STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Vývoj videohry v enginu Godot

**Monika Rozínková**

**Středočeský kraj Mělník 2024**

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Vývoj videohry v enginu Godot

Game development in Godot Engine

**Autor: Monika Rozínková**

**Škola: Gymnázium Jana Palacha, Mělník, Pod Vrchem 3421**

**Kraj: Středočeský kraj**

**Konzultant: Mgr. Markéta Wolfová**

**Mělník 2024**

Prohlášení:

Prohlašuji, že jsem svou práci SOČ vypracovala samostatně a použila jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

Mělník dne 27.11. 2024 Monika Rozínková

Poděkování:

Mé díky patří zejména vedoucí mé práce Mgr. Markétě Wolfové. Dále chci poděkovat Kateřině Michálkové za pomoc s korekturou a podobou finálního znění, Timofeji Kožuchovi, BSc. za cenné rady a psychickou podporu při procesu praktické i teoretické části a nakonec Mgr. Lukáši Procházkovi za pomoc s testováním a hledáním chyb ve hře.

Anotace

Informace zahrnuté v této práci popisují vývoj videohry v enginu Godot za použití jazyka GDScript. Rozdělena je do dvou hlavních částí. Umělecká, nastiňující kreslení a designování herního světa a karet, a technická zabývající se mechanikami a kódem hry.

Klíčová slova

Videoherní vývoj; karetní hra; Godot; programovací jazyky

Annotation

Infomation that is held in this work describes development of a game in Godot engine with usage of language GDScript. It is divided into two main parts. Artistic one, which describes designing and drawing the game world, and cards and technical one which deals with game mechanics and code.

Keywords

Game development; card game; Godot; programming languages

Obsah

[Úvod 6](#_Toc187049920)

[1 Žánr karetních her 7](#_Toc187049921)

[2 Teorie práce v enginu Godot 8](#_Toc187049922)

[2.1 Popis jazyka GDScript 8](#_Toc187049923)

[2.2 Objekty 9](#_Toc187049924)

[2.2.1 Node, scéna, signál 10](#_Toc187049925)

[2.2.2 Node2D a Node3D 11](#_Toc187049926)

[2.2.3 Area2D a CollisionObject2D 11](#_Toc187049927)

[2.2.4 BoxContainer a Button 11](#_Toc187049928)

[2.2.5 TextureRect, ColorRect a Label 11](#_Toc187049929)

[3 Rozčlenění dokumentů, jména proměnných a funkcí 12](#_Toc187049930)

[4 Vytváření designu hlavních postav 14](#_Toc187049931)

[5 Design nepřátel 16](#_Toc187049932)

[6 Design karet 17](#_Toc187049933)

[7 Mapa 19](#_Toc187049934)

[8 Ikonky 21](#_Toc187049935)

[9 Programování mechanik hry 22](#_Toc187049936)

[9.1 Mechanismus karet a jejich stádií 22](#_Toc187049937)

[9.1.1 Základní stádium 23](#_Toc187049938)

[9.1.2 Zakliknuté stádium 23](#_Toc187049939)

[9.1.3 Posuvné stádium 24](#_Toc187049940)

[9.1.4 Uvolněné stádium 24](#_Toc187049941)

[9.2 Mechanismus karetních vlastností 25](#_Toc187049942)

[9.3 Hráč 26](#_Toc187049943)

[9.3.1 Tah hráče 28](#_Toc187049944)

[9.4 Nepřítel 29](#_Toc187049945)

[9.4.1 Akce nepřítele 29](#_Toc187049946)

[9.4.2 Tah nepřítele 31](#_Toc187049947)

[9.5 Úrovně 32](#_Toc187049948)

[10 Průběh hry 33](#_Toc187049949)

[Závěr 35](#_Toc187049950)

Úvod

Teoretická část popsána v následujících kapitolách přibližuje vývoj praktické části odborné práce. Skládá se ze tří hlavních částí. V první části jsou zahrnuty informace o herním enginu (motoru), ve kterém byla hra tvořena, a jeho jednotlivých dílech. Následuje popis použitého jazyka GDScript a jeho porovnání s běžnějším CSharpem a Pythonem. V druhé části je ukázáno, jak probíhal postup tvoření designu karet, prostředí a postav. Důraz je kladen na kreativní proces, volbu nástrojů a na rozdíly ve způsobech zpracovávání jednotlivých designů. Poslední část zahrnuje vysvětlení kódů a použitých mechanik, jež jsou zásadní pro funkčnost hry. Zabývá se kartami, hráčem, nepřítelem a důležitými scénami ve hře. Nakonec je popsáno, jak by celý průběh hry měl vypadat.

Téma tvorby karetních her, zejména pomocí herního enginu (programu) Godot a jazyka GDScript, je relativně nové a méně prozkoumané v porovnání s jinými částmi herního průmyslu. Dosavadní odborné práce se většinou zaměřují na všeobecný popis herních žánrů nebo na populárnější enginy (programy) jako Unity či Unreal Engine. Z tohoto důvodu se většina dostupných zdrojů týkajících se Godotu a GDScriptu nachází v anglickém jazyce. České podklady jsou v těchto oblastech omezené nebo zcela chybí. To vedlo k nutnosti provést podrobnou analýzu zahraničních zdrojů, mezi nimiž dominují oficiální dokumentace, fóra a komunitní příspěvky. Při rešerši nebyl nalezen jediný využitelný tištěný zdroj.

Při této práci byly použity metody zahrnující analýzu dostupných elektronických zdrojů a dokumentace, praktického experimentování s enginem (programem) Godot a empirického testování vytvořených mechanik.

Hlavním cílem této práce je nejen vytvoření funkční karetní hry s bojovým a karetním systémem, ale také poskytnutí podrobného návodu a reflexe celého vývojového procesu. Tento dokument by mohl sloužit jako inspirace či učební materiál pro další vývojáře, kteří se rozhodnou věnovat tvorbě her podobného typu.

1. Žánr karetních her

Nejprve je důležité začít vysvětlením, proč byl pro tento projekt vybrán žánr karetních her. Karetní hry byly vždy velmi oblíbené nejen ve videoherním průmyslu, ale i mezi stolními hrami. Většinou jsou poměrně jednoduché na pochopení a různé variace strategií a vytváření vlastních balíčků jim přidávají na zábavnosti. Důvody, proč byla hra vytvořena jen pro jednoho hráče spočívá v pozbytí nutnosti řešit implementaci komunikace mezi zařízeními. Další pozitivní vlastnosti jsou, že hráč nepotřebuje nikoho dalšího, aby si mohl hru užít, a nemusí být ani připojen k internetu. Projekt může být v budoucnu o funkci multiplayer („hra s více hráči“) obohacen.

Mechaniky a design jsou inspirovány hrami jako Slay the Spire nebo Magic: The Gathering. První je velmi úspěšná rougelike („hra na hrdinu“) karetní hra koncipovaná jen pro jednoho hráče. Druhá je stolní karetní hra, jež neztratila popularitu ani po 31 letech od svého vydání (Garfield, 2013).

1. Teorie práce v enginu Godot

Godot je open-source (otevřený software) herní engine (motor), který byl vyvinut na univerzitě MIT a spadá pod její licenci. Jeho největšími výhodami jsou kompatibilita vytvořeného softwaru/hry na všech operačních systémech (včetně těch pro mobilní zařízení) a nulové požadavky na finanční podíl, pokud se autor softwaru rozhodne ho vydat a prodávat. Engine i jeho dokumentace jsou velmi jednoduché na pochopení, a proto je skvělou volbou i pro začátečníky v oblasti herního vývoje. Proto byl vybrán na vývoj této hry.

Programování jednotlivých funkcí je možné standardně ve dvou programovacích jazycích, CSharp a GDScript a za pomoci rozšíření i v C a C++. Další kapitola bude rozebírat právě GDScript.

* 1. Popis jazyka GDScript

GDScript se funkčností velmi podobá jazyku CSharp a svou jednoduchostí zase jazyku Python. Je to vysokoúrovňový, imperativní, objektově orientovaný a postupně typovaný jazyk. Rozdíl oproti CSharpu je jen v postupné typovosti, jinak jsou právě v prvních třech ohledech identické. Byl přímo navržen pro Godot, pro dosažení co nejlepší optimalizace a flexibility (Linietsky, 2014).

Vysokoúrovňový jazyk se definuje jako jazyk, který se svou funkčností podobá lidskému přemýšlení. Programátorům tento typ jazyků velmi usnadňuje vývoj, protože jsou jednodušší na pochopení a lépe čitelné. Tyto jazyky potřebují kompilátor, jenž je „přeloží“ do nízkoúrovňového jazyka čitelného pro počítač.

Dále jazyk využívá dvou programovacích paradigmat, a to imperativní a objektově orientované (dále jen OOP). Obě paradigmata jsou úzce spjata, OOP se v 80. letech 19. století vyvinulo právě z imperativního. Imperativní paradigma je založeno na určité posloupnosti příkazů (krok za krokem), která tvoří algoritmus řešící daný problém (Přispěvatelé Wikipedie, 2021), na rozdíl od objektově orientovaného, jenž využívá objektů schopných odděleně vykonávat jednotlivé úkony a propojovat je v hlavním kódu.

Postupně typované jazyky jsou kategorií spojující dynamické a statické typování. Rozdíl v těchto dvou přístupech je v kontrole chyb. Dynamické ji provádí za běhu programu (tzv. runtime) a statické ji provede před spuštěním. Postupná typovost je unikátní spojení statické bezpečnosti a dynamické flexibility, a také nabízí samotnému vývojáři větší kontrolu nad kódem.

S CSharpem má velmi podobné definování funkcí, jak je zjevné i z obrázků (viz Obrázek 1 a 2). V prvním nadřazeném řádku kódu je jméno funkce a potřebné proměnné. CSharp ještě požaduje navíc definovat, zda je funkce veřejná či soukromá (public/private).

Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky

Obrázek 1: Funkce v GDScriptu

Obsah obrázku text, snímek obrazovky, software

Popis byl vytvořen automaticky

Obrázek 2: Funkce v CSharpu

Zásadním rozdílem, mezi těmito dvěma jazyky je syntax. V ní je GDScript více podobný Pythonu. Nejprve je třeba se zaměřit na pojmenovávání proměnných, u kterých se v GDScriptu většinou využívá tzv. Snake case. Slova jsou napsána malými písmeny a jsou oddělena podtržítkem (příklad z Obrázku 1: char\_stats). U CSharpu se využívá tzv. Pascal case. První písmeno se píše malé a každé další slovo v názvu je odděleno velkým písmenem (příklad z Obrázku 2: aktivniLokace). Další podobností je využití odsazení pro determinování vzájemných závislostí řádků kódu namísto například složených závorek (CSharp). Toho je možné si všimnout u podmínek (tzv. if a else). Jsou vždy nadřazené kódu, jenž se má provést po vyplnění podmínky, a proto je na rozdíl od nich odsazen o jeden tabulátor.

* 1. Objekty

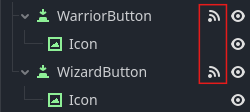
Programy vytvořené v Godotu stojí primárně na jednotlivých objektech, ke kterým mohou být připojovány zdrojové kódy a jsou jimi ovládány. V následujících kapitolách budou nejdříve popsány tři hlavní a poté i další komplexnější objekty, jež byly využity k vývoji.

* + 1. Node, scéna, signál

Node je základní kámen každého projektu. Je to objekt, který může dědit, mít děti nebo být rodičem celé větve (pokud je nějaký objekt podřazen jinému, nazývá se jeho dítětem). Je nespočet typů nodů sloužících k různým účelům například pro kolize, grafiku nebo zvuk.

Větev nodů se nazývá scéna. Obsahuje prostor, na kterém jsou nody umístěné a spojuje je všechny dohromady, aby mohly spolu interagovat. Má možnost být tzv. „zabalena“ a přidána do větve jiné scény mezi nody.

Signály se využívají pro jednoduchou komunikaci mezi nody, i když nejsou navzájem ve vztahu dítě/rodič. Deklarují se podobně jako proměnné, ale mohou obsahovat prostor pro data k přenosu. Signály přidávají programu důležitou přizpůsobivost.



Obrázek : Objekty napojené na signály



Obrázek 4: Objekty v Godotu

* + 1. Node2D a Node3D

Node2D a Node3D jsou objekty velmi podobné jednoduchému nodu. Navíc ale obsahují škálu, rotaci, pozici a viditelnost. Často se využívají jako hlavní nody ve scéně nebo větvi a připojují se k nim obvykle Sprite2D zahrnující grafiku nebo Label s písmem. Mezi nimi je rozdíl jen v počtu dimenzí dané hry.

* + 1. Area2D a CollisionObject2D

Area2D a CollisionObject2D jsou spolu velmi úzce spjaty, protože Area2D (i několik dalších objektů) potřebuje ke své správné funkčnosti CollisionObject2D. Engine bez jeho přidání vždy hlásí chybu, stejně jako když je CollisionObject2D přidán do větve bez Area2D (nebo podobného objektu). Společně zajišťují tzv. kolizi, která je důležitá pro interakci objektů mezi sebou nebo samotného hráče s objektem. CollisionObject2D determinuje tvar kolize a Area2D dokáže zaznamenávat překrytí či náraz cizích kolizí nebo „dotyk“ myši. To lze využít, například pokud má objekt něco udělat po kliknutí či najetí kurzorem.

* + 1. BoxContainer a Button

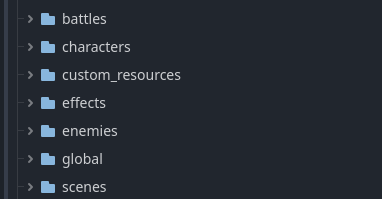
BoxContainer funguje jako nádoba pro objekty, které jsou jejími dětmi. Lépe se pak aranžují do scény, drží mezi sebou stále stejné mezery a lze jimi pohybovat najednou. Existuje více typů těchto kontejnerů, např. VBoxContainer pro vertikální členění, HBoxContainer pro horizontální členění. Button, v překladu tlačítko, funguje velmi intuitivně. Po jeho zmáčknutí spustí funkci anebo vyšle signál. Grafiku, kolizi i text drží sám, a proto není třeba, aby měl děti obstarávající tyto náležitosti. Je možné upravovat jeho vizuál i v různých stádiích, třeba při najetí, zmáčknutí nebo vypnutí.

* + 1. TextureRect, ColorRect a Label

TextureRect a ColorRect jsou velmi podobné objekty a jejich hlavním účelem je měnit texturu a barvu. Využívají se často k vytvoření pozadí pro scénu. Label má také jednoduchý účel. Ten spočívá v držení textu. Může být měněn pomocí kódu a podporuje i odlišné zabarvení či tloušťku jednotlivých znaků.

1. Rozčlenění dokumentů, jména proměnných a funkcí

Velmi zásadní a často opomíjenou částí herního vývoje je rozdělování dokumentů a nazývání proměnných. Jejich důležitost spočívá v přehlednosti napomáhající k mnohonásobnému zrychlení a ulehčení práce pro samotného vývojáře nebo skupinu vývojářů. Postup, jenž byl využit při členění, byl následující. Pokud byl vytvořen kód nebo zdroj, u něhož bylo očekáváno více podobných (kódy karet, scény), vždy byl uložen do složky s nadřazeným jménem. V těchto složkách (viz Obrázek 5) začaly přibývat i podsložky v případě potřeby. Například nepřátelé mají separátní kódy pro jejich schopnosti, proto potřebují každý vlastní složku. V průběhu vývoje byly často vytvářeny nové složky nebo byly soubory vyměňovány a přesouvány tak, aby následná práce s nimi byla co nejsnadnější a nejefektivnější.



Obrázek 5: Složky

U kódu, jeho proměnných a funkcí už by měla být přehlednost uzpůsobena i předpokladu, že ho bude číst jiný programátor, nebo s ním dokonce pracovat. Nejideálnější proto je vybírat jména stručná a jednoznačně popisující. Názvy typu x a y jsou v tomto typu projektů nejméně vhodné.

Obsah obrázku text, snímek obrazovky, menu, Písmo

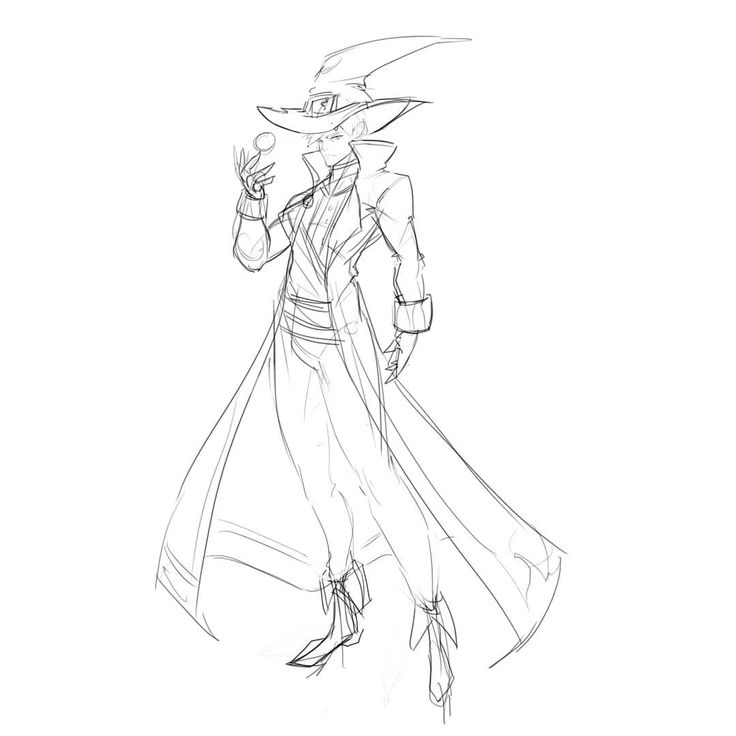
Popis byl vytvořen automaticky

Obrázek 6: Jména funkcí a proměnných

Na předešlém obrázku (viz Obrázek 6) lze zpozorovat, že každá funkce má jméno podle toho, na jaké tlačítko reaguje při jeho stisku. Jméno funkce \_on\_five\_pressed zahrnuje všechny potřebné informace (reaguje na stisknuté tlačítko číslo pět) a je také velmi stručné, takže nedělá kód nepřehledným. Stejně tak signál stage4\_entered nebo campfire\_entered.

1. Vytváření designu hlavních postav

U hlavních postav je nejdříve důležité si ujasnit, co by měly představovat nebo jakými bytostmi by měly být. Ve většině her je standardní výběr z postav bojovník a mág. Proto byl použit i v tomto případě. Po ujasnění postav bylo třeba vybrat předlohu těla, obličeje a barevné spektrum. Od toho se designování odvíjí. V tomto bodě je vše připraveno pro samotné kreslení, jež bylo provedeno v programu Clip Studio Paint.



Obrázek 7: Předloha těla postavy



Obrázek : Finální postava

1. Design nepřátel

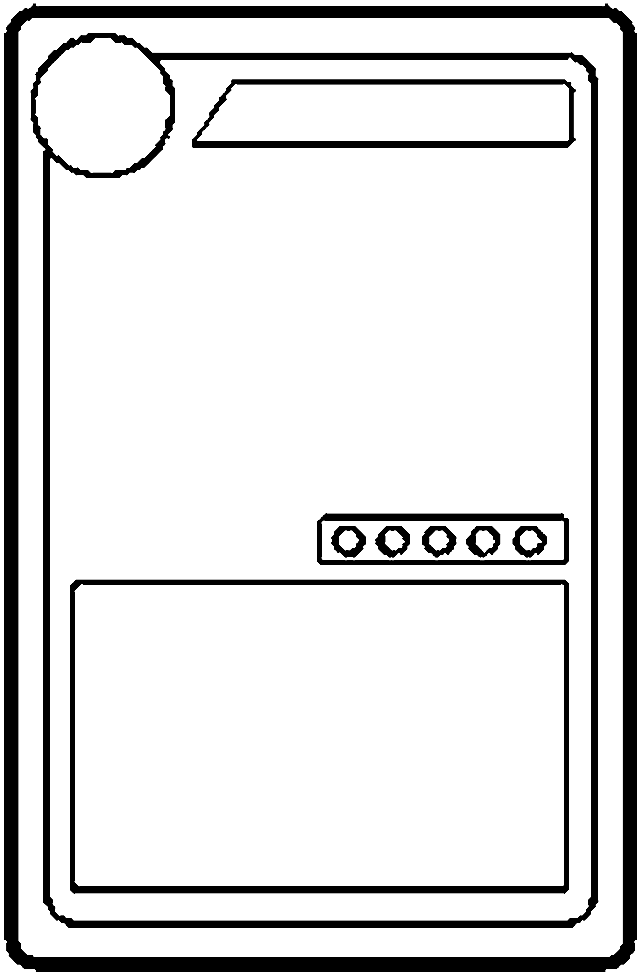
Obsah obrázku tma, světlo, černá, ve tmě

Popis byl vytvořen automatickyU designu nepřítele je nutné nejdříve determinovat, zda má mít humanoidní nebo jinou stavbu těla. Pro tuto hru byly vybrány nehumanoidní nepřátelé, již jsou nějakým způsobem děsiví a nebezpečně vypadající, aby to v hráči vzbuzovalo pocit potřeby se ve hře bránit. Příkladem jsou had, pavouk nebo netopýr. Postup byl dále stejný jako u hlavních postav.

Obrázek 9: Nepřítel – pavouk

1. Design karet

Na začátku designování bylo potřeba vybrat okraj a základní vzor pro všechny karty. Pod to spadá vnitřní kompozice a tvar. Bylo třeba vyhledat předlohu, jež se upraví. K tomu byl využit program Clip Studio Paint, ve kterém si umělci navzájem sdílí různé štětce, reference, návrhy apod. V předloze karty byly upraveny okraje a doprostřed byl přidán kruh, jenž byl následně využit pro raritu karty. Horní volný prostor byl ponechán pro hlavní grafický design karty. Spodní prázdný obdélník slouží pro informaci o ceně karty (mana) a k případnému doplnění textem.

Obsah obrázku skica, Obdélník, bílé, design

Popis byl vytvořen automaticky

Obrázek 10: Předloha karty Obrázek 11: Finální verze karty

Design karet pro horní část byly tvořeny stejným způsobem jako hlavní postavy (viz výše).

Obsah obrázku text, kreslené, nachový, Purpurová

Popis byl vytvořen automaticky

Obrázek : Příklad designu pro horní část karty Obrázek : Celá karta ve hře

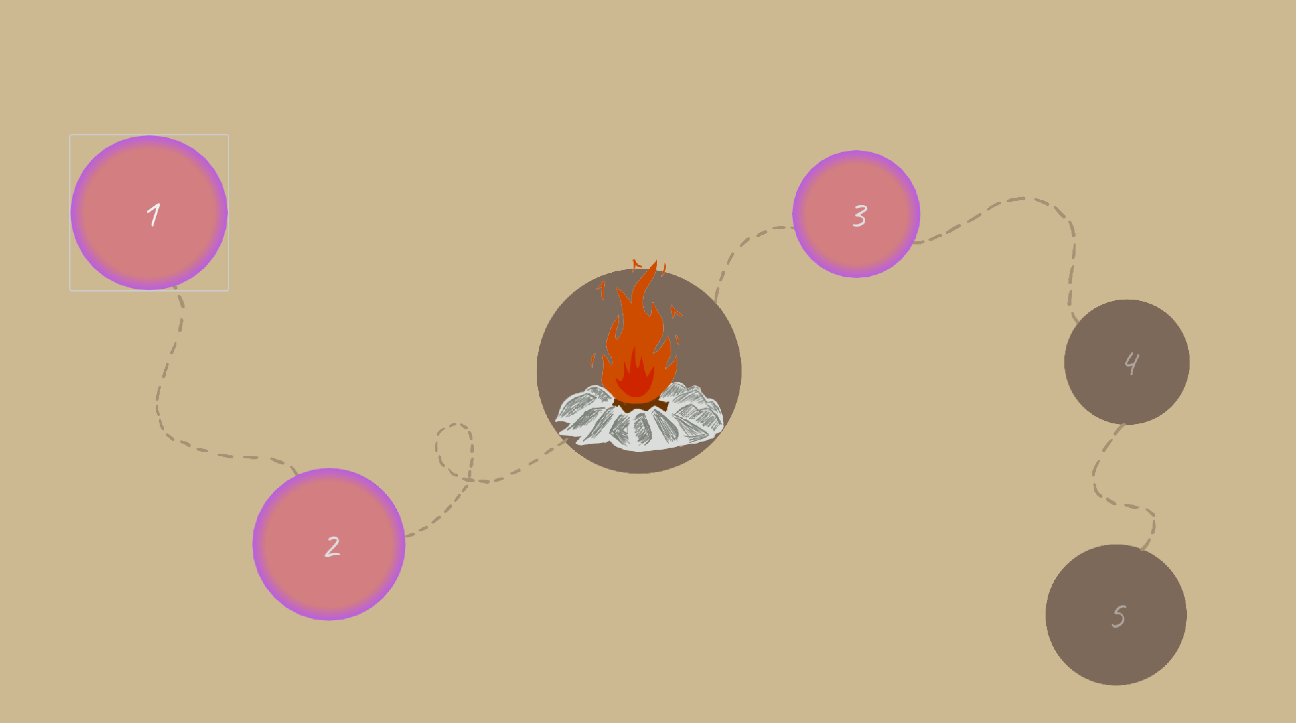
1. Mapa

Mapa je jedna z nejdůležitějších částí hry, protože po každé bitvě se k ní hráč navrací. Inspirací byly mapy pokladu z pohádek, tedy byla koncipována podobně. Předem byla vytvořena paleta barev, které se k sobě navzájem hodily a jejichž kontrast byl dostatečný pro vyniknutí prvků na podkladu. Pro tlačítka úrovní byly využity nástroje v Godotu, se kterými je možné uzpůsobit vizáž i podle toho, co se s nimi děje, např. pokud jsou stisknutá změní barvu. Jediné unikátní místo na mapě je ohniště, jež bylo nakresleno, protože pouhá modifikace přes Godot by nebyla dostatečná. U tohoto stanoviště bylo důležité podtrhnout jeho důležitost pro hráče.

Obsah obrázku kruh, umění, ilustrace

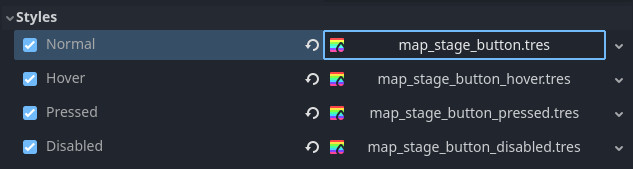
Popis byl vytvořen automaticky

Obrázek 14: Nástroje pro modifikaci tlačítek Obrázek 15: Tlačítko v běžném a stisknutém stavu



Obrázek 16: Mapa s úrovněmi

Pro design byla také využita funkce stylů, usnadňujících práci, aby každé tlačítko nemuselo být designováno jednotlivě. Výsledkem byl ušetřený čas.



Obrázek : Styly

1. Ikonky

Ikonky jsou malé obrázky doplňující například počet životů nebo obranu. Pro jejich design bylo důležité se držet jednoduchosti, aby hráč pochopil, co mají reprezentovat. Klíčovým vodítkem je zákon asociace neboli „co si s danou věcí spojujeme“. Dobrým příkladem je třeba zdraví, pro které byl vybrán obrázek srdce, protože je často spojováno s tímto termínem.

Obsah obrázku kresba, klipart, ilustrace, kreslené

Popis byl vytvořen automaticky

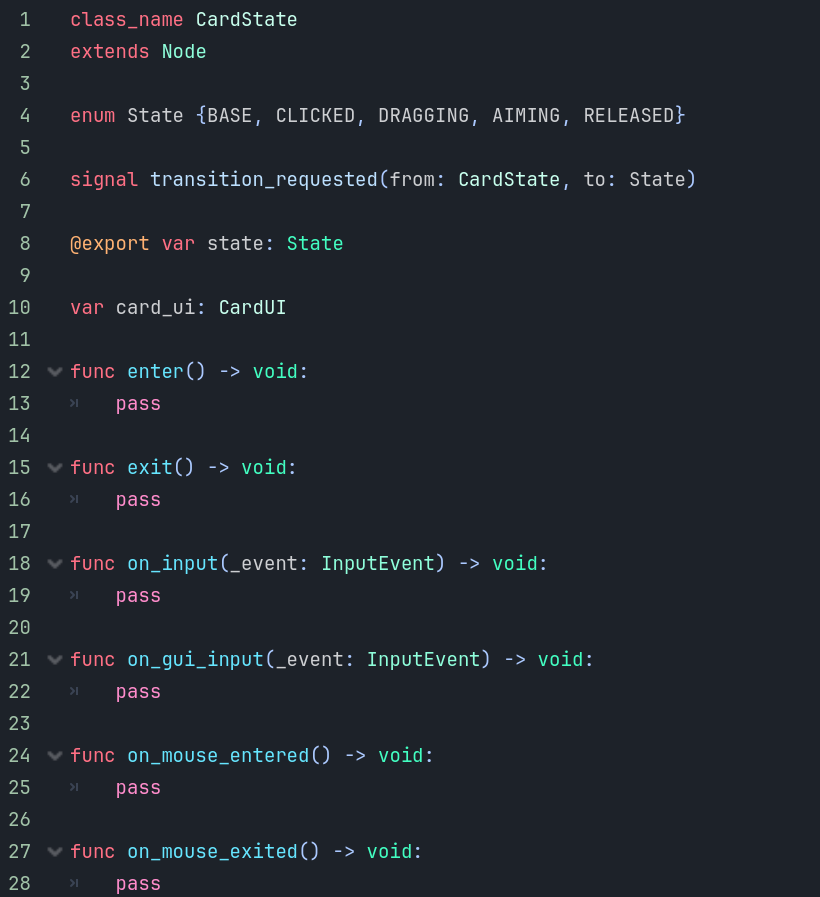
Obrázek 18: Příklad ikonky zdraví Obrázek 19: Příklad ikonky ohniště

1. Programování mechanik hry

Základem celé hry je manipulace s kartami, a proto je třeba na začátek vysvětlit, jak fungují. Dále je třeba se zaměřit na tahy mezi hráčem a nepřítelem a ukázat, jak fungují samotní. V neposlední řadě je důležité vysvětlení fungování útoků entit, přecházení mezi scénami a průběh hry.

* 1. Mechanismus karet a jejich stádií

Pokud má být karta vyložena z ruky a zahrána, musí projít nejprve několika procesy, aby hra fungovala správně. O to se stará tzv. „mechanismus karetních stádií“, který všechny tyto procesy spojuje. Skládá se ze 4 kroků: základního, zakliknutého, posuvného a uvolněného.  
  
Na začátku byla vytvořena class (třída), ze které ostatní stádia dědí funkce a proměnné. V ní byla definována výčtová proměnná, do které byla vložena všechna stádia, aby je mohla samotná karta střídat. První nadefinovaná funkce se jmenuje *enter* (vstup) a určuje to, co se s kartou děje, když do stádia poprvé vstoupí. Další velmi podobná funkce je *exit* (výstup), která definuje, co se stane, když stádium opustí. Potom dvojice funkcí on\_input a on\_gui\_input. Obě se starají o vyhodnocování vstupů, ale on\_gui\_input jen o vstupy, jež se přímo dotýkají karty, zatímco on\_input se stará o všechny. Následující dvojice funkcí on\_mouse\_entered a on\_mouse\_exited monitoruje, zda se myš dotýká karty nebo ne, aby následně bylo možné například zjišťovat její polohu na obrazovce. Všechna stádia jsou na sebe napojena signály a zároveň je spojuje program, který je jejich rodičem a volá je podle toho, co se s kartou děje.



Obrázek : Třída karetních stádií

* + 1. Základní stádium

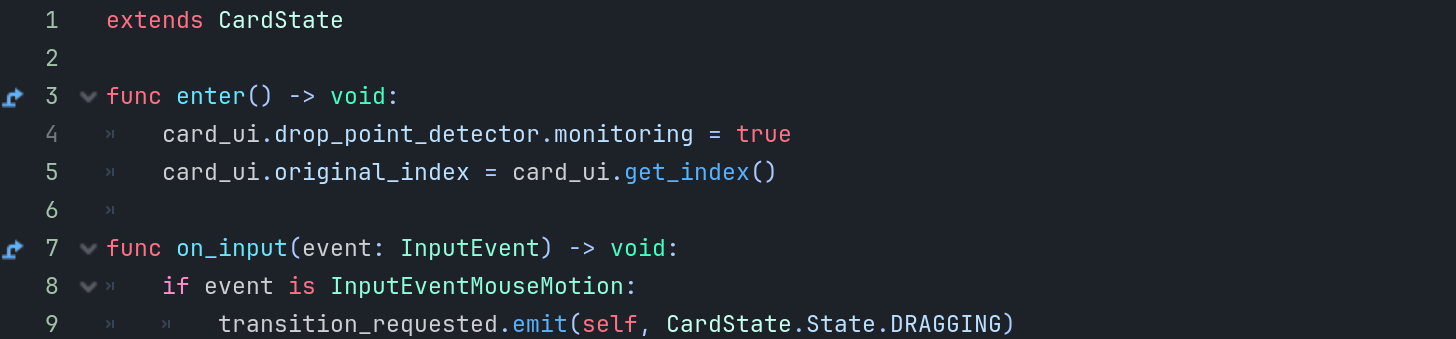
V tomto stádiu je karta vždy, když je v ruce hráče. Po vyhodnocení vstupu myši funkce on\_gui\_input následně zjistí, zda je karta hratelná (např. pokud hráč nemá dostatek many, nebude). V případě doteku myší se karta zabarví, to docílí funkce on\_mouse\_entered, a pokud je podržena levým tlačítkem, přesune se do zakliknutého stádia.



Obrázek : Kód základního stádia

* + 1. Zakliknuté stádium

Kód je zde velmi jednoduchý, protože stádium bývá velmi krátké. Po dosažení toho stádia, funkce enter zajistí, že při stisknutí levého tlačítka myši, karta začne vyhodnocovat kolize. Tato funkce také zjistí index karty, neboli „kde v ruce byla“, aby se mohla vrátit na stejné místo. Pokud je s kartou pohybováno, přesune se do posuvného stádia pomocí funkce on\_input.



Obrázek : Kód zakliknutého stádia

* + 1. Posuvné stádium

Funkce on\_input zajišťuje, aby se karta posouvala s myší, tudíž sleduje její pozici. Dále monitoruje kolize na obrazovce a vstup myši. Pokud hráč uvolní kartu v hracím poli, budou její efekty provedeny. Když hráč stiskne pravé tlačítko myši, bude navrácena do ruky.



Obrázek : Funkce on\_input v kódu posuvného stádia

* + 1. Uvolněné stádium

V kódu tohoto stádia jsou pouze dvě funkce enter a on\_input. Enter zajišťuje vyslání signálu pro provedení efektů a on\_input kontroluje, zda byla karta puštěna v hracím poli. Pokud nebyla, vrátí se zpět do ruky.

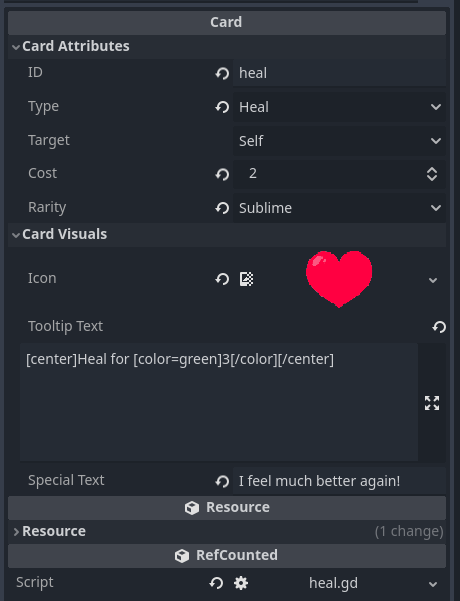
Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky

Obrázek : Kód uvolněného stádia

* 1. Mechanismus karetních vlastností

Každá karta ve hře má svůj unikátní kód a zdroj, které definují její vlastnosti a schopnosti. Ve zdroji je uložených devět informací. První je ID, jež je jménem karty, využívající se pro odlišení. Dále Typ a Cíl (Type a Target) definují, jestli je karta obranná, útočná nebo léčící a také zda má působit na hráče nebo na nepřítele. Cena (Cost) je počet many potřebný k zahrání dané karty. Rarita (Rarity) hraje roli při získávání odměn po vyhrané bitvě, protože čím vyšší rarita, tím menší pravděpodobnost je, že bude v nabídce na konci úrovně. Následující tři atributy jsou jen kosmetickými úpravami. Ikonka (Icon) je vizuál karty, Popis (Tooltip text) říká hráči, co karta dokáže, a Speciální text (Special text) doplňuje kratší citáty z estetických důvodů. Skript (Script) je již zmíněný kód dodávající kartě funkčnost.



Obrázek 25: Karetní atributy

Každý script karty dědí z class (třídy) Card (karta). Funkce \_get\_targets vytváří array (pole) cílů, na které mají karty působit. Funkce play poté následně kartu zahraje a odečte hráči příslušný počet many. Apply\_effects je definováno v každém kódu karty jednotlivě.

Obsah obrázku text, snímek obrazovky, software

Popis byl vytvořen automaticky

Obrázek : Kód třídy Card

Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky

Obrázek : Příklad kódu karty

Například z obrázku (viz Obrázek 26) je zřejmé, že se v kódu nejprve vytvoří efekt léčení, určí jeho hodnotu a poté se provede. Nakonec je třeba zmínit, že každý typ efektu má svoji vlastní třídu, jež definuje funkci execute.

* 1. Hráč

Hráčem manipulují především 2 hlavní kódy: player a player\_handler. Player se zaměřuje spíše na hráče jako objekt samotný a drží v sobě referenci na jeho design a vizuál jeho stats (statistik). První funkce set\_character\_stats definuje jeho stats, jako jsou například životy a mana. Update\_player se využívá k aktualizaci dat o hráči a drží v sobě funkci update\_stats, jež se v tomto případě zaměřuje jen na stats (statistiky). Take\_damage zajišťuje sebrání zdraví hráče při útoku nepřítele a k tomu vázající se animaci. Heal naopak hráči zdraví přidává.

Obsah obrázku text, snímek obrazovky

Popis byl vytvořen automaticky

Obrázek : Kód player

Kód stats se stará o zdraví, štíty a manu hráče a nepřátel. Na začátku hry vždy nastaví na maximum zdraví a manu. V průběhu hry se funkce v kódu využívají na změnu těchto stats (statistik) a jsou zahrnuty i v kódu Player a Enemy (hráč a nepřítel).

Obsah obrázku text, snímek obrazovky

Popis byl vytvořen automaticky

Obrázek : Kód stats

* + 1. Tah hráče

Jako první je v úrovni vždy na řadě hráč. Na začátku se pomocí funkce \_ready připojí signál, jenž zaznamenává zahrání karty, k funkci \_on\_card\_played. Ta ji přesune do odhazovacího balíčku. Následuje činnost funkce start\_battle, která připraví balíček a přidá příslušný počet karet do ruky funkcí draw\_cards. Na konci zavolá start\_turn. Poté se hráči resetuje mana a s kartami v tu chvíli může začít interagovat. Kolo lze ukončit tlačítkem „ukončit tah“, nebo výhrou. Pokud bylo ukončeno tlačítkem, zavolá se funkce end\_turn a začíná hrát nepřítel. Před začátkem každého tahu se hráči doplní mana a chybějící karty do ruky. Mechanika balíku je koncipována tak, aby nikdy karty nedošly, takže pokud jich v balíku není dostatek, použité karty se znovu zamíchají a vrátí zpět. To má na starosti funkce reshuffle\_deck\_from\_discard.

Obsah obrázku text, snímek obrazovky

Popis byl vytvořen automaticky

Obrázek : Část kódu player\_handler

* 1. Nepřítel

Nepřítel má, stejně jako hráč, dva hlavní kódy. Kód enemy se zaměřuje na nepřítele samotného a drží v sobě referenci na jeho design a vizuál jeho stats (statistik) a intent (úmyslu). Funkce set\_current\_action zajišťuje, aby se nad nepřítelem zobrazila akce, kterou provede v dalším kole a její aktualizaci provádí update\_action. Tuto akci provede do\_turn. Set\_enemy\_stats nastavuje statistiky a jejich aktualizaci zajišťuje update\_stats. Funkce take\_damage je využita, pokud je nepřítel zraněn. Update\_enemy aktualizuje celého nepřítele a zahrnuje funkci update\_stats. Akce nepřítele vybírá class (třída) EnemyActionPicker a její funkce jsou v kódu enemy (nepřítel) volány pomocí setup\_ai.

* + 1. Akce nepřítele

To, co nepřítel bude ve svém tahu dělat, určují dvě zásadní kritéria: stav ve hře a náhodný výběr. Nepřítel má dva typy akcí. Jedna je podmíněná stavem hry a je nadřazená těm, které mohou být náhodně vybrány. Například v případě, že má nepřítel podmíněnou akci, jež se musí vykonat, pokud má počet životů pod určitou hodnotou (jestli ještě nebyla za hru použita), nadřadí ji jakékoliv jiné činnosti, kterou si předtím vylosoval.

Losování akcí funguje velmi jednoduše. Při předpokladu, že funkce get\_action vyhodnotí pomocí get\_first\_conditional\_action stav hry jako neodpovídající žádné podmíněné akci, nepřítel začne losovat funkcí get\_chance\_based\_action akci nepodmíněnou. Všechny jeho nepodmíněné činnosti mají nějakou číselnou hodnotu, která určuje míru pravděpodobnosti, že bude vylosována. Poté je jim přiřazena tzv. akumulovaná hodnota, jež je součtem všech předcházejících hodnot včetně vlastní. To zařizuje funkce setup\_chances. Nepřítel pak losuje číslo mezi nulou a celkovým součtem hodnot akcí. Aby se akce mohla vykonat, musí vylosované číslo být menší či rovno její akumulované hodnotě a zároveň větší než akumulovaná hodnota činnosti předcházející. Algoritmus jednoduše hledá první číslo, které je větší než vylosovaná hodnota.

Příklad:

Existují akce útok, obrana a léčení. Akce útok má hodnotu 3, obrana hodnotu 1 a léčení hodnotu 2. Celková hodnota všech akcí je 6. Akumulované hodnoty akcí jsou:

útok = 3,

obrana = 4,

léčení = 6.

Pokud nepřítel vylosuje číslo 5, vykoná se akce léčení. Pokud vylosuje 2, zaútočí apod.

Obsah obrázku text, snímek obrazovky, menu

Popis byl vytvořen automaticky

Obrázek : Část kódu enemy\_action\_picker

Každý samotný kód akce dědí z class (třídy) EnemyAction. Tato třída má několik atribut: intent (úmysl), type (typ), chance\_weight (tíha náhody) a accumulated\_weight (akumulovaná tíha). Dále obsahuje dvě funkce. Is\_performable, kterou používají jen podmíněné akce, v nichž jsou podmínky pro provedení. Perform\_action, jež determinuje, co se stane při vykonání této činnosti. Například zraní hráče o 5 zdraví nebo blokuje 2x.

Obsah obrázku text, snímek obrazovky

Popis byl vytvořen automaticky

Obrázek : Kód enemy\_action

* + 1. Tah nepřítele

Po hráči začíná kolo nepřítele. O kola nepřátel se stará kód enemy\_handler. Nepřítel nad sebou ukazuje, co bude v tahu dělat, například zda bude útočit, nebo se bránit. Pokud je ve hře nepřátel více, každý hraje jednou a jejich akce se vykonají podle toho, jak jsou za sebou seřazeni ve větvi. Jejich střídání provádí funkce \_on\_enemy\_action\_completed. Na začátku hry jsou připraveni funkcí setup\_enemies. Začátek kola nepřátel obstarává funkce start\_turn. Po odehrání všech nepřátel se jejich tah ukončuje a cyklus se znovu opakuje, dokud nejsou všichni poraženi nebo dokud neporazí hráče.

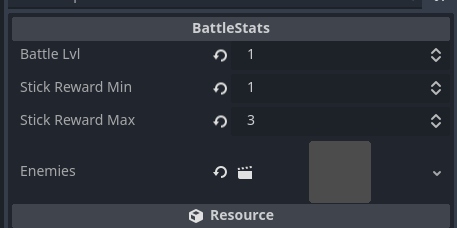
Obsah obrázku text, snímek obrazovky

Popis byl vytvořen automaticky

Obrázek : Část kódu enemy\_handler

* 1. Úrovně

Každá úroveň má unikátní nepřátele a průběh. Začíná se od první, a pokud se hráči podaří vyhrát, odemkne se mu další. Ke každé se váže zdroj, který definuje, jak daleko je na mapě úroveň (Battle Lvl), minimální a maximální počet klacíků (Sticks Reward Min/Max), jež může hráč získat, a nakonec je také přiložena scéna s nepřáteli (Enemies).



Obrázek 34: Zdroj úrovně

1. Průběh hry

Na začátku je hráč uvítán scénou, ve které je uvedeno jméno hry a tlačítko pro spuštění, jež ho pošle do scény s výběrem postavy. Zde nalezne dva charaktery. Tlačítka v této scéně jsou napojena na kód a po úspěšné selekci se zavolá funkce set\_current\_character.

Obsah obrázku grafický design, ilustrace, snímek obrazovky, kreslené

Popis byl vytvořen automaticky

Obrázek : Scéna s výběrem postav

Po zmáčknutí tlačítka „start“ se zavolá funkce \_on\_start\_button\_pressed, vytvoří se nová instance třídy RunStartup a před hráčem se objeví mapa. RunStartup v sobě drží informace o hráči a přenáší je celou hrou. Na mapě lze stisknout tlačítko s číslem 1, pod nímž se ukrývá první úroveň. Pokud se hráči zadaří porazit všechny nepřátele, dostane 2 odměny – kartu a klacíky. Může si vybrat jednu ze tří karet a klacíky mu připadnou všechny. Po zdolání druhé úrovně se odemkne ohniště zajišťující léčení výměnou za klacíky. Celá hra končí výhrou všech úrovní. Pokud hráč zemře, vše se vyresetuje a začíná se opět výběrem postavy.

O změnu mezi scénami mapy, ohniště a úrovní se stará class (třída) Run. Na začátku se zavolá funkce \_setup\_event\_connections, která připojí signály k tlačítkům na mapě, aby se po kliknutí na ně objevila úroveň nebo ohniště. Pokud je zmáčknuto tlačítko úrovně, zavolá se funkce \_on\_battle\_room\_entered, jež zapne samotnou bitvu. Tato funkce v sobě volá \_change\_view, která vytvoří instanci scény a schová mapu. Funkce \_show\_map ji zase dokáže ukázat. Funkce \_on\_battle\_won se stará o to, aby se hráči ukázaly odměny po výhře, a také monitoruje, kolik úrovní zdolal, aby bylo možné determinovat konec hry. O horní lištu s informacemi, jako je počet životů a klacíků, se stará funkce \_setup\_top\_bar.

Obsah obrázku text, snímek obrazovky

Popis byl vytvořen automaticky

Obrázek : Část kódu run

Závěr

Cíl vytvořit funkční hru se podařil splnit. Během práce se však vyskytlo několik překážek, které bylo třeba vyřešit, aby projekt mohl být úspěšně dokončen. Největším problémem byla absence odpovídajících překladů anglických termínů do češtiny, které bylo nutno přeložit na základě analogického odvození od již existujících názvů. Dalším problémem byla oprava chyb v praktické části práce. Tyto chyby byly buď úspěšně vyřešeny, nebo byly nefunkční části kódu odstraněny. Projekt má značný potenciál k dalšímu rozvoji, například přidáním nových postav, nepřátel, úrovní či dalších herních mechanik, jako jsou pokročilejší karetní efekty nebo komplexnější nepřátelské útoky.

Práce vykazuje řadu úspěchů, které přesahují pouhé splnění cíle. Jedním z nich je úspěšné přeložení technických termínů do českého jazyka, čímž byla vyplněna mezera v dostupné terminologii. Tento krok může sloužit jako cenný zdroj pro další vývojáře, kteří se setkají s podobnými problémy.

Dalším úspěchem je vytvoření detailního popisu práce s enginem (programem) Godot, který má potenciál sloužit jako tutoriál pro začínající i pokročilé vývojáře. Vysvětlení herních mechanik v této práci představuje přínos pro další programátory, kteří mohou na těchto základech stavět své projekty.

Aby byla tato práce dostupná veřejnosti, byl celý projekt uložen na veřejném GitHub repositáři. Na tomto repositáři je k dispozici nejen zdrojový kód a teoretická část práce, ale také zkompilovaná hra, kterou si mohou zájemci sami vyzkoušet. Tento krok podtrhuje důležitost sdílení znalostí a výsledků, přispívajících k rozvoji komunity vývojářů a zároveň poskytujících zpětnou vazbu pro další zlepšování projektu.

Celkově lze tuto práci hodnotit jako velmi úspěšnou, protože splnila svůj hlavní cíl a zároveň přinesla řadu vedlejších přínosů, které mohou být užitečné pro ostatní.

Zdroje

**Garfield, Richard. 2013.** The Creation of Magic: The Gathering. *Magic: The Gathering.* [Online] Wizards of the Coast LLC, 3. 12. 2013. [Citace: 13. 11. 2024.] https://magic.wizards.com/en/news/making-magic/creation-magic-gathering-2013-03-12.

**Gulacsi, Adam. 2023-2024.** *Godot 4 Intermediate Card Game Course.* [Video] 2023-2024.

**Linietsky, Juan. 2014.** *Godot docs.* [Online] 2014. [Citace: 3. 6. 2024.] https://docs.godotengine.org/en/stable/index.html.

**Přispěvatelé Wikipedie. 2021.** *Imperativní programování.* [Online] Wikipedie: Otevřená encyklopedie., 3. 12. 2021. [Citace: 24. 9. 2024.] https://cs.wikipedia.org/wiki/Imperativní\_programování.

**Rogers, Paul. 2024.** *41 Video Gaming Statistics 2024 (Users, Market Size & Revenue).* [Online] Answeriq.com, 21. 4. 2024. [Citace: 08. 04. 2024.] https://www.answeriq.com/video-gaming-statistics/.

Obrázky

Obrázky, u nichž není uvedená citace pochází z archivu autorky

[Obrázek 1: Funkce v GDScriptu 9](#_Toc186903853)

[Obrázek 2: Funkce v CSharpu 9](#_Toc186903854)

[Obrázek 3: Objekty napojené na signály 10](#_Toc186903855)

[Obrázek 4: Objekty v Godotu 10](#_Toc186903856)

[Obrázek 5: Složky 12](#_Toc186903857)

[Obrázek 6: Jména funkcí a proměnných 12](#_Toc186903858)

[Obrázek 7: Předloha těla postavy 14](#_Toc186903859)

Simple wizard design today :D. Online. 2019. Dostupné z: Instagram, https://www.instagram.com/p/BtRiNJRlcrt/?utm\_source=ig\_share\_sheet&epik=dj0yJnU9RnVIU1VPdnZ0QUJGMWs3WHB0aTRWOFZvaVVraXNzcnQmcD0wJm49RUhwNXBKeHY2RFk0MjRualVpdXdWZyZ0PUFBQUFBR2RGeU1F. [cit. 2024-11-26].

[Obrázek 8: Finální postava 15](#_Toc186903861)

[Obrázek 9: Nepřítel – pavouk 16](#_Toc186903862)

[Obrázek 10: Předloha karty 17](#_Toc186903863)

Card Painting (カード絵). Online. 2023. Dostupné z: CLIP STUDIO ASSETS, https://assets.clip-studio.com/en-us/detail?id=2027118. [cit. 2024-11-26].

[Obrázek 11: Finální verze karty 17](#_Toc186903864)

[Obrázek 12: Nástroje pro modifikaci tlačítek 19](#_Toc186903865)

[Obrázek 13: Tlačítko v běžném a stisknutém stavu 19](#_Toc186903866)

[Obrázek 14: Mapa s úrovněmi 20](#_Toc186903867)

[Obrázek 15: Styly 20](#_Toc186903868)

[Obrázek 16: Příklad ikonky zdraví 21](#_Toc186903869)

[Obrázek 17: Příklad ikonky ohniště 21](#_Toc186903870)

[Obrázek 18: Třída karetních stádií 22](#_Toc186903871)

[Obrázek 19: Kód základního stádia 23](#_Toc186903872)

[Obrázek 20: Kód zakliknutého stádia 23](#_Toc186903873)

[Obrázek 21: Funkce on\_input v kódu posuvného stádia 24](#_Toc186903874)

[Obrázek 22: Kód uvolněného stádia 24](#_Toc186903875)

[Obrázek 23: Karetní atributy 25](#_Toc186903876)

[Obrázek 24: Kód třídy Card 26](#_Toc186903877)

[Obrázek 25: Příklad kódu karty 26](#_Toc186903878)

[Obrázek 26: Kód player 27](#_Toc186903879)

[Obrázek 27: Kód stats 28](#_Toc186903880)

[Obrázek 28: Část kódu player\_handler 29](#_Toc186903881)

[Obrázek 29: Část kódu enemy\_action\_picker 30](#_Toc186903882)

[Obrázek 30: Kód enemy\_action 31](#_Toc186903883)

[Obrázek 31: Část kódu enemy\_handler 32](#_Toc186903884)

[Obrázek 32: Zdroj úrovně 32](#_Toc186903885)

[Obrázek 33: Scéna s výběrem postav 33](#_Toc186903886)

[Obrázek 34: Část kódu run 34](#_Toc186903887)