proces herního vývoje lze aplikovat i na ně.

Samotná stádia vytváření hry poté vypadají následovně: [2]

- **Plánování** – Samotný začátek vývoje, ve kterém je potřeba vymyslet přibližně o čem hra bude, jaké prostředky budou využity, základní příběh apod. Jsou položeny základy, na kterých se bude výsledná hra stavit.

- **Předprodukce** – Jednotlivé skupiny týmu se musí domluvit a sjednotit v tom, co přesně bude jejich cílem práce. Například se radí o tom, jakou barevnou paletu bude hra využívat, navrhují fyzikální zákony a jak budou implementovány, nebo upřesňují detaily příběhu.

- **Produkce** – Tato fáze je ze všech nejdelší. Zde se uskutečňuje vše, co bylo v předchozích stádiích navrženo. To znamená kupříkladu, že programátoři implementují společně s návrháři výsledný svět, umělci vytváří textury a zvukaři nahrávají zvukové efekty.

- **Testování** – Zde přichází na řadu testeři, kteří musí projít do detailu každý aspekt a funkcionalitu výsledné videohry. Pokud někde objeví chybu, předají ji vývojářům a ti

ji musí opravit. Tento proces se točí do chvíle, kdy se hra dostane do funkční podoby a může se vypustit do veřejnosti.

- **Zveřejnění bety** – Některé hry využijí takzvaného *Early Access*, což znamená, že vybranému vzorku lidí je zpřístupněna beta verze hry ještě před oficiálním vydáním. V tuto chvíli dostane tým první zpětnou vazbu od lidí, kteří nemají s vývojem nic společného. Přípomínky od uživatelů jsou zapracovány a jsou opraveny další nalezené chyby.

- **Oficiální spuštění** – Veřejnost dostane možnost hru zakoupit a stáhnout. V ideálním případě je hra již zfinalizovaná a všechny chyby opraveny. To je ovšem velmi vzácný stav.

- **Postprodukce** – Zde přichází na řadu oprava chyb nalezených po oficiálním spuštění. Podle toho, o jakou hru se jedná, můžou být vytvořeny takzvané DLC (Downloadable Content). Většinou to jsou nové úrovně a příběhy, které si může uživatel zakoupit.

3.4 Herní engine

Počítačových her bylo vytvořeno již mnoho. Samozřejmě je možné pro každou novou hru začít veškerou funkcionalitu implementovat od začátku, avšak není to jakkoliv efektivní. Aby si vývojáři ušetřili práci a námahu, přišli s konceptem herních enginů. Ty jim umožní přepoužívat různé komponenty z již existujících her a modifikovat je jak uznají za potřebné.

Jason Gregory ve své knize *Game Engine Architecture* [4] popisuje jejich původ. Pojem „herní engine“ pochází již z 90. let minulého století. V té době byla vydána počítačová hra *Doom*[2] společností *id Software*, u které bylo z vývojářského hlediska dobře vidět rozdělení jednotlivých komponent, sady herních pravidel a grafické části. Poté bylo možné pro vývojáře znovu používat již existující komponenty pro další hry a ulehčit si tím značně práci.

Ke konci devadesátých let byly vytvářeny hry, u kterých se již kladl důraz na znovupoužitelnost. Jako příklad lze uvést sérii videoher *Quake*[3], která využívala skriptovací jazyk *QuakeC* vytvořen speciálně pro ni. Tento jazyk umožnil jednoduché přidávání nových zbraní do hry, měnění logiky hry a jejích pravidel.

Engine je tedy jakési srdce hry, které poskytuje základní komponenty, jako například renderování scény a její zobrazení, zachytávání vstupu, nebo fyzikální zákony. Tím pádem nemusí vývojář programovat veškerou funkcionalitu od začátku a může si jí podle potřeby své hry modifikovat. Existuje mnoho různých herních enginů. Mezi nejznámnější patří Unity, Unreal Engine, Godot, nebo CryEngine. Pro hru popsanou v této práci byl použit engine Unity.

3.5 Herní mechaniky

Herní mechaniky jsou jádro hry. Definují její pravidla, jak reaguje hra na vstup od hráče nebo jaké akce vyústí ve výhru, či porážku. [1] Hráč musí tyto mechaniky pochopit a ovládnout pro úspěšné zdolání hry. Lze sem zařadit např. zda má hráč nějaké speciální schopnosti a jak na něj působí, jak ovlivňuje hráče okolní svět a mnoho dalšího.

[2] https://cs.wikipedia.org/wiki/Doom
[3] https://en.wikipedia.org/wiki/Quake_(video_game)

3.6 Editor úrovní

Některé hry umožňují hráči vytvářet vlastní mapy a úrovně. Jakým způsobem je toho dosaženo záleží už na návrhu samotné hry. Ve většině případů poskytuje vlastní uživatelské rozhraní, ve kterém si člověk poskládá z palety bloků výslednou mapu. Tyto editory mohou fungovat jak ve 2D, tak 3D prostředí.

Dalším způsobem, využívaným u 2D her, je reprezentace úrovně podle obrázku. Hráč jej nakreslí v externím nástroji pro malování a nahraje ho do hry. Podle předem definovaných pravidel se z načteného obrázku vygeneruje úroveň, kterou může uživatel testovat. Tento způsob je přítomný i u této práce a využívá obrázky ve formátu TIFF.

3.7 Formát TIFF

Souborový formát TIFF, také známý jako *Tag Image File Format*, je používán k ukládání rastrové počítačové grafiky. Není stvořený specificky pro jednu konkrétní platformu a proto je snadno přepoužitelný a kompatibilní napříč různými zařízeními. Má rozsáhlý seznam vlastností, které mohou být výhodou pro různé oblasti. Jako příklad lze uvést podporu různých metod komprese. Pro účely této práce je ovšem nejdůležitější znát jeho možnost vrstvení.

Obrázek ve formátu TIFF se může skládat z různých vrstev, z nichž každá obsahuje jiná obrazová data. Dohromady poté dají celistvý obraz, ovšem informace o jednotlivých vrstvách v souboru zůstane zachována.

Již v roce 1986 byla vydána první verze tohoto formátu. V té době se ovšem jednalo pouze o černobílé obrázky. S příchodem TIFF 4.0 o rok později se dostavila podpora RGB barev. V červnu 1992 byla zveřejněna verze TIFF 6.0, která je dnes nejběžněji používaná. V ní totiž byla dodána podpora barevných palet CMYK a YCbCr. Existuje mnoho dalších rozšíření, ovšem ta již nejsou tak běžná. [6]

3.8 Unity

Unity je jedním z nejvíce rozšířených herních enginů, používaným obrovským počtem vývojářů. Oproti ostatním enginům je tento často vyhledáván začátečníky v herním vývoji, jelikož je pro ně přívětivější. Poskytuje zároveň vývojové prostředí se snadno pochopitelným uživatelským rozhraním. Jen toto prostředí umožňuje vytváření her aniž by bylo potřeba sepsat tisíce řadků kódu. Uživatel jej může využít ke stavbě úrovní, editaci vlastností využitých bloků, správě jejich animací atd.

Unity byl dříve známý pod názvem „Unity3D“, ovšem postupem času se z něj stal i velmi dobrý nástroj pro 2D vývoj. V dnešní době valná většina vývojářů mobilních her sáhne právě po tomto enginu, hlavně jelikož umožňuje vývoj i pro zařízení se slabší výpočetní silou. Samozřejmě je možné jej využít i pro vývoj výpočetně náročných her, ovšem pro ty je častěji využit například Unreal Engine nebo CryEngine. [8]

Ačkoliv se o Unity mluví jako „herním enginu”, nepoužívá se pouze pro vývoj počítačových a mobilních her. Na jeho webových stránkách[4] je označen zároveň jako nástroj pro architektonické nebo automobilové vizualizace. Také je uvedena možnost renderování filmových efektů.

Pro pochopení používání Unity je potřeba znát několik pojmů, které jsou popsány níže.

[4] https://unity.com/

GameObject

V Unity je `GameObject` základním stavebním blokem každé entity ve scéně. Může se jednat o postavu hráče, jakýkoliv blok nebo pozadí scény. Lze k němu připojit libovolné komponenty, které mu dodají kolizní hranice, aplikují na něj fyzikální zákony nebo z něj udělají zdroj světla.

Každý objekt má svojí pozici, velikost a rotaci ve světě. Také může být organizován do rodičovské hierarchie a obsahovat dětské objekty. Koncept `GameObject` umožňuje vývojáři vytvářet komplexní scény spravováním jednotlivých entit.

Prefab

Uživatel si vytvoří `GameObject` s několika komponentami, které si ponastavuje dle potřeby. Tento objekt může využít ve scéně. Avšak pokud ho chce do scény přidat znovu, musí ho buďto zkopírovat, nebo nastavit znovu od začátku. K ulehčení přepoužitelnosti objektů existují *prefaby*. Prefab je možné vytvořit z jakéhokoliv objektu. Uchovává si všechny jeho komponenty tak, jak byly nastaveny. V případě, kdy uživatel upraví některou z jeho komponent, propíše se tato změna do všech instancí, kde je prefab použit.

Kapitola 4

Návrh hry

Tato kapitola se zaměří na návrh hry a jednotlivých mechanik v ní využitých.

4.1 Pravidla hry

Postava se objeví v počátečním bodě mapy a pro její dokončení musí doběhnout do cíle. S postupem hry a dosažením nových úrovní se tento úkol stává náročnějším. Na cestě jsou rozmístěny různé speciální bloky, které mohou hráči buď uškodit, a nebo mu naopak pomoci k dokončení mapy. Všechny typy těchto bloků jsou popsány níže.

4.2 Prostředí hry

Herní úrovně se odehrávají v temných místnostech se sci-fi nádechem. Ve hře jsou k dispozici dva typy osvětlení a každá z map má definované své vlastní. Prvním typem je globální osvětlení, které osvětluje celou mapu stejnou intenzitou a zajišťuje rovnoměrnou viditelnost ve všech jejích oblastech. Druhým typem je bodové osvětlení. To znamená, že se na mapě vyskytuje několik svítidel. Jejich počet je pro každou uroveň jiný a je omezen pouze velikostí mapy. Každý z takových zdrojů světla ozařuje pouze malý prostor okolo sebe a čím dále je bod od zdroje, tím méně je viditelný. Barva bodového osvětlení může být libovolná a není závislá na barvách okolních světel. To znamená, že se v jedné úrovni může naráz vyskytovat kombinace různě barevných svítidel.

V případě použití bodových svítidel mohou na mapách vznikat temná místa, kam není vidět. Ovšem aby měl hráč nějakou možnost do nich nahlédnout, je i on sám zdrojem světla. Tím pádem, kamkoliv jde, vždy vídí na určitou vzdálenost kolem sebe, ať už je oblast osvětlená či nikoliv. Navíc může využít i speciální schopnost, která mu umožňuje vidět dále, než za normálních podmínek. Tato možnost je více dopodrobna rozebrána v kapitole o speciálních schopnostech.

4.3 Pohyb postavy

Při zmáčknutí šipky pro pohyb po horizontální ose postava nezrychluje postupně, ale zrychlí ihned na maximální rychlost. Jakmile hráč šipku pustí, postava neprodleně zastaví. Oproti tomu skok se chová tak, aby lépe reprezentoval reálný skok. Stisknutí mezerníku vyvolá výskok a postavě je přidána síla směrem nahoru. To znamená, že se vymrští určitou rychlostí, která se postupně snižuje, než začne postava opět padat. Obyčejným výskokem se může dostat až dva bloky do výšky. Vyskočit může pouze ve chvíli, kdy stojí na zemi, aby nemohl skákat do nekonečna.

Chování pohybu postavy může být ovlivněno jeho speciálními schopnostmi a okolními bloky. Konkrétní případy jsou více popsány v kapitolách níže.

4.4 Kamera

Aby mohl hráč sledovat, co se ve virtuálním světě děje, musí mít scéna připravenou nějakou kameru, která mu zprostředkovává hlavní dění. Pro hru *Skinwalker* je kamera navržená tak, aby hráč vždy viděl postavu, kterou ovládá. Kamera se snaží držet postavu vždy uprostřed obrazovky. Avšak v případě, že postava doběhne ke kraji mapy, kamera se musí zastavit, jelikož se dále nic nevyskytuje. Pokud by hráč měl vypadnout za hranice mapy, ihned zemře. Kamera jej poté dále nesleduje, ale zůstane na posledním místě, kde byla postava vidět a byla naživu.

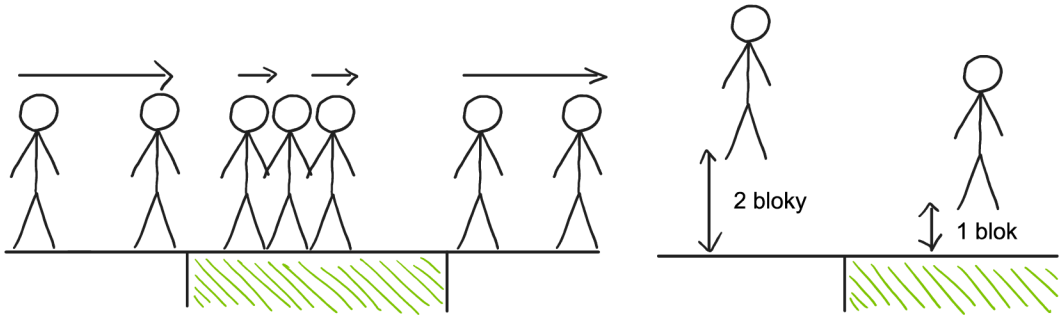
Ze začátku úrovně je kamera nastavena tak, aby byla vidět co největší část mapy. Některé ovšem mohou být příliš velké a kdyby měla být na obrazovce vidět celá, postava by byla velmi malá a špatně viditelná. Z tohoto důvodu může hráč pomocí kolečka na myši mapu přibližovat a oddalovat. Zároveň má kamera přidané omezení velikosti kamery podle počtu viditelných bloků na výšku. Velikost se může pohybovat mezi 8 a 40 bloky.

4.5 Speciální bloky

Zpomalovací blok

Jedná se o jednu z nejobvyklejších překážek ve 2D plošinových hrách. Jakmile přijde hráč do kontaktu s tímto blokem, jeho rychlost se značně sníží, a to přesně na polovinu. Pokud v danou chvíli využívá speciální schopnost, která mu jakkoliv upravuje základní rychlost, bude se po tomto bloku pohybovat poloviční rychlostí oproti této pozměněné rychlosti.

Kromě zpomaleného pohybu je postava omezená i při skákání. Výše jeho skoku, ať už obyčejného, nebo vylepšeného schopností, bude poloviční. Původní rychlosti a výšky skoku hráč opět dosáhne až ve chvíli, kdy zpomalovací blok opustí a postaví se na jiný blok.

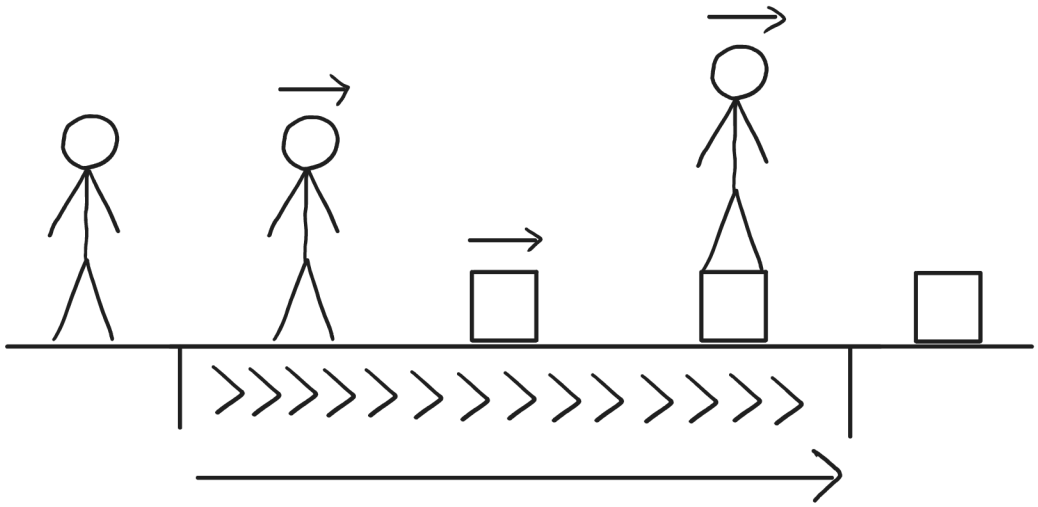


Obrázek 4.1: Návrh zpomalovacího bloku. Zeleně zvýrazněná oblast odpovídá tomuto bloku. Levá část návrhu reprezentuje zpomalení rychlosti hráče a pravá představuje výšku skoku hráče.

Pohyblivý pás

Tento blok ovlivňuje horizontální pohyb postavy. Pokud postava stojí nehybně na pohyblivém pásu, je posouvána směrem jeho orientace. Stejným způsobem jsou ovlivněny i krabice, které jsou na něm umístěny. Pokud postava vyskočí na krabici, která se v tu chvíli pohybuje na pásu, bude také ovlivněna. To znamená, že oba objekty budou posouvány společně.

Pohyblivý pás má vliv také na rychlost běžící postavy. Pokud utíká po směru orientace posunu, jeho rychlost se zvýší. V opačném případě, tedy když hráč běží v protisměru, se jeho rychlost sníží. Každopádně i tak bude mít hráč stále možnost doběhnout na začátek pásu. To znamená, že pokud omylem nastoupí na pás a začne být posouván špatným směrem, může se bez větších potíží vrátit zpět.

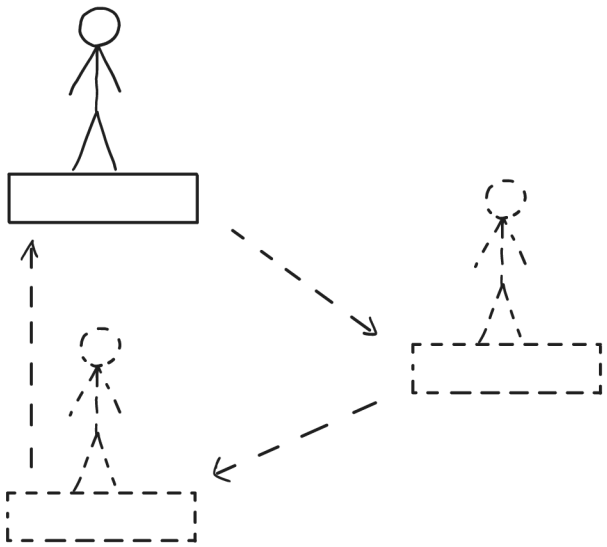


Obrázek 4.2: Návrh pohyblivého pásu. Spodní šipka naznačuje směr pásu, který je vyšrafován černou barvou. Vše, co je umístěno na pásu, se v daném směru pohybuje. Ostatní entity zůstávají na místě. Postava stojící na krabici je posouvána společně s danou krabicí.

Pohyblivé platformy

Pohyblivé platformy mají na hráčovu postavu obdobný vliv, jako výše zmíněný pás. Nejpodstatnějším rozdílem je to, že zatímco obyčejný pás posouvá hráče vodorovně pouze jedním směrem, platforma se může pohybovat libovolným směrem, a to jak vertikálně tak i horizontálně.

Takové platformy putují mezi několika předem určenými body v pevně zvoleném pořadí. Hráč na ně může naskočit a nechat se převézt. Takovéto platformy mohou být libovolných rozměrů a množství bodů, mezi kterými cestují, je omezené pouze velikostí mapy.

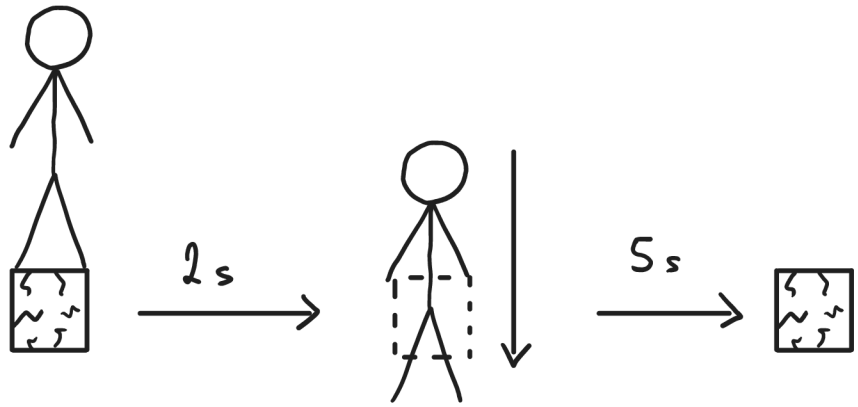


Obrázek 4.3: Návrh pohyblivých platforem. Jedná se o příklad platformy, která se pohybuje mezi třemi body. Postava zůstává stále na platformě a samovolně z ní nespadne.

Křehký blok

Jakmile vstoupí postava na křehký blok, začne tento blok postupně mizet. Po krátké chvíli zmizí úplně a pokud na něm hráč zůstal stát, propadne se dolů. V onom malém časovém okně má chvíli na to z bloku utéct. Je tedy donucen jednat v časovém presu.

Následně se po delší chvíli blok znovu objeví a hráč na něj opět může nastoupit. Pokud tak učiní, začne blok znovu mizet a celý proces se opakuje.

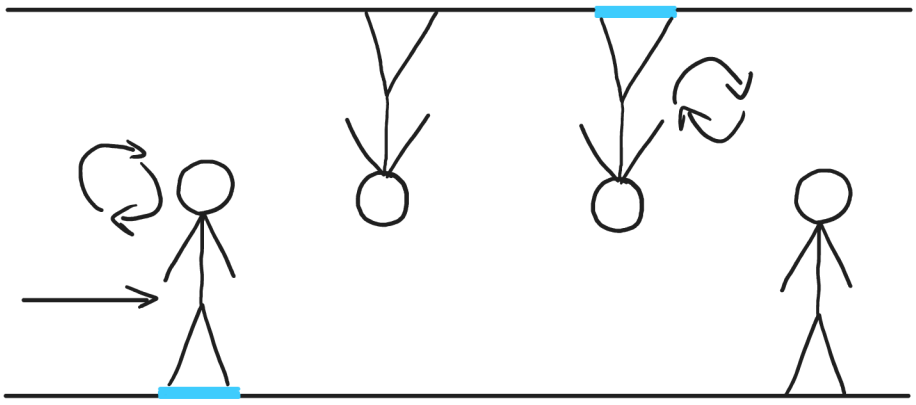


Obrázek 4.4: Znázornění časového cyklu křehkého bloku. Dvě vteřiny poté, co hráč na blok nastoupí, blok zmizí a hráč se jím propadne. Po pěti vteřinách se znovu objeví.

Gravitační blok

Ve chvíli, kdy hráč nastoupí na gravitační blok, obrátí se přitažlivost celého světa. Podle návrhu úrovně potom může být účel tohoto bloku různý. Buďto může hráče dostat do pozice, ze které není možné uniknout a tím ho donutit začít úroveň od začátku, nebo ho dostane do míst, kam by se normálně nedostal a přiblíží ho k cíli.

Krabice po kontaktu s tímto blokem také spustí změnu gravitace. Vždy jsou ovlivněny stejně jako hráč, tedy jejich gravitace se změní stejným způsobem.



Obrázek 4.5: Návrh gravitačních bloků. Mohou být umístěny na spodní i horní část mapy. Po nástupu na blok se postava otočí o 180 stupňů a gravitace světa se změní.

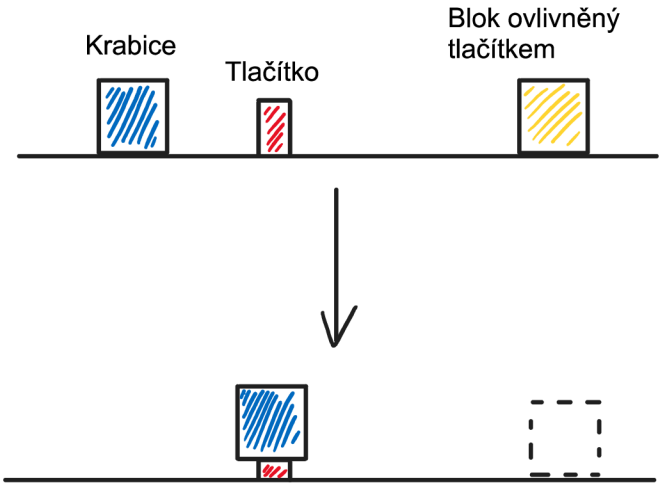
Krabice

Krabice je volně posouvatelný prvek na mapě. Je čtvercového tvaru a hráč jej může postrkovat svým tělem. Ovlivňují a zároveň jsou ovlivněny gravitačními bloky, což znamená, že při změně gravitace jsou stejně jako hráč přitahovány k opačné straně světa.

Pro hráče mají velkou výhodu pro překonávání map s elektřinou, jelikož je mohou využít ke zmáčknutí tlačítek a odblokovat tak překážku, která mu stála v cestě. Podobně mu může pomoct v situaci, kdy potřebuje vyskočit na blok umístěný příliš vysoko. Stačí si tuto krabici posunout na potřebné místo a vyskočit z ní.

Tlačítka

Tlačítka představují do hry prvek elektřiny. Skokem, či přiběhnutím na tlačítko jej hráč zmáčkne. Tímto se viditelně změní i vzhled tlačítka aby reprezentoval, že je ve stlačeném stavu. Toto tlačítko pak může ovlivňovat libovolný počet bloků. Hráči může některý z bloků, který je ovlivněn tlačítkem, bránit v cestě. Proto musí přijít na to, které tlačítko a jakým způsobem stlačit, aby si uvolnil cestu. K tomu využije ve valné většině případů výše zmíněné krabice, které také může použít jako zátěž držící tlačítko ve stlačené poloze.



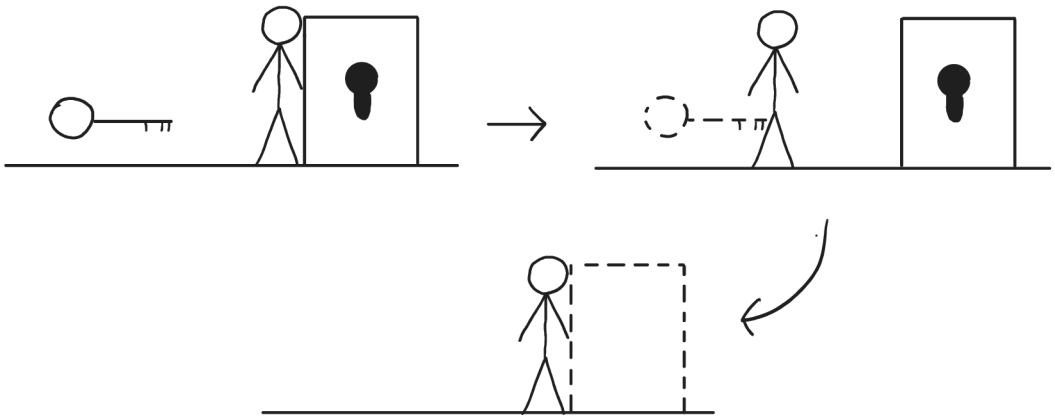
Obrázek 4.6: Návrh fungování elektřiny. Modrý čtverec reprezentuje posouvatelnou krabici a žlutý čtverec je blok, který po zmáčknutí tlačítka zmizí. V tomto konkrétním případě je pro stisk použita krabice, avšak hráč jej může stisknout i svým tělem.

Klíče a dveře

Dalším typem překážky, se kterým se může hráč setkat, jsou zamknuté dveře. Tyto dveře může otevřít pouze pokud dříve sebral klíče k těmto konkrétním dveřím. V takovém případě mu stačí ke dveřím přiběhnout a otevřou se mu automaticky. Pokud odpovídající klíče ovšem nesebral, dveře zůstavají po kontaktu s hráčem zavřené.

Dveří se může nacházet na mapě několik a každé z nich potřebují různé klíče. Hráč nemůže použít jeden klíč k otevření různých dveří, avšak k otevření jedněch dveří může hráč potřebovat klíčů více. Počet takových klíčů může být libovolný a záleží na návrhu mapy. Aby bylo hráči jasné, ke kterým dveřím patří konkrétní klíč, vyzařují oba objekty obarvené světlo. Skupina klíčů a dveří k nim patřící je takto odlišena jednou barvou.

Pro vytvoření jisté výzvy pro hráče mohou pak klíče být uloženy na těžko dosažitelných místech, což hráče donutí využít dalších speciálních bloků, případně svých schopností.



Obrázek 4.7: Návrh klíčů a dveří. Bez sebrání klíče se při kontaktu hráče se dveřmi nestane nic. Jakmile klíč zvedne, dveře se mu otevřou.

Ostny

Poslední překážkou pro hráče jsou ostny. Pokud s ním přijde hráč do kontaktu, neprodleně zemře a je donucen opakovat úroveň od úplného začátku. Proto musí dělat vše, co je v jeho silách, aby se jim vyhnul. Hráčův skok je dostatečně vysoký, aby jej mohl přeskočit. V případě, kdy jsou rozmístěny tak, že přeskočení není možné, může využít svojí speciální schopnost. Ta mu umožní osten zničit pouhým dotykem. O této schopnosti bude více níže.

4.6 Schopnosti postavy

Pro zdolání úrovně může hráč využít svých speciálních schopností. Jedná se o nejdůležitější mechaniku této hry, kterou je potřeba pro dokončení hry ovládnout. Na výběr má ze 4 možností popsaných v této kapitole.

Zvýšená rychlost

Zapnutí této schopnosti ovlivní hráčovu rychlost, a to tak, že se zvětší o polovinu. Dále záleží na hráči, jak této schopnosti využije. Může mu pomoct k překonání velké vzdálenosti na rovné ploše pro ušetření času. Primárně mu ale poslouží k překonání dlouhých propastí. Zvýšená rychlost totiž znamená, že při rozběhu ke skoku postava nabere více energie. Díky tomu doskočí dále, než by se mu podařilo s jeho obyčejnou rychlostí.

Tato schopnost ovšem není určena pro neustálé využití. Hráči značně ztíží přesnost a například pro skákání na krátké vzdálenosti po malých platformách je vhodnější schopnost vypnout.

Vysoký skok

Tato schopnost přidá sílu k hráčovu skoku. Oproti normálnímu stavu, kdy hráč vyskočí 2 bloky do výšky, se s touto schopností dostane až 5 bloků do výšky. Tento speciální skok je určen primárně k dosažení míst, kam by se hráč obyčejně nedostal. Díky tomuto také doskočí horizontálně dál, ovšem pro skok do dálky se mu vyplatí více použít výše zmíněnou

schopnost přidávající mu rychlost. Podle situace musí hráč sám zhodnotit, kterou z nich využít.

Nesmrtelnost

Na mapě se mohou objevovat ostny, kterých se hráč nesmí dotknout, jelikož by způsobily smrt jeho postavy a restart úrovně. Nicméně někdy se mohou objevit místa, která nelze s obyčejnou postavou projít nebo přeskočit, jelikož se na nich vyskytuje příliš mnoho ostnů. Pro tato místa byla postavě přidána schopnost nesmrtelnosti. Pokud přijde hráč do kontaktu s ostnem a má zapnutou tuto schopnost, nezemře, ale naopak zničí daný osten. Tímto si může pročistit cestu, kterou by jinak nemohl projít. Zničené ostny se do konce úrovně již neobjeví.

Pokročilejší hráči mohou tuto schopnost využít například jako poslední záchranu. V případě, že spadnou z některé platformy a míří na ostny, ještě za letu mají možnost přepnout na nesmrtelnost. Jestliže tak udělají včas, zachrání se. Avšak aby neměl hráč po celou dobu hraní tuto schopnost zapnutou, přidává mu značné snížení rychlosti, a to přibližně na třetinu. Proto by měl mít tuto schopnost aktivní pouze ve vzácných případech.

Noční vidění

V neposlední řadě má hráč k dispozici schopnost nočního vidění. Některé mapy mohou být temnější a mít pouze pár malých zdrojů světla. Mimo osvětlené části těmito světly je viditelnost silně omezená. V základu vidí postava bez problému do vzdálenosti přibližně pěti bloků okolo sebe, a to i v neosvětlených částech. Pokud se potřebuje dostat do neosvětlené části, má možnost do ní předem nahlédnout použitím této schopnosti. Ta značně rozšíří okruh hráčova vidění a dříve neosvětlené prostory již vidí.

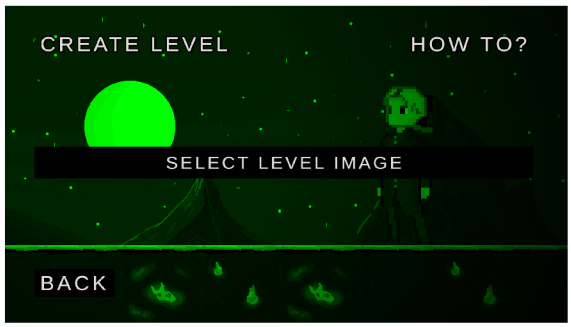
Stejně, jako pro schopnost nesmrtelnosti, i zde je hráč při aktivním používáním nočního vidění značně znevýhodněn. Jeho rychlost se také sníží na třetinu. Hlavní použití této schopnosti je určeno pro pouhé nahlédnutí do temných oblastí. Hráč si musí rozložení mapy zapamatovat a podle toho pokračovat.

4.7 Uživatelské rozhraní

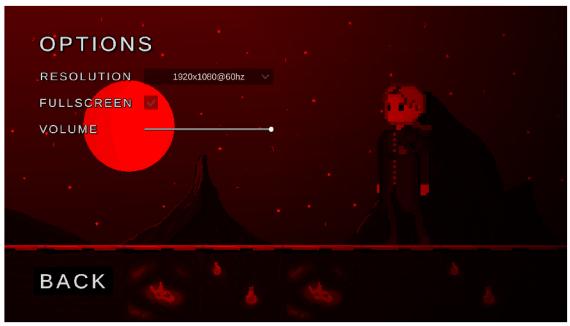
Obrázek 4.8: Hra *Skinwalker* využívá poměrně minimalistické uživatelské rozhraní. Při zapnuté úrovni vidí uživatel pouze samotnou hru a ikony, ukazující kterou schopnost má momentálně zapnutou, na spodní části obrazovky. Na tomto příkladu využívá schopnost nesmrtelnosti.

Hlavní menu

(a) Hlavní menu



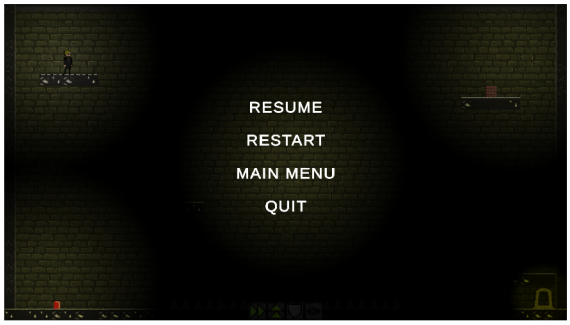
(b) Navazující menu pro výběr obrázku



(c) Navazující menu s nastavením

Obrázek 4.9: První, s čím se uživatel při zapnutí hry setká, je hlavní menu a je důležité, aby ho na první pohled zaujalo. Pro tuto hru bylo vytvořeno jednoduché menu s několika tlačítky, které navigují hráče do navazujících menu. Za tlačítky je postava hry, kterou může hráč ovládat, jako by jí mohl ovládat v úrovních. Jediným rozdílem je, že nemůže používat speciální schopnosti. Jako pozadí je použit obrázek s noční oblohou a měsícem, který vyzařuje bílé světlo. Při přepnutí do vedlejšího menu se barva tohoto světla změní. Každé menu má vlastní specifickou barvu.

Menu pozastavení, smrti a výhry



(a) Menu pozastavené hry

(b) Posmrtné menu

(c) Výherní menu

Obrázek 4.10: Při stisknutí tlačítka *Escape* se zobrazí nabídka umožňující ukončení nebo restart hry. Úroveň se pozastaví, tedy veškeré pohyblivé bloky se přestanou hýbat a hráč již nesmí ovládat postavu. Hra zůstane na pozadí viditelná, ale lehce se ztmaví. Pokud hráč vyhraje, nebo prohraje, zobrazí se podobné menu s lehkou obměnou tlačítek. Rozdílem je, že se na pozadí hra nezastaví, ale zůstává v pohybu. Postava v takovém případě přestane přijímat vstup a tím pádem ji nemůže hráč ovládat.

Editor úrovní

Pro vytváření nových úrovní má uživatel k dispozici uživatelské rozhraní. V něm vidí primárně jeho výslednou mapu, kterou může rovnou testovat. Poté má k dispozici dva panely. První je na vrchní části obrazovky. V tomto panelu má k dispozici tlačítko pro otevření nápovědy a pro odchod z editoru. V nápovědě si může připomenout mapování bloků na konkrétní barvy v hlavní vrstvě. Druhý panel je na spodní části a obsahuje tlačítko pro jednoduché přenačtení úrovně. Ve chvíli, kdy hráč udělá změnu v obrázku, který právě testuje, může toto tlačítko stisknout a nové změny se mu načtou.

Při vytváření úrovní může nastat, že udělá uživatel chybu, kterou generátor úrovně nedokáže zpracovat. K tomuto byl implementován systém varovných a blokujících hlášení. **Varovné hlášení** oznamuje chyby, které nepředstavují blokaci při generování úrovně. Jedním takovým hlášením může být upozornění na chybné jméno vrstvy obrázku. V takovém případě je vrstva ignorována a uživatel je o tom informován. Tato hlášení jsou zobrazena ve spodním panelu editoru. Pokud uživatel vytvoří úroveň, kterou není možné vygenerovat, je o tom informován **blokujícím hlášením**. Kdyby se uživatel pokusil vygenerovat úroveň,

kde by například chyběla postava, kterou by mohl hráč ovládat, je přesměrován na černou obrazovku s hlášením upozorňujícím ho na jeho chybu.

Obrázek 4.11: Uživatelské rozhraní pro editor úrovní. Na spodní části obrazovky je vypsané varování pro aktuálně načtený obrázek.

4.8 Generování úrovní

Samotné generování využívá souborů ve formátu TIFF. Jedná se o obrázky s jednou důležitou vlastností, kterou je vrstvení. Díky tomu se v každé vrstvě obrázku může ukládat jiná informace. Jednu vrstvu bude mít uživatel pro definování trajektorie pohyblivých platform, jinou pro vytvoření elektrického systému tlačítek a bloků, které jsou stisknutím ovlivněny. Všechny typy vrstev jsou popsány níže.

Informace o tom, který blok bude na určitém místě, je předána pomocí barevného kódu. Editor využívá barevného modelu RGBA. To znamená, že každá barva je reprezentována pomocí čtyř osmibitových čísel. K dispozici je tedy 32 bitů pro předání informace o bloku. Ona čtyři čísla reprezentují úroveň červené, zelené a modré složky a průhlednost pixelu. Pro všechny vrstvy poté platí, že pokud je pixel průhledný, tedy číslo reprezentující průhlednost pixelu je nulové, je ignorován. Jeden pixel v obrázku reprezentuje jeden blok ve výsledné úrovni.

To, o kterou vrstvu se jedná, pozná generátor podle jejího jména. To musí být napsáno přesně podle níže uvedené konvence. Pokud by se vrstva jmenovala jinak, generátor jí bude ignorovat.

Pojmenování	Vrstva
main	Hlavní vrstva
electricity	Vrstva elektřiny
movingPlatforms	Vrstva pohyblivých platform
keys	Vrstva klíčů a dveří
light	Světelná vrstva

Tabulka 4.1: Přehled vrstev. V levém sloupci jsou uvedena pojmenování, která očekává generátor pro přijetí vrstvy. V pravém sloupci je napsáno, o kterou vrstvu se jedná.

Hlavní vrstva

V hlavní vrstvě si uživatel vytváří primární rozložení scény. Má tu k dispozici valnou většinu všech bloků, které může jakkoliv rozmístit. Výjimkou jsou pohyblivé platformy, tlačítka a klíče. Pro ty má k dispozici další vrstvy, které jsou popsány níže. Každý blok má přiřazenu jednu barvu, kterou může uživatel použít k jeho umístění.

Obrázek 4.12: Snímek obrazovky z editoru úrovní. Zde jsou vypsány veškeré bloky a barvy k nim přiřazené, společně s náhledem bloku.

Vrstva elektřiny

Do vrstvy elektřiny si uživatel nadefinuje tlačítka a bloky, které jsou danými tlačítky ovlivněny. Hodnota červené složky pixelů sjednocuje skupinu bloků napojenou na jeden zdroj elektřiny. Jelikož červená složka je osmibitové číslo, může nabývat 256 různých hodnot. To znamená, že v rámci jedné úrovně může být až 256 různých tlačítek, kde každé bude ovládat vlastní skupinu bloků. Pokud bude ve skupině pixelů se stejnou červenou složkou pouze jeden pixel, stane se z něj osamocené tlačítko, které nebude ovládat žádné bloky. Tímto způsobem lze vytvořit úroveň, kde se bude nacházet velké množství tlačítek a pouze některá z nich budou funkční. Hráčovým úkolem by poté bylo najít ono funkční tlačítko.

To, který blok ze skupiny pixelů bude tlačítko je určeno pomocí modré složky. Ten pixel, který má nejnižší hodnotu modré složky, je vybrán jako tlačítko. Ostatní bloky v dané

skupině jsou tímto tlačítkem ovlivněny a na hodnotě jejich modré složky již nezáleží. V případě, že ve skupině bloků není jasně určeno, který blok má být tlačítkem a nejnižší hodnotu modré složky má více bloků, není přesně definováno, který blok se stane tlačítkem.

Vrstva pohyblivých platform

Tato vrstva umožňuje definovat body, mezi kterými budou cestovat pohyblivé platformy. Pixely se stejnou hodnotou červené složky tvoří jednu trasu. Modrou složkou poté uživatel definuje, v jakém pořadí bude platforma putovat mezi body této trasy. Začátek cesty reprezentuje nejnižší hodnota modré složky. Konec naopak reprezentuje nejvyšší hodnota. Hodnota zelené složky představuje velikost platformy. Pokud je například rovna 3, výsledná platforma se bude rozpínat do šíře tří bloků.

Vrstva klíčů a dveří

Tato vrstva slouží k navržení rozmístění dveří a klíčů, které tyto dveře odemykají. Pro otevření dvěří může být podle návrhu úrovně potřeba více klíčů. Zároveň ovšem může být v úrovni dveří více, které také potřebují své vlastní klíče. Z těchto důvodů je i zde potřeba zavést systém skupin pixelů rozlišující jednotlivé dveře a jejich klíče.

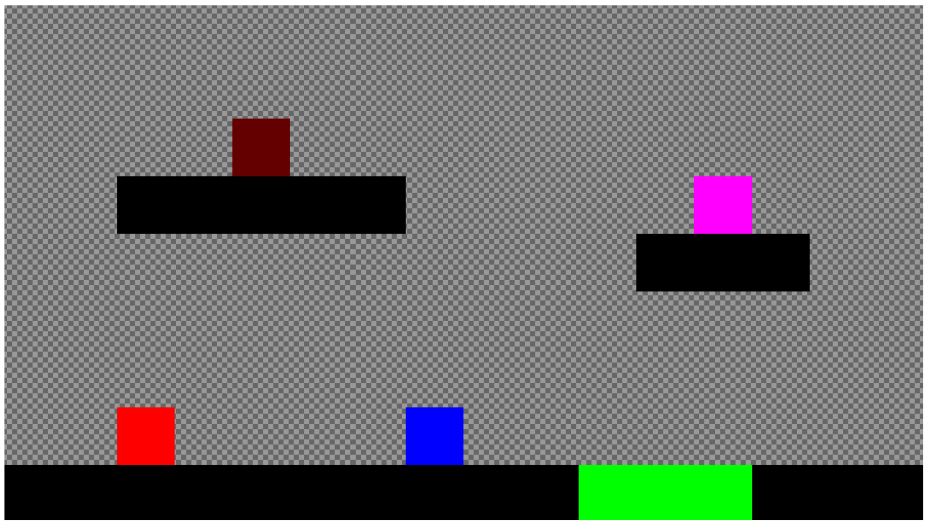
Jedna taková skupina je sjednocena společnou hodnotou alfa kanálu pixelu, tedy jeho průhledností. Všechny pixely reprezentující klíče musí mít černou barvu. Ten pixel, který má jako jediný barvu jinou, reprezentuje dveře.

Jelikož může být ve scéně více klíčů a dveří, jsou odlišeny barvou světla, které vyzařují. To, jakou barvu bude mít daná skupina klíčů, si hráč navolí podle toho, jakou barvu použije na definování dveří. Takže pokud dveře zakreslí například modrou barvou, budou tyto dveře a klíče k nim patřící všechny vyzařovat modré světlo.

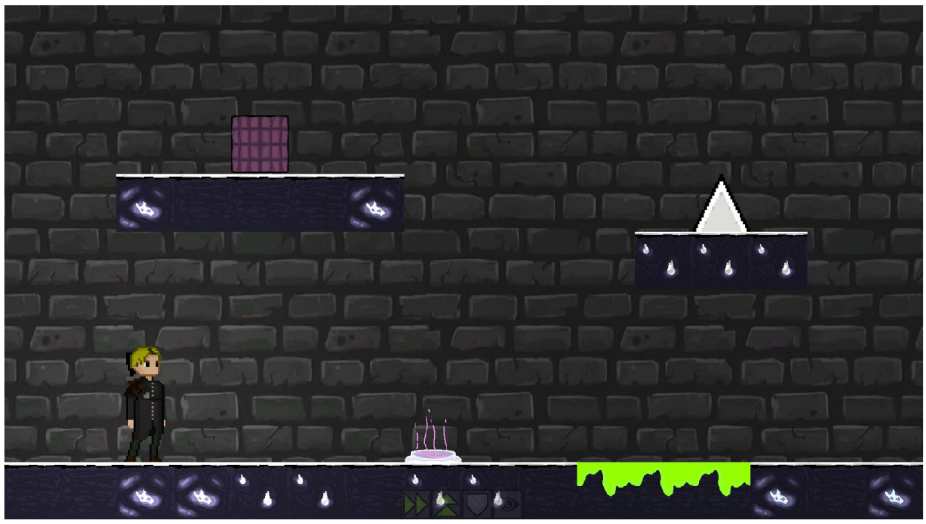
Světelná vrstva

V neposlední řadě si může uživatel nadefinovat vlastní osvětlení scény ve světelné vrstvě. Zde si zakreslí několik bodů. Při generování úrovně se z každého bodu stane zdroj světla. Barva tohoto světla je stejná, jako barva bodu nakresleného uživatelem.

V případě, že tato vrstva není přítomná, použije se základní osvětlení. To znamená, že úroveň je osvětlená globálním zdrojem světla a nikde se nevyskytují stíny. Ovšem pokud přítomná je, globální osvětlení je skoro nulové, tudíž veškeré osvětlení si musí nadefinovat uživatel sám. V tomto případě pak může hráč využívat své schopnosti nočního vidění.



(a) Vstupní obrázek



(b) Vygenerovaná úroveň

Obrázek 4.13: Příklad generování jednoduché úrovně z obrázku ve formátu TIFF.

Kapitola 5

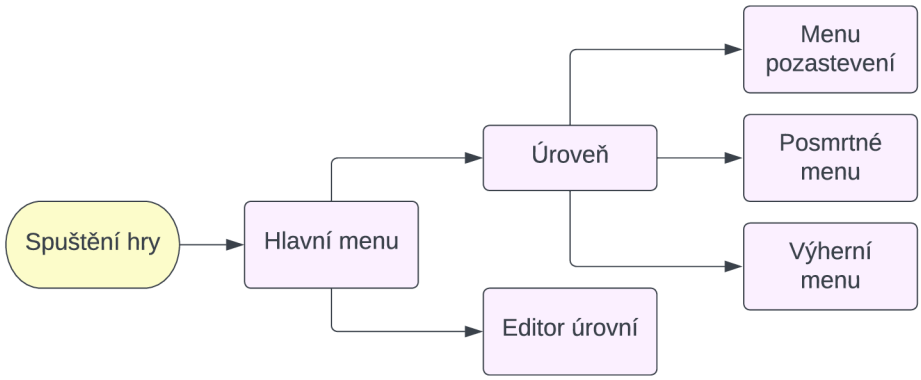
Implementace

Tato kapitola prochází veškeré části hry a popisuje, jak byly implementovány. Hra byla vytvořena za použití herního enginu a vývojového prostředí Unity. Primární část provedení funkcionality hry se odehrává pomocí skriptů v jazyce C#.

Většina textur je originálně vytvořena speciálně pro tuto hru. Vyjímkou jsou textury a animace ústřední postavy[1], pozadí úrovní[2] a animace klíčů[3]. Zároveň s zvuky kroků postavy[4] byly převzaty.

5.1 Uživatelské rozhraní

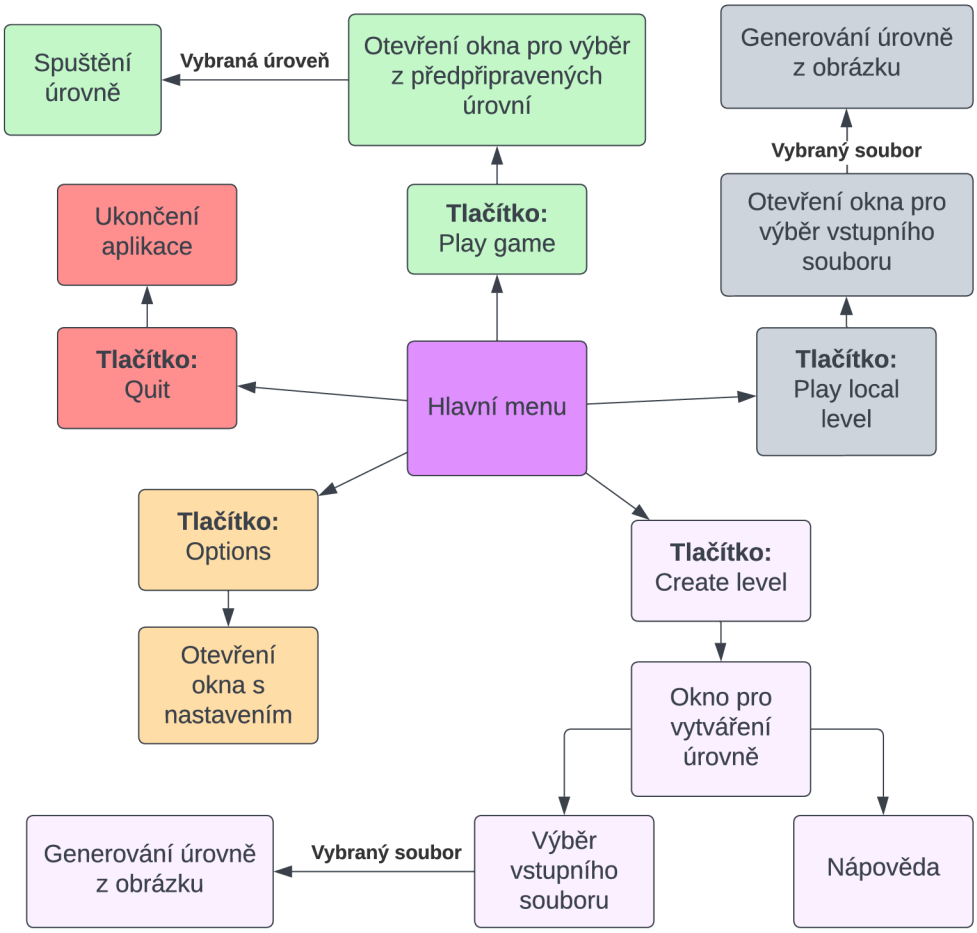
Pro vytváření všech menu byl využit `Canvas`. Jedná se o `GameObject`, který je určený pro uživatelské rozhraní. Vyskytuje se ve vrstvě mezi samotnou hrou a kamerou. Všechna tlačítka, nadpisy apod. zde byly naskládány a `Canvas` byl nastaven tak, aby byl pro všechna rozlišení stejně velký. Vzhled uživatelského rozhraní se nachází v podkapitole 4.7



Obrázek 5.1: Struktura aplikace, na které jsou vyobrazena všechna vytvořená uživatelská rozhraní.

[1] https://assetstore.unity.com/packages/2d/characters/ethan-the-hero-237992
[2] https://assetstore.unity.com/packages/2d/environments/hand-painted-platformer-dungeon-240848
[3] https://drxwat.itch.io/pixel-art-key
[4] https://assetstore.unity.com/packages/audio/sound-fx/foley/footsteps-essentials-189879

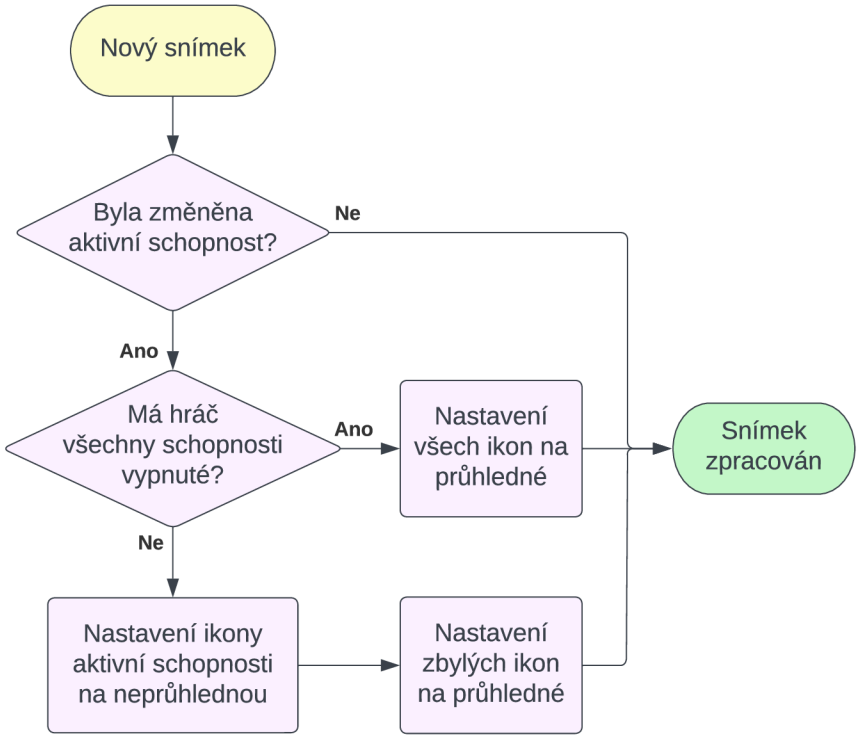
Hlavní menu



Obrázek 5.2: Struktura hlavního menu. Stisknutí tlačítka v hlavním menu buďto přesměruje hráče na další okno, nebo provede akci. Pro výběr vstupního souboru je využit základní prohlížeč souborů pro daný operační systém. Byl využit balíček `StandaloneFileBrowser`[5], který umožňuje otevřít prohlížeč a zpracovat jeho vstup.

[5] https://github.com/gkngkc/UnityStandaloneFileBrowser

Úroveň



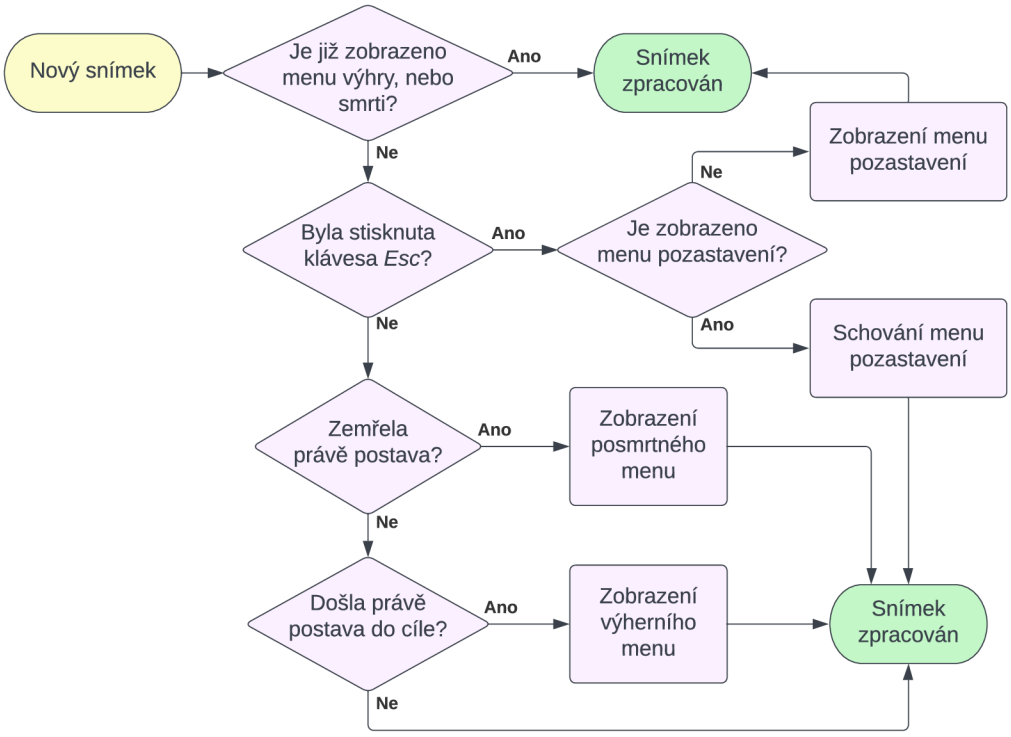
Obrázek 5.3: Ve spodní části obrazovky se u každé úrovně zobrazuje několik ikon. Každá ikona reprezentuje jednu schopnost. Podle toho, kterou schopnost hráč v určitou chvíli využívá, je zvýrazněná příslušná ikona. Pro každý snímek je vyhodnocováno, zda je potřeba změnit zvýraznění ikon. Pokud ano, příslušná ikona je zvýrazněna a zbylé jsou zprůhledněny.

Editor úrovní

Editor úrovní je realizován pomocí dvou panelů, které jsou umístěny na horním a spodním panelu obrazovky. Mezi nimi uživatel vidí vygenerovanou mapu, kterou může v danou chvíli hrát. Jelikož je obrazovka ohraničena těmito panely, musí být upraveny hranice kamery. Podle toho, jestli se jedná o scénu editoru úrovně, se hranice na y-ové ose zmenší. Tím pádem vidí hráč vše, co je potřeba a žádná důležitá část hry není zakrytá.

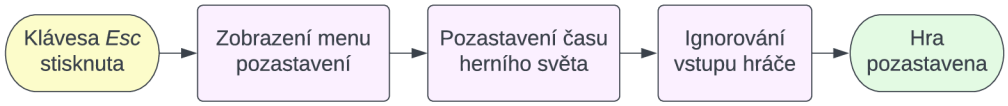
Zde má hráč také k dispozici tlačítko pro otevření návodu, které mu otevře předpřipravené okno, vytvořené ve vývojovém prostředí Unity, se seznamem bloků a jim přiřazenými barvami. Dále může zmáčknout tlačítko „Reload", které přenačte celou scénu. To má za následek, že je momentální úroveň zahozena a generátor načte vstupní soubor znovu. Díky tomu má jeho nejaktuálnější verzi.

Menu pozastavení, smrti a výhry



Obrázek 5.4: Každý snímek je probíhá kontrola, zda by se nemělo některé menu zobrazit. Pokud hráč zemře, nebo vyhraje, nemá již možnost otevřít obyčejné menu pozastavení.

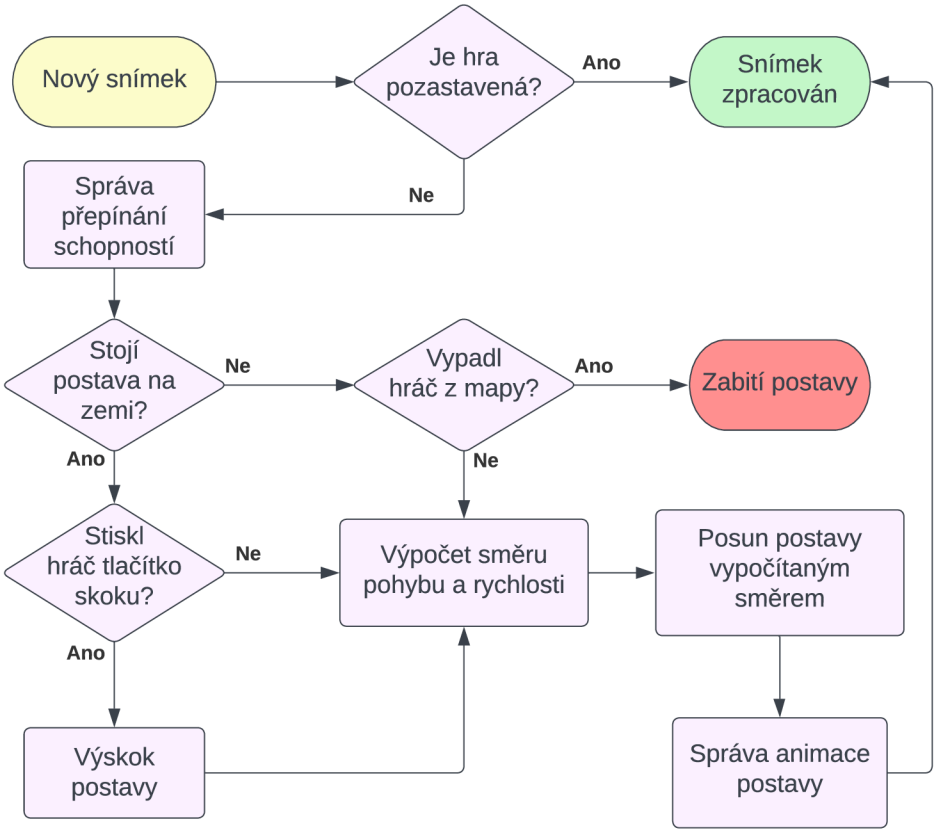
V každém z těchto menu má hráč možnost úroveň restartovat, vrátit se do hlavního menu, nebo hru úplně vypnout. Výherní menu má oproti ostatním ještě tlačítko pro spuštění následující úrovně. Kromě tlačítka pro vypnutí aplikace jsou všechna implementována pomocí přecházení mezi scénami a jejich načítání. Scény jsou předpřipravená okna. Může se jednat například o hlavní menu, nebo specifickou úroveň. Vždy může být aktivní pouze jedna scéna.



Obrázek 5.5: Proces pozastavení hry. Opětovné spuštění všechny tyto kroky vrátí do původního stavu. K tomu může uživatel buďto zmáčknout znovu klávesu *Esc*, nebo stisknout tlačítko „Resume" dostupné v menu pozastavení.

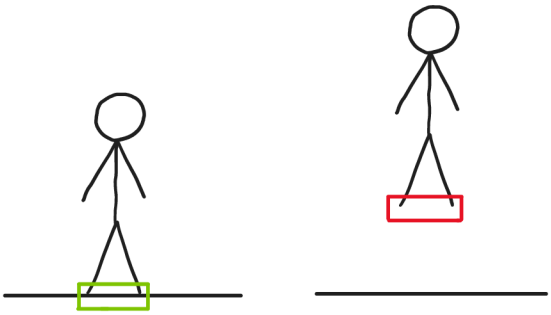
5.2 Postava

Jedná se o `GameObject`, který k sobě má připojených několik různých komponent. Základní komponentou je skript `Player.cs`. Tento skript může přistupovat k ostatním komponentám a měnit jejich hodnoty a nastavení. Spouští zvuky a animace, zajišťuje ovládání pohybu atd. Jeho hlavní funkcionalita je popsána v grafu níže.



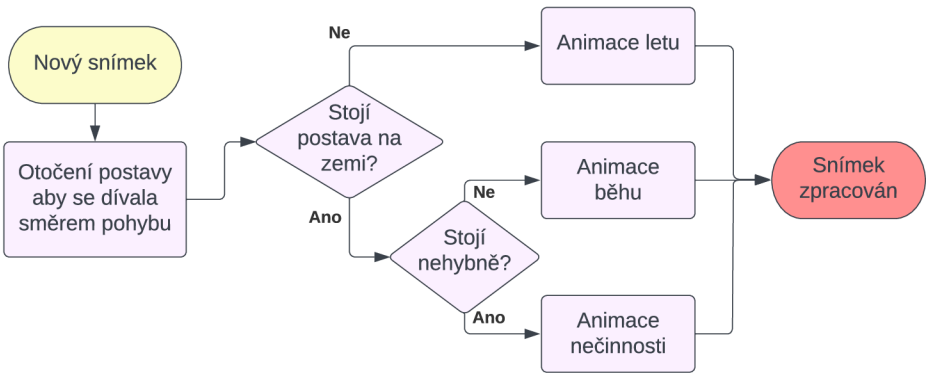
Obrázek 5.6: Pro každý snímek je vyvolán tento algoritmus, který namapuje vstup od hráče na postavu ve hře. Primárně spravuje pohyb postavy ve scéně. Správa přepínání schopnosti je více rozebrána v podkapitole 5.4. Správa animace postavy je znázorněna grafem 5.8 v této kapitole. Pohyb po horizontální ose je zajištěn přičítáním k x-ové složce polohy hráče. Jak velká hodnota je přičtena udává předem vypočtené číslo, které vychází ze všech potřebných faktorů, jako zda má hráč aktivovanou některou ze speciálních schopností, nebo jestli je nějak ovlivněn svým okolím. Výskok postavy je zajištěn jednorázovým přidáním síly na postavu směrem nahoru.

Aby nemohl hráč skákat do nekonečna, je potřeba získat informaci o tom, zda je postava již ve vzduchu, nebo stojí na zemi. V případě, že není uzemněná, nemůže vyskočit. K této kontrole může být použito mnoho různých přístupů. Tato práce využívá `BoxCast`.



Obrázek 5.7: `BoxCast` sleduje určitou oblast a vrací informaci, zda se v ní nachází nějaký blok, případně o jaký se jedná. [9] Pro každý snímek je tato funkce zavolána na oblast, která obaluje nohy postavy (na nákresu vyznačena barevným obdélníkem). Pokud je zvýrazněna zeleně, je v ní detekujován nějaký blok. V případě, že je červená, žádný blok v ní není. Poté je jisté, že je postava ve vzduchu.

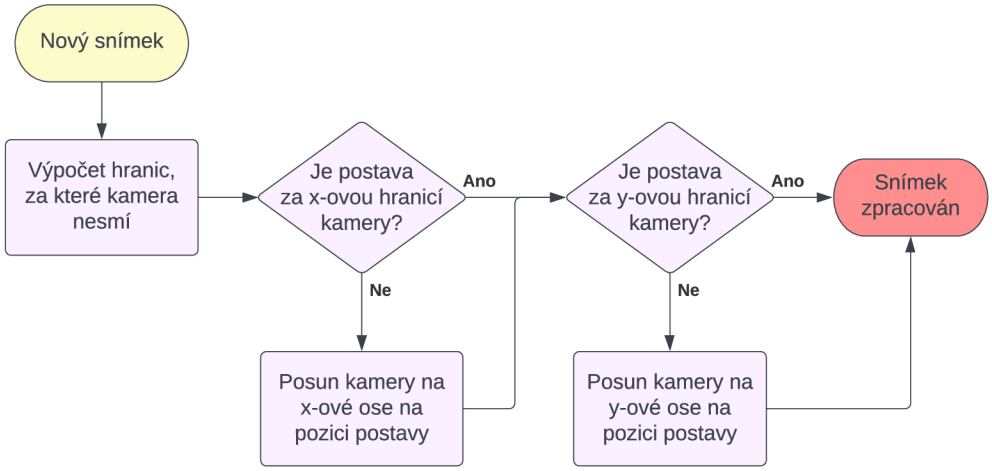
Jak bude daná entita ve scéně vypadat vychází z obrázku, kterému se říká *sprite sheet*. V něm se nachází libovolný počet podobrázků, kterým se říká *sprite*. Ty mohou vypadat jakkoliv a většinou jsou čtvercových rozměrů. Vzhled entit ve scéně pak vychází z těchto spritů. Pro bloky, které nejsou animované, stačí pouze jeden sprite. V tomto případě je ovšem potřeba, aby se postava hýbala a její vzhled reprezentoval zda běží, stojí, či skáče. Každý stav má vlastní animaci.



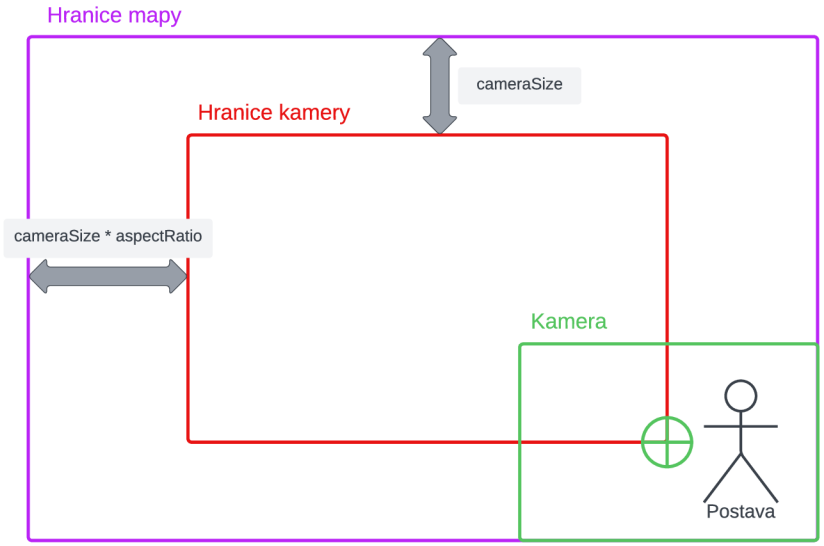
Obrázek 5.8: Správa animace postavy. Tato komponenta využívá předem vypočítané hodnoty pro určení směru pohybu. Pro každý snímek určí, jaká animace má být použita.

Obrázek 5.9: Sprite sheet použitý pro vzhled běžící postavy.

5.3 Kamera



Obrázek 5.10: Proces nastavování kamery opakovaný pro každý snímek. Kamera se snaží držet postavu uprostřed obrazovky, tedy ve svém středu. Avšak v případě, kdy postava dorazí na okraj mapy, by hráč viděl za její hranici, což je nežádoucí chování. Kameře proto nestačí nastavit stejnou polohu, jakou má postava, ale je potřeba jí přepočítat. Ze všeho nejdříve jsou zjištěny její hranice, a poté je zastavena, pokud by se za ně měla dostat.



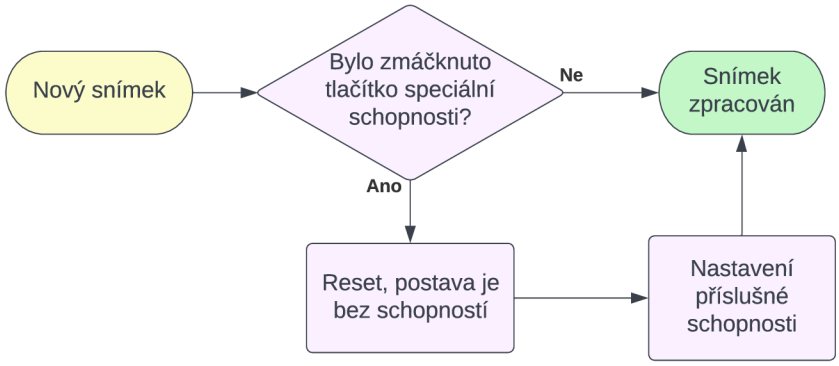
Obrázek 5.11: Zde jsou vyobrazeny hranice a jak spolu souvisí. V tomto případě je postava v pravém spodním rohu mapy. Jelikož je za hranicí kamery, kamera jej nemůže udržet ve svém středu. Je tedy zaseknutá na hranici a nehýbe se dále. Hodnota `cameraSize` značí polovinu vertikální výšky kamery.

Konkrétní vzorce pro jednotlivé strany hranic poté vypadají následovně.

```
minY = cameraSize - 0.5
maxY = mapHeight - cameraSize - 0.5
minX = cameraSize * aspectRatio - 0.5
maxX = mapWidth - cameraSize * aspectRatio - 0.5
```

Konstanta 0.5, která je ve všech vzorcích odečítána, zajišťuje, že i začátek mapy bude správně viditelný. Při generování úrovně se blok, který se nachází ve levém dolním rohu, vygeneruje na pozici [0, 0]. Jelikož má blok v souřadnicovém systému šířku rovnou jedné, zasahuje na obou osách až do vdálenosti 0.5. Kamera tedy musí s touto odchylkou také počítat. Pro výpočet poloviny šířky kamery je nutné vynásobit hodnotu `cameraSize` poměrem stran kamery. U většiny monitorů dnešní doby se jedná o 16:9 – v tomto případě by tedy byl použit jako poměr stran zlomek 16/9.

5.4 Speciální schopnosti



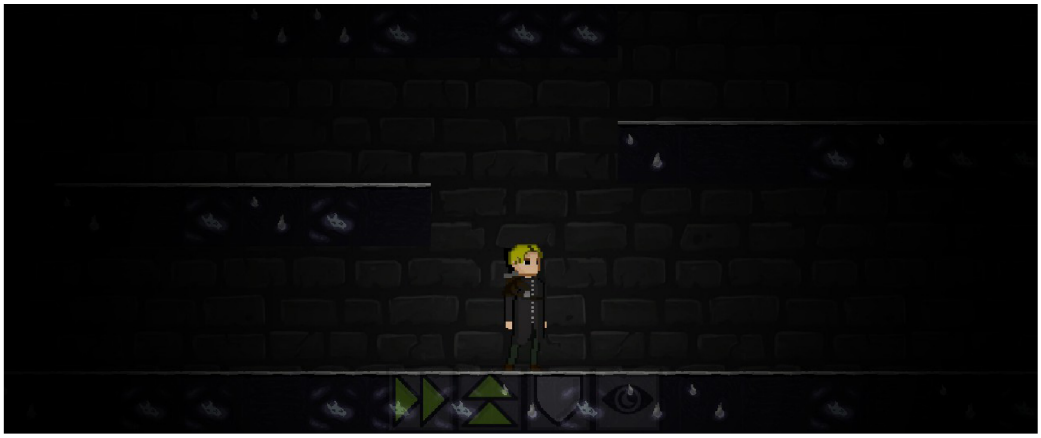
Obrázek 5.12: Algoritmus, který je vyvolán pro každý snímek. Hráč může přepínat mezi speciálními schopnostmi číslicemi od 1 do 5. Každému nastavení schopnosti musí předcházet vypnutí předchozí schopnosti, aby jich nemohl mít hráč aktivovaných několik naráz.

Pro schopnost zvyšující hráčovu rychlost je v skriptu `Player.cs` připravena proměnná `_highSpeedMultiplier`, kterou je následně násobena hráčova rychlost. V základu je tato proměnná rovna jedné, čímž nijak rychlost neovlivní. Pokud je ovšem schopnost zapnuta, nastaví se tato proměnná na hodnotu 2, což znamená, že výsledná rychlost hráče bude dvakrát vyšší. Při přepnutí na jinou schopnost se do této proměnné vrátí hodnota 1.

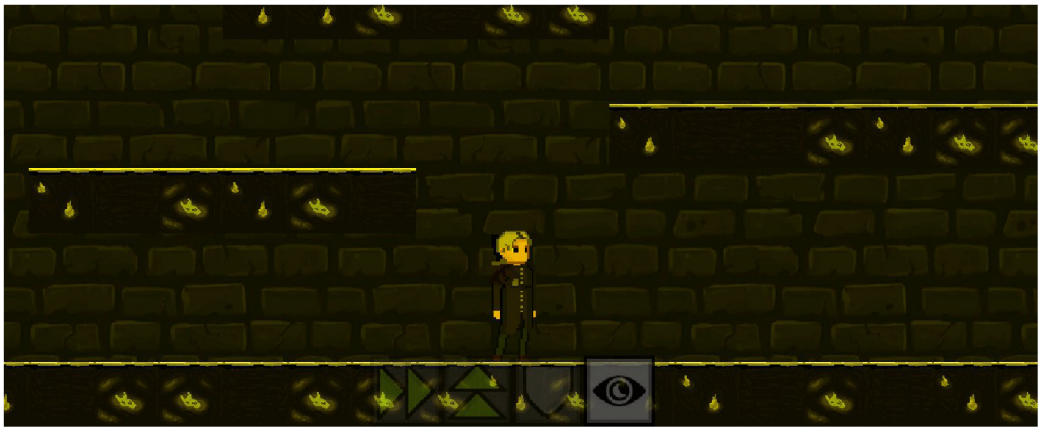
Schopnost vysokého skoku funguje na velmi podobném principu. Hlavním rozdílem je, že pro zvětšení výšky skoku je násobena síla, která je aplikována na postavu při zmáčknutí klávesy skoku.

Další schopností je nesmrtelnost. Když přijde postava do kontaktu s jakýmkoliv objektem, je vyvolána funkce, ve které jsou dostupné informace o daném objektu. Pokud je tímto objektem osten, který může hráče zabít, provede se kontrola, zda je hráč nesmrtelný. Pokud ano, je osten zničen, jinak je spuštěna funkce, která hráče zabije.

Schopnost nočního vidění je umožněna pomocí `Light2D`. Jedná se o komponentu napojenou na postavu, ovšem může existovat i sama o sobě. Její funkcionalita spočívá ve vyzařování světla a osvětlování scény v okolí. V tomto případě je využito bodové světlo, tedy takové osvětlující pouze kruh okolo sebe. Má definovanou svojí barvu, dosah, vzdálenost, po které začne osvětlení slábnout atp. V základu má hráč okolo sebe poměrně slabé bílé světlo. Jakmile zapne noční vidění, toto světlo značně zesílí a rozšíří svůj dosah. Navíc změní svojí barvu na odstín žluté, aby bylo ihned jasně viditelné, že je ona schopnost zapnuta.



(a) Světelný kruh bez zapnuté schopnosti



(b) Světelný kruh se zapnutou schopností

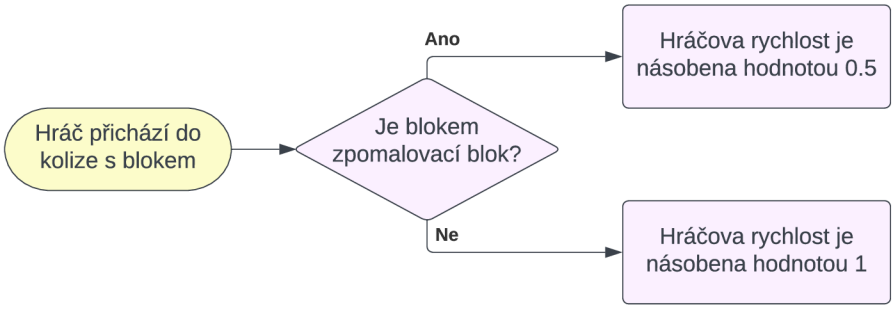
Obrázek 5.13: Při zapnuté speciální schopnosti je světelný kruh okolo hráče značně zesílen. Pokud hráč nemá kameru velmi oddálenou, samotné hranice tohoto kruhu ani nevidí.

5.5 Speciální bloky

Ve vývojovém prostředí byly připraveny jednotlivé bloky a uloženy jako prefaby. Byly jim přiděleny příslušné komponenty a skripty ovládající jejich funkcionalitu. Při generování úrovní jsou tyto prefaby načítány a vytvářeny jejich instance ve scéně. O tomto více v kapitole 5.6.

Tato podkapitola se zaměřuje detailně na to, jak byla zrealizována implementace všech speciálních bloků.

Zpomalovací blok

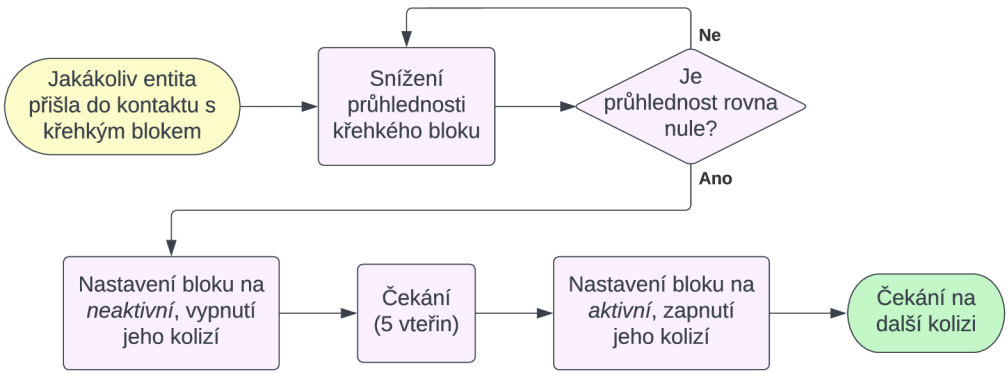


Obrázek 5.14: Ovládání zpomalovacího bloku se děje plně v `Player.cs` skriptu. Hráčova rychlost je vždy násobena proměnnou, která je nastavována v závislosti na kontaktu se zpomalovacím blokem. Pokud se ho dotkne, je nastavena na hodnotu 0.5. Jakmile přijde do kontaktu s jakýmkoliv jiným blokem, nastaví se na hodnotu 1.

Pohyblivý pás

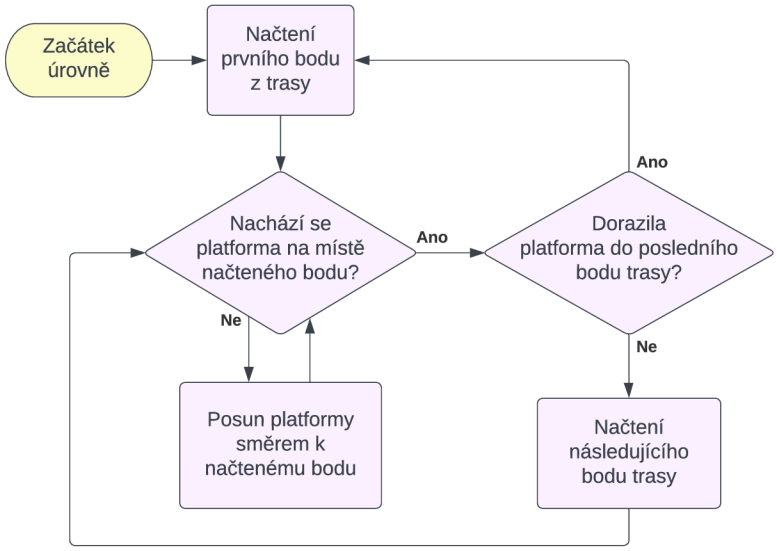
Funkcionalita pohyblivého pásu je realizována pomocí komponenty `SurfaceEffector2D`. Ta aplikuje určitou sílu předem vybraným směrem na všechny objekty, které s ním přijdou do kontaktu. V tomto případě se může jednat o hráče, nebo krabici.

Křehký blok



Obrázek 5.15: Entita křehkého bloku neustále čeká, než s ní nějaký objekt přijde do kontaktu. Jakmile se tak stane, spustí se tento algoritmus. Blok ze všeho nejdřív postupně zmizí. Následně se objeví a začíná znovu čekání na kolizi s jakýmkoliv objektem.

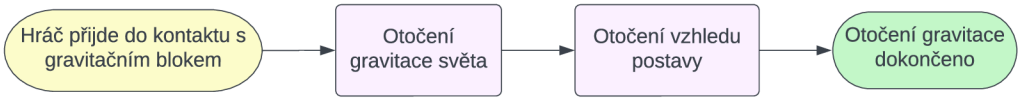
Pohyblivé platformy



Obrázek 5.16: Pohyblivá platforma putuje mezi body její trasy, které má uložené a seřazené v poli souřadnic. Načítání bodů do trasy je popsáno v podkapitole 5.6. Tento algoritmus je nekonečný a přerušen je pouze ukončením úrovně. Platforma se postupně pohybuje směrem k dalšímu bodu její trasy. Jakmile do něj dorazí, začne putovat do dalšího bodu. Pokud se nachází již v posledním bodě trasy, začne putovat zpět na začátek a začíná svojí cestu znovu.

Jakmile naskočí postava na platformu, musí se pohybovat společně s ní. Proto ve chvíli, kdy přijdou tyto dvě entity do kolize, je v hierarchii `GameObjectů` platforma nastavena jako rodič postavy. To způsobí, že změny v pozici platformy jsou aplikovány zároveň na postavu. Ve chvíli, kdy z platformy seskočí, je tato změna navrácena a postava je ve scéně opět bez rodičovského objektu. Toto samé funguje i pro krabice.

Gravitační blok



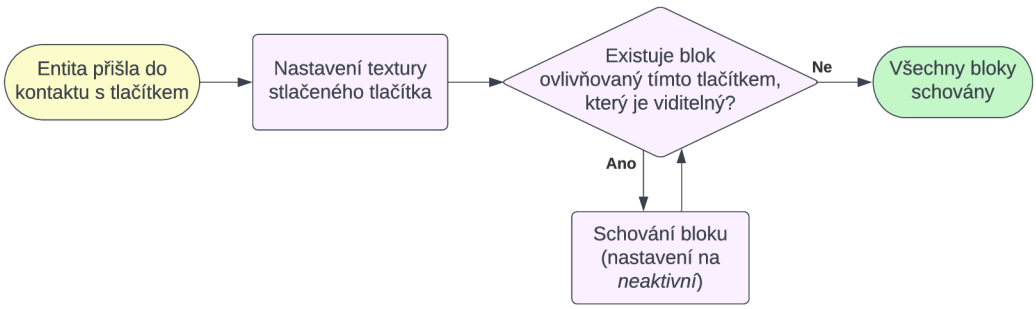
Obrázek 5.17: Algoritmus, který je spuštěn kdykoliv hráč přijde do kontaktu s gravitačním blokem. Tento blok může být položen na zemi, i na stropě mapy. Samotné otočení gravitace je realizováno pomocí `Physics2D`, které pro ní umožňuje nastavit vlastní hodnoty. Pro normální gravitaci je nastavena na y-ové ose hodnota 9.8. V opačném případě je tato hodnota záporná.

Krabice

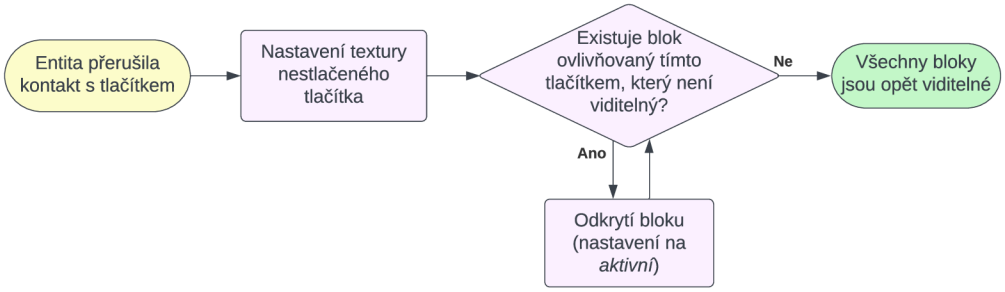
Krabice je `GameObject`, který má narozdíl od statických objektů přiřazenou komponentu `Rigidbody2D`, což znamená, že na něj budou aplikovány fyzikální zákony. Jinými slovy na něj začne působit gravitace a je přitahován k současnému bodu přitažlivosti.

Tlačítka

Tlačítko využívá vlastního skriptu, který je k němu připojen jako komponenta. Tento skript ovládá to, jaké akce se vykonají při stlačení a uvolnění tlačítka. Má neustále k dispozici pole bloků, které ovládá. Toto pole je naplněno při generování úrovně (více v podkapitole 5.6).



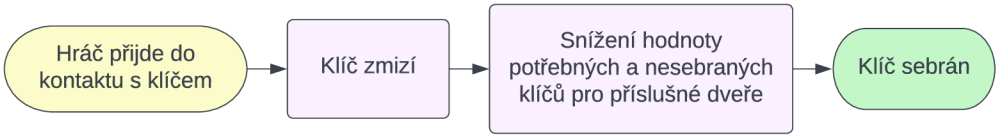
Obrázek 5.18: Při zmáčknutí tlačítka je proiterováno pole ovlivňovaných bloků a všechny jsou postupně schovány.



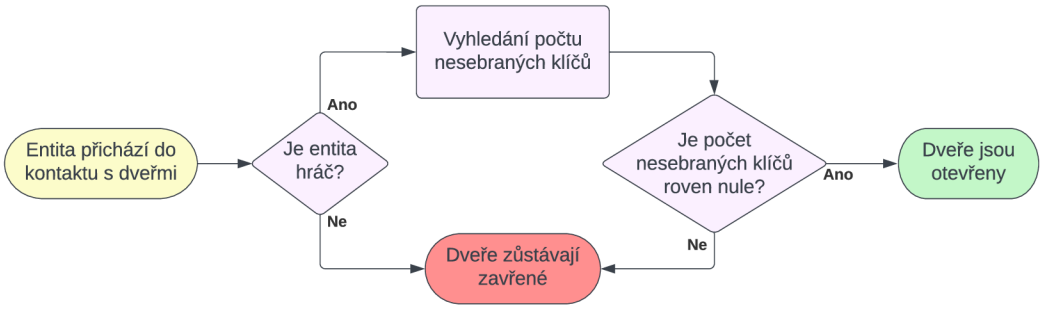
Obrázek 5.19: Algoritmus, který vrací změny provedené stlačením tlačíka. Funkcionalita je velmi podobná jako při stisknutí tlačítka 5.18, akorát místo schování bloků jsou odkryty.

Klíče a dveře

Entity klíčů a dveří využívají seznam dveří, ke kterým je dodána informace o tom, kolik klíčů je ještě potřeba sebrat, aby byly otevřeny. Tento seznam je statický, takže k němu je možné přistoupit z více entit a upravit jej.

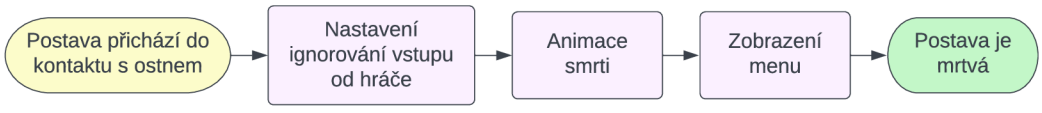


Obrázek 5.20: Průběh sebrání klíče. V seznamu dveří je snížen počet klíčů, který je potřeba pro jejich otevření, o jedna.



Obrázek 5.21: Pro otevření dveří musí být počet potřebných klíčů roven nule, tzn. všechny musí být sebrány. Tato informace pochází z výše zmíněného seznamu.

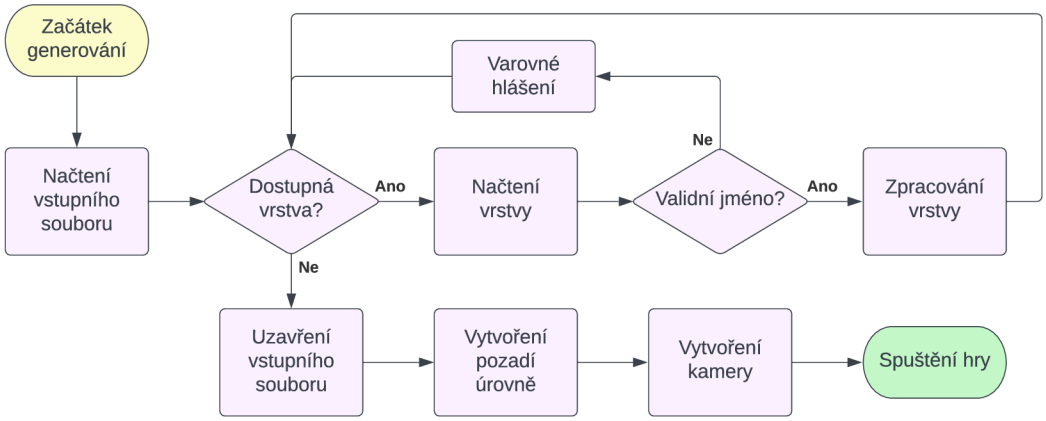
Ostny



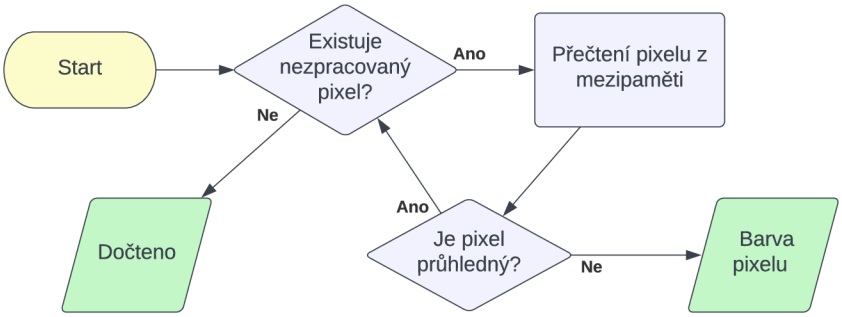
Obrázek 5.22: Posloupnost událostí, které se spustí při kontaktu postavy s ostnem. Poté, co hráč zemře, je mu znemožněno již jakkoliv ovládat postavu.

5.6 Generování úrovně

Hráč má možnost vytvářet vlastní úrovně nakreslením obrázku ve formátu TIFF. Pro jejich čtení byla využita knihovna `LibTiff` od `BitMiracle`[6]. Generátor úrovně je `GameObject`, který je umístěn samotný ve scéně. Při spuštění dané scény započne svojí práci níže popsaný algoritmus.



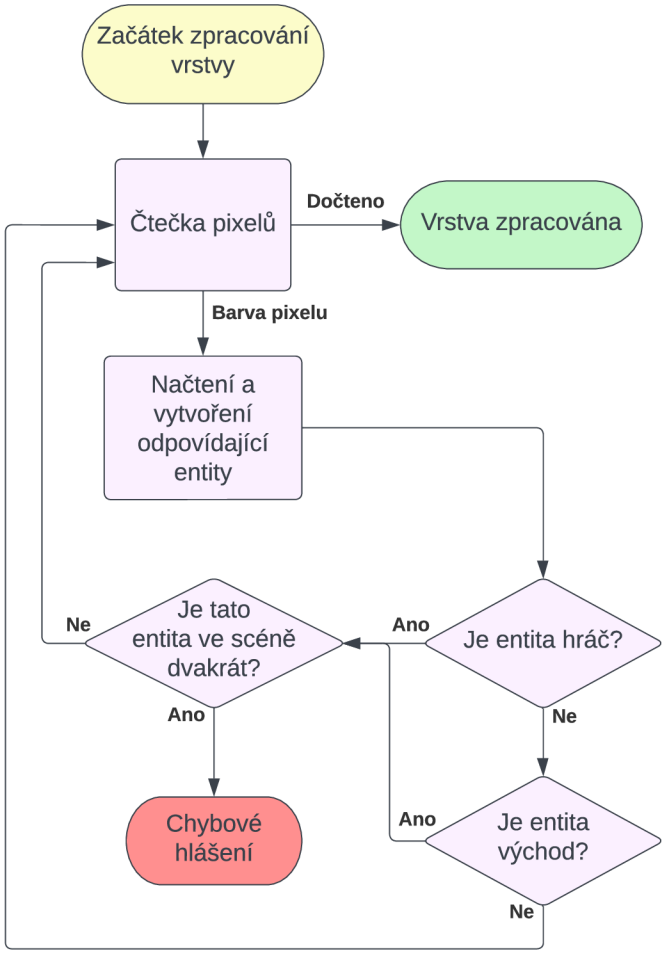
Obrázek 5.23: Vývojový diagram znázorňující průběh načítání souboru a následné generování úrovně. Algoritmus přečte postupně všechny vrstvy. Ze všeho nejdříve zjistí z jejich tagu jméno vrstvy. Pokud je validní, je zavolána příslušná funkce. Po přečtení poslední vrstvy uvolní načtený soubor, vytvoří pozadí a kameru a hráč začíná hrát. Postup generování jednotlivých vrstev je popsán v grafech níže.



Obrázek 5.24: Vývojový diagram čtečky pixelů. Tato komponenta je využita ve všech algoritmech pro zpracování vrstvy. Zeleně vyznačené části reprezentují výstup komponenty, který bude v dalších diagramech použit.

[6] https://bitmiracle.com/libtiff/

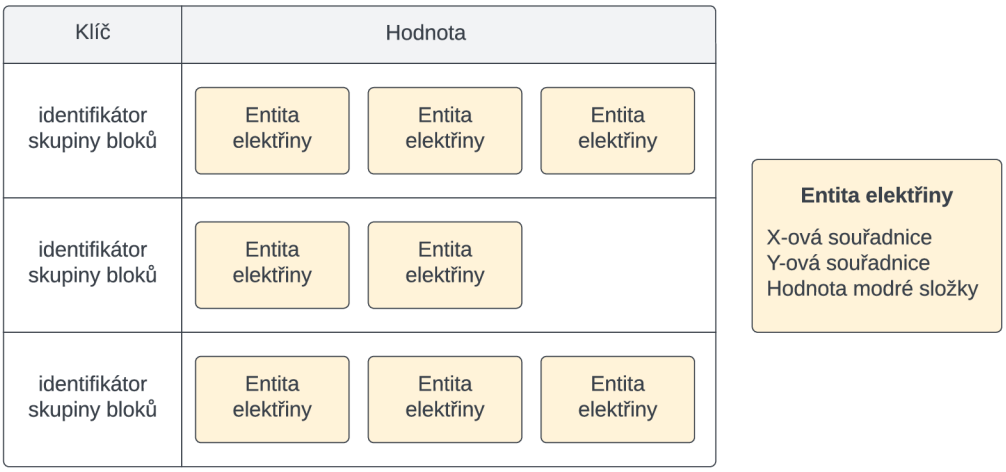
Hlavní vrstva



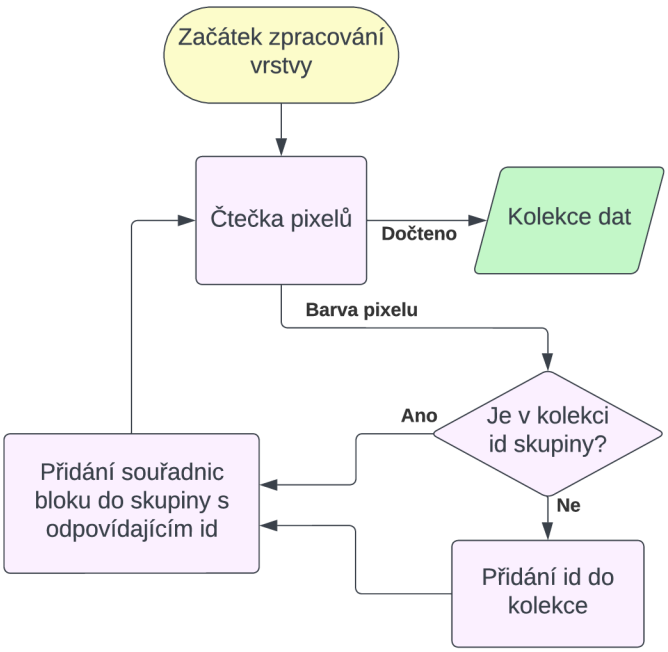
Obrázek 5.25: Postup procházení hlavní vrstvy a zpracování jednotlivých pixelů. Zde se generuje většina bloků. Jedná se o ty, které mohou existovat jako samostatné jednotky a nejsou funkcionalitou navázané na jiné bloky, například obyčejný blok země, zpomalovací blok, nebo pohyblivý pás. Mapování bloků na jim přiřazené barvy je napevno definováno v kódu ve formě pole objektů obsahující barvu a cestu k prefabu daného bloku. Ve chvíli, kdy má být některá z entit vložena do scény, generátor využije cestu k prefabu z pole mapování. Tato cesta je relativní ke složce *Resources*, ze které lze pak snadno prefab načíst a vytvořit jeho instanci ve scéně na požadovaných souřadnicích.

Vrstva elektřiny

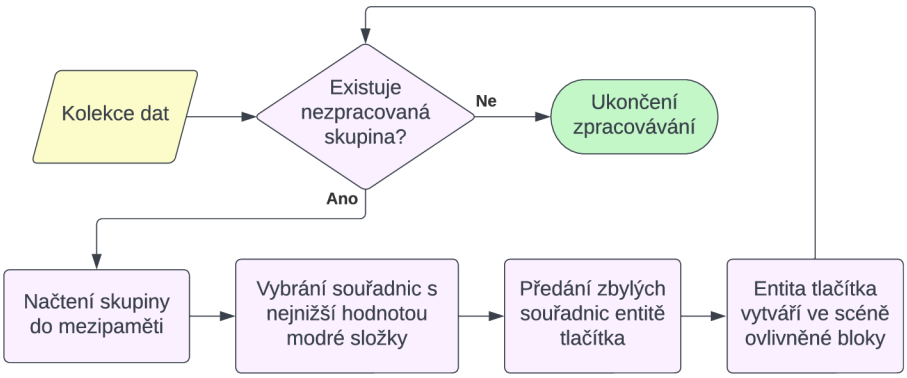
Obrázek 5.26: Zpracování vrstvy elektřiny se odehrává ve dvou na sebe navazujících krocích. Data jsou mezi fázemi přeposílány pomocí objektů třídy `Dictionary` (dále kolekce) dostupné v C#. Tato třída umožňuje vytvořit pole indexované vlastním klíčem.



Obrázek 5.27: Znázornění kolekce dat předáváné mezi fázemi zpracování vrstvy. V kolekci může být mnoho záznamů, na tomto příkladu jsou pouze tři. Indexovány jsou pomocí klíče, kterým je identifikátor skupiny, nebo-li červená složka barvy pixelu. Každý záznam drží informace pro danou entitu vypsané v pravé části obrázku. V této fázi ještě není rozhodnuto, která z entit bude tlačítkem. Tím bude ten s nejnižší hodnotou modré složky.

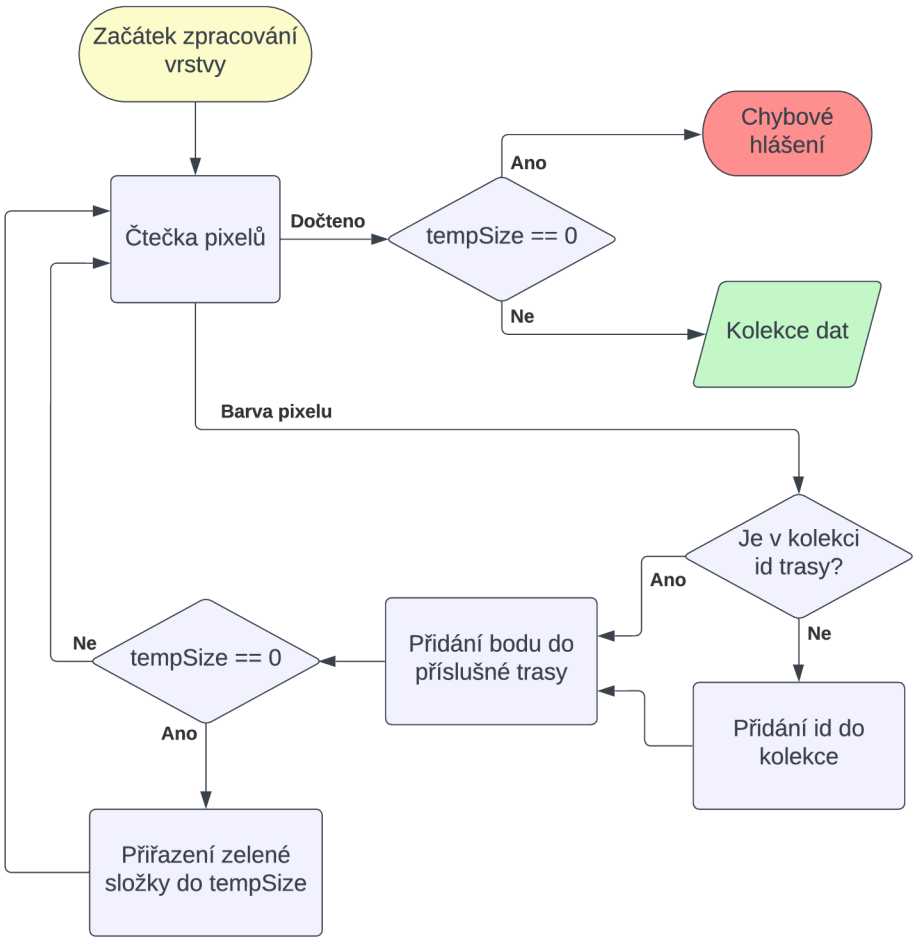


Obrázek 5.28: První fáze zpracování vrstvy elektřiny. Zde jsou načteny souřadnice všech pixelů a roztříděny podle toho, které skupiny elektřiny se týkají. Jednu skupinu tvoří tlačítko a bloky na něj navázané. Ve vstupním obrázku jsou propojeny stejnou hodnotou červené složky. Výstup, zvýrazněný zelenou barvou, je kolekce dat podle struktury vyobrazené na 5.27.



Obrázek 5.29: Druhá fáze zpracování vrstvy elektřiny. Na vstupu, vyznačeném žlutou barvou, jsou všechna načtená data z první fáze. Zde jsou tato data zpracována. Algoritmus vytvoří ve scéně všechna tlačítka a předá jim seznam souřadnic bloků, které budou ovlivňovat. Každé z tlačítek je zodpovědné za vytvoření těchto bloků ve scéně. Takto je bude mít pod kontrolou a tím pádem je má možnost snadno ovládat.

Vrstva pohyblivých platform



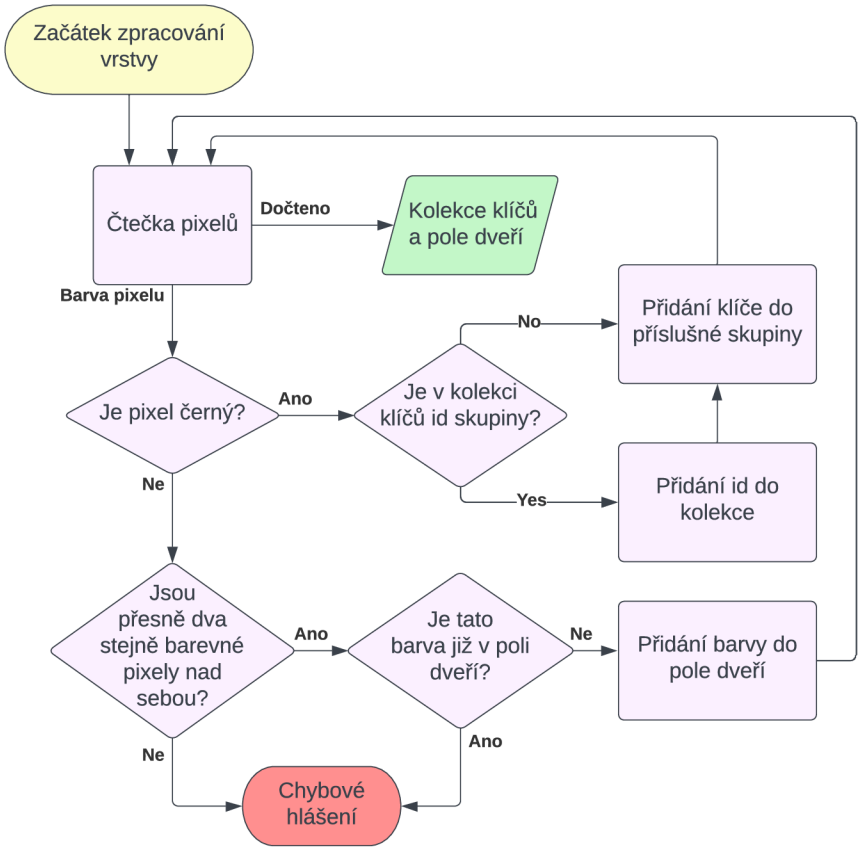
Obrázek 5.30: Generování pohyblivých platform se liší od generování elektřiny v několika detailech. Alespoň jeden z pixelů trasy pohyblivé platformy musí mít nastavenou zelenou složku na nenulovou hodnotu. První, který tuto podmínku splňuje, je využit pro nastavení velikosti platformy. Pokud by měl nějaký další pixel nastavený jinou hodnotu v zelené barvě, je ignorována. V případě, že všechny zelené složky jsou nulové, je zobrazeno chybové hlášení.

Následné zpracování kolekce funguje na stejném principu, jako v případě vrstvy elektřiny (viz. 5.29). Pomocí cyklu je přečtena celá kolekce a pro každou skupinu je vybrán pixel s nejnižší hodnotou modré složky. Tento pixel reprezentuje začátek trasy pohyblivé platformy. Zde je vytvořen blok, kterému je předána velikost výsledné platformy a body jeho trasy.

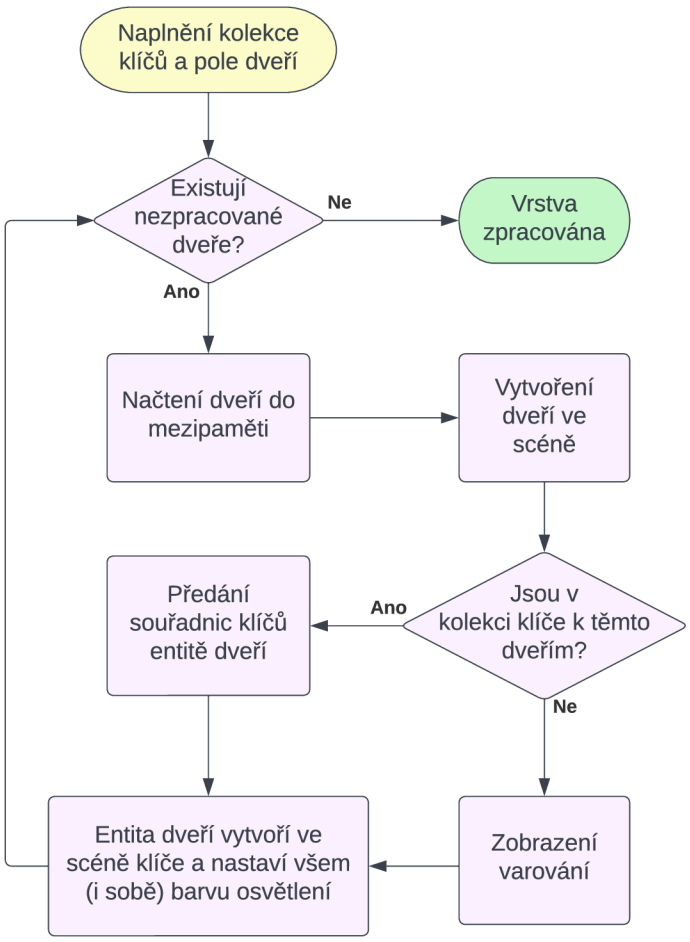
Nově vytvořený blok vytvoří další bloky tak, aby jejich výsledný počet odpovídal velikosti platformy. Každý blok je vytvořen na místě posunutém o jednu hodnotu x-ové souřadnice doprava a všechny body jeho trasy jsou posunuté stejným způsobem. Výsledkem je několik bloků cestujících neustále vedle sebe po předem definované trase.

Vrstva klíčů a dveří

Obrázek 5.31: Vytváření klíčů a dveří funguje ve dvou větších fázích. Narozdíl od vrstvy eletřiny a pohyblivých platform je mezi fázemi předáváno více dat, a to pole všech dvěří a kolekce klíčů patřících k nim.



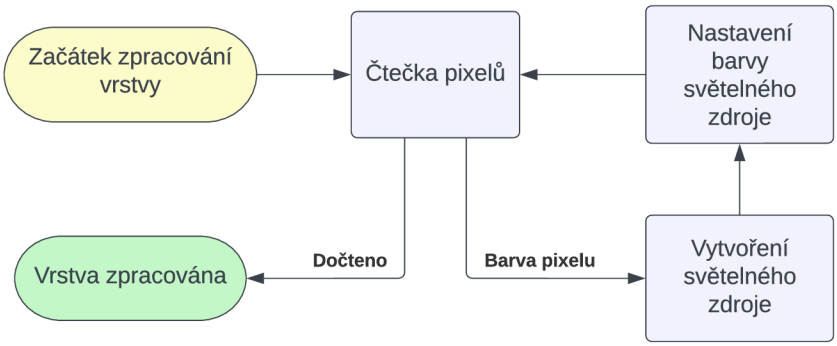
Obrázek 5.32: V této fázi je z obrázku naplněno pole všech dveří a kolekce klíčů. Klíče jsou do prvků kolekce rozděleny podle toho, ke kterým dveřím patří. Jeden prvek kolekce je pole souřadnic všech klíčů, patřícím k jedné skupině. V kolekci jsou indexovány pomocí identifikátoru skupiny klíčů a dveří. Tím je složka průhlednosti pixelu. Prvek pole dveří obsahuje jejich souřadnice a barvu světla, kterou bude ve scéně skupina vyzařovat.



Obrázek 5.33: Po dokončení první fáze je spuštěn tento algoritmus. Pole dveří je proiterováno a pro každou položku jsou ve scéně vytvořeny dveře. Každé entitě je předáno pole souřadnic klíčů, které je budou odemykat. Vytvořené entity dveří se poté starají o vytvoření všech jejich klíčů ve scéně a obarvení světel celé skupiny.

Světelná vrstva

Tato práce využívá URP (Universal Render Pipeline), kterou poskytuje Unity. Pro fungování osvětlení scény je nezbytná. Až poté, co je v projektu použita, lze přidávat entity typu `Light2D`. [9] V této vrstvě je nadefinováno odkud budou svítit bodová světla.



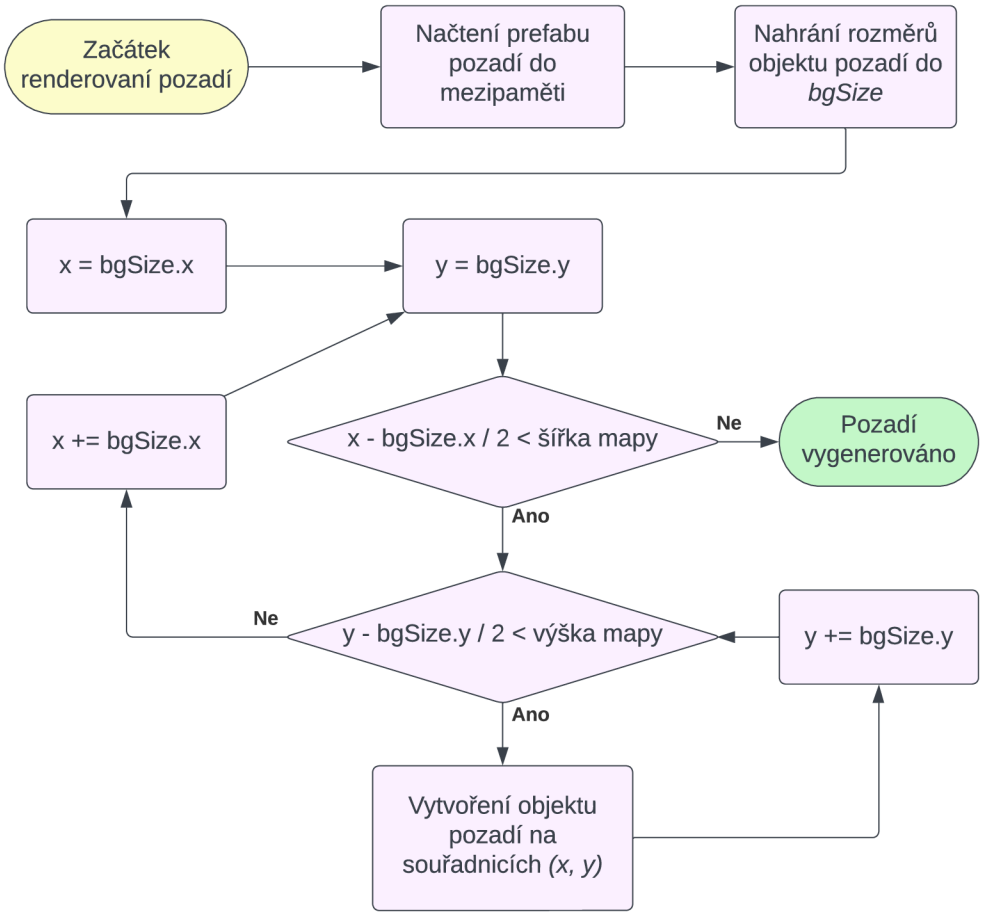
Obrázek 5.34: Algoritmus pro zpracování světelné vrstvy souboru. Pro každý pixel je vytvořen zdroj světla, který je následně obarven příslušnou barvou.

Po zpracování všech vrstev je přidáno do scény globální osvětlení s velmi nízkou intenzitou. Pokud není světelná vrstva ve zdrojovém obrázku přítomná, nastaví se globálnímu světlu intenzita značně vyšší, aby mohla být úroveň stále hratelná.

Vytvoření kamery

Pro kameru je připravený prefab, který na sobě má připojený skript. Tento skript ovládá velikost kamery a zároveň její umístění. Generátor úrovně jí pouze umístí na střed mapy a nastaví její velikost tak, aby se přesně vešla do mapy a nepřesahovala její hranice. Způsob, jakým si kamera následně nastavuje svoje umístění a velikost je popsáno podrobně v podkapitole 5.3.

Pozadí úrovně



Obrázek 5.35: Algoritmus pro vytvoření pozadí. Práce využívá texturu pozadí, která může být duplikována a navazována sama na sebe díky neviditelným přechodům. Toto je žádoucí, jelikož rozměry mapy nejsou předem definované a nemůžeme použít stejně velkou texturu pro různě velké mapy. Musela by totiž být deformována, aby pokryla celé pozadí. Tento algoritmus vytvoří tolik entit textur, kolik je potřeba, aby dohromady vytvořily celistvé pozadí přes celou mapu. Pokud je umístěna textura pozadí na souřadnicích [x, y], bude se rozpínat do vzdálenosti poloviny její velikosti. Proto se nachází v algoritmu výpočet, který od souřadnic [x, y] odečítá polovinu velikosti textury pozadí. Ten má za účel zjistit, jestli nějaká část potenciálně vytvořené textury bude na mapě viditelná. Pokud ne, není ji potřeba vytvářet.

Kapitola 6

Závěr

Cílem této práce bylo navrhnout a implementovat plošinovou hru ve 2D s jednoduchým způsobem vytváření vlastních úrovní. Tyto cíle práce splňuje. Hra byla vytvořena s několika úrovněmi, které byly postaveny za použití vlastního editoru úrovní. Tento editor je přístupný také běžným uživatelům. Pro vytvoření nové úrovně je potřeba, podle předepsaných pravidel, nakreslit obrázek ve formátu TIFF, který je následně hrou načten a překonvertován na hratelnou úroveň. Hra byla implementována za použití herního enginu Unity. Hlavní funkcionalita, včetně generování vstupních úrovní, byla vytvořena v programovacím jazyce C#. Zároveň byla vytvořena a zveřejněna upoutávka[1], kde byla použita volně dostupná hudba od ComaStudio[2].

Ze všeho nejdříve byly prozkoumány již existující hry, které následně sloužily jako inspirace pro vytvoření této práce. V dalším kroce byly navrženy všechny využité herní mechaniky, např. jakým způsobem hráč ovládá postavu, jak vypadá herní prostředí a jak se chová kamera zabírající dění ve hře. Zároveň byly připraveny všechny speciální bloky, které se mohou vyskytovat v herním prostředí a ovlivňovat hráče, a speciální schopnosti ústřední postavy. Také byla popsána korelace mezi prostředím hry a schopnostmi hráče, tedy jak jedno ovlivňuje druhé. Následovalo navržení funkcionality editoru úrovní a způsobu, kterým bude ze vstupního obrázku generována úroveň. Vstupní obrázky mají několik vrstev a každá obsahuje jiná data, kupříkladu: v jedné vrstvě je definováno osvětlení scény, zatímco v jiné je navržen systém klíčů a dveří. Všechny tyto návrhy byly nakonec implementovány a převedeny do samotné hry.

Po dokončení byla hra otestována malým vzorkem lidí. Ohlasy byly vesměs kladné a uživatelé hlásili, že se při hraní dobře bavili. Předem připravené mapy zvládli dohrát přibližně do hodiny, a proto ocenili možnost vytváření vlastních nových úrovní. Ti, kteří se o vytvoření plnohodnotné úrovně pokusili, měli ze začátku lehké problémy s pochopením všech pravidel pro kreslení vstupního obrázku. Po chvíli každopádně veškeré koncepty pochopili a vytvořili poměrně zajímavé úrovně, které si mezi sebou sdíleli. Uživatelé hodnotili vlastní úrovně jako zábavný způsob vyzývání ostatních hráčů a byli překvapeni, že se s tímto typem editoru do této chvíle nesetkali.

Ačkoliv byl cíl práce splněn a hra byla implementována podle plánu, existuje, díky její povaze, stále mnoho aspektů, které by bylo možné do hry přidat. Mohou to být další speciální bloky a schopnosti, nové oficiální úrovně nebo i optimalizace běhu aplikace.

[1] https://youtu.be/_IxSA_PCDE0
[2] https://pixabay.com/music/beats-stylish-rock-beat-trailer-116346/

Literatura

[1] Adams, E. a Dormans, J. *Game Mechanics: Advanced Game Design*. 1. vyd. New Riders, 2012. ISBN 978-0321820273.

[2] Bramble, R. *The Seven Stages of Game Development* [online]. 2023 [cit. 2023-04-19]. Dostupné z: https://gamemaker.io/en/blog/stages-of-game-development.

[3] Cambridge. *Cambridge Dictionary* [online]. [cit. 2022-12-29]. Dostupné z: https://dictionary.cambridge.org/dictionary/english/video-game.

[4] Gregory, J. *Game Engine Architecture*. 3. vyd. A K Peters/CRC Press, 2018. ISBN 978-1138035454.

[5] Koster, R. *A Theory of Fun for Game Design*. 2. vyd. O'Reilly Media, 2013. ISBN 978-1449363215.

[6] Murray, J. D. a VanRyper, W. *Encyclopedia of Graphics File Formats: The Complete Reference*. 2. vyd. O'Reilly Media, 1996. ISBN 978-1565921610.

[7] Schreier, J. *Blood, Sweat, and Pixels: The Triumphant, Turbulent Stories Behind How Video Games Are Made*. 1. vyd. Harper Paperbacks, 2017. ISBN 9780062651242.

[8] Sinicki, A. *What is Unity? Everything you need to know* [online]. 2021 [cit. 2022-12-30]. Dostupné z: https://www.androidauthority.com/what-is-unity-1131558/.

[9] Technologies, U. *Unity User Manual 2021.3 (LTS)* [online]. 2023 [cit. 2023-04-21]. Dostupné z: https://docs.unity3d.com/Manual/index.html.