SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

2D hra - JumpAdventure

Autor: Filip Kožaný

Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování

Třída: IT4

Školní rok: 2024/25

Poděkování

Děkuji panu učiteli Mgr. Markovi Lucnemu a Ing. Petru Grussmanovi za rady

Prohlášení

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým a prezentačním účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 1. 1. 2025

. . . . . .

Podpis autora

Abstrakt

Tato práce se zaměřuje na návrh a vývoj 2D plošinové hry, která je postavena na jednoduchém a zábavném herním mechanizmu – skákání. Hráč ovládá postavu, jejímž cílem je překonávat různé překážky, zdolávat platformy a dosáhnout konce jednotlivých úrovní. Herní design klade důraz na přesné načasování, pohyb a rostoucí obtížnost, která motivuje hráče k zlepšování svých dovedností.

Hra zahrnuje několik úrovní, které jsou postupně složitější, s dynamickými prvky, jako jsou pohyblivé platformy, nebezpečné pasti a interaktivní objekty. Důležitým aspektem je také vizuální styl, který vytváří poutavé prostředí, a soundtrack, který podporuje atmosféru hry.

Cílem projektu je vytvořit zábavnou a návykovou hru, která bude přístupná široké škále hráčů, od příležitostných hráčů až po nadšence, kteří hledají výzvy. Hra demonstruje základní principy herního designu a poskytuje vhled do tvorby 2D herních mechanik.

Klíčová slova

2D plošinovka, skákání, herní design, pohyblivé platformy, obtížnost, herní mechaniky, vizuální styl, herní vývoj, interaktivní objekty, zábavná hra.

Abstract

This project focuses on the design and development of a 2D platformer game built around a simple and engaging gameplay mechanic: jumping. The player controls a character whose goal is to overcome various obstacles, traverse platforms, and reach the end of each level. The game design emphasizes precise timing, movement, and progressively increasing difficulty, encouraging players to improve their skills.

The game features multiple levels that become more challenging over time, with dynamic elements such as moving platforms, hazardous traps, and interactive objects. A key aspect of the project is the visual style, which creates an immersive environment, along with a soundtrack that enhances the game's atmosphere.

The objective of the project is to create an enjoyable and addictive game that is accessible to a wide range of players, from casual gamers to enthusiasts seeking challenges. The game demonstrates fundamental principles of game design and provides insight into the development of 2D game mechanics.

Keywords

2D platformer, jumping, game design, moving platforms, difficulty, game mechanics, visual style, game development, interactive objects, fun game.

Obsah

Úvod **3**

1 TEOTETICKÁ ČÁST **5**

1.1 Co je 2D hra 5

1.2 Co je to pixel art 5

1.3 Co je Godot engine 6

1.4 Co je Krita 6

1.5 Co je to Tilesetter 6

1.6 Historie herniho vývoje 6

2 TECHNOLOGIE **9**

2.1 Godot 9

2.2 Krita 10

2.3 Tilesetter 10

3 Grafická část projektu (Frontend) **11**

3.1 Kresba textur 11

3.2 TextureMap 11

3.3 Vytváření scén a Node 11

4 Backend **13**

4.1 Pohyb a animace charakteru hráče 13

4.2 Příklad kódu 13

4.3 Intro 15

4.4 Menu 15

Úvod

Tato dokumentace se věnuje návrhu, vývoji a implementaci 2D plošinové hry, která staví na základní herní mechanice skákání. Cílem projektu je vytvořit hru, která je jednoduchá na ovládání, ale zároveň nabízí postupně rostoucí obtížnost a výzvy, jež udrží zájem hráče po celou dobu hraní.

Hra je navržena jako návyková zábava pro široké spektrum hráčů, od příležitostných hráčů hledajících relaxaci, až po zkušenější uživatele toužící po zdokonalení svých reflexů a přesnosti. Projekt si klade za cíl nejen zprostředkovat herní zážitek, ale také demonstrovat praktické využití herních mechanik, designových prvků a technologických postupů při tvorbě 2D her.

V dokumentaci jsou popsány všechny klíčové aspekty vývoje hry – od počáteční analýzy a návrhu herního konceptu, přes implementaci herních mechanik a grafiky, až po testování a optimalizaci. Projekt rovněž zdůrazňuje důležitost herní estetiky a zvukového doprovodu, které dohromady vytvářejí poutavé herní prostředí.

Dokumentace slouží jako průvodce nejen pro pochopení celého procesu vývoje, ale také jako zdroj inspirace pro podobné projekty v oblasti herního designu.

1 TEOTETICKÁ ČÁST

1.1 CO JE 2D HRA

2D hra je videohra, která se odehrává ve dvourozměrném prostoru, kde jsou herní objekty reprezentovány na rovině s osami X (vodorovná osa) a Y (svislá osa). Na rozdíl od 3D her, které zahrnují také hloubkovou osu (Z), 2D hry používají ploché grafické prvky a jednoduchou perspektivu.

Hratelnost v 2D hrách se obvykle odehrává z pohledu shora (top-down), z boku (side-scroller), nebo izometrického pohledu, kde objekty vypadají jako trojrozměrné, ale zachovávají si omezení 2D prostoru. Mezi nejběžnější typy 2D her patří plošinovky, adventury, puzzle hry, akční hry, arkádové tituly nebo strategické hry.

1.2 CO JE TO PIXEL ART

Pixel art je forma digitálního umění, která se zaměřuje na vytváření obrázků a grafiky prostřednictvím jednotlivých pixelů, což jsou nejmenší jednotky obrazu na obrazovce. V pixel artu jsou tyto pixely záměrně viditelné a využívají se k vytvoření detailních nebo stylizovaných vizuálů, často s retro estetikou.

1.2.1 Použití pixel artu

- Videohry: Pixel art je populární v nezávislých hrách (např. Stardew Valley, Celeste nebo Terraria) i retro hrách (např. Super Mario Bros., Pac-Man).
- Animace: Pixel art lze animovat pohybem jednotlivých pixelů, což je časté u postav, objektů nebo efektů ve hrách.
- Digitální ilustrace: Výtvarníci vytvářejí statické obrazy, od minimalistických portrétů až po složité krajiny.

Pixel art je skvělý způsob, jak vyjádřit kreativitu, a je ideální pro malé týmy nebo nezávislé vývojáře her, kteří chtějí vytvořit vizuálně zajímavý obsah s minimálními zdroji.

1.3 Co je Godot engine

Godot Engine je open-source herní engine určený pro vývoj 2D i 3D her. Je známý svou flexibilitou, jednoduchostí a tím, že je zdarma k použití, a to i pro komerční projekty. Godot je vyvíjen komunitou a financován především prostřednictvím darů na platformách jako Patreon nebo GitHub Sponsors.

1.4 Co je Krita

Krita je open-source a zdarma dostupný software určený především pro digitální malbu, kreslení a tvorbu ilustrací. Je vysoce oblíbený mezi profesionálními i amatérskými umělci díky svým výkonným funkcím, snadnému použití a pravidelným aktualizacím. Krita je vyvíjena komunitou pod licencí GNU General Public License (GPL).

1.5 Co je to Tilesetter

Tilesetter je nástroj pro snadné a efektivní vytváření dlaždicových (tile-based) map a jejich správu. Používá se hlavně při tvorbě 2D her, kde herní prostředí (například podlahy, stěny, platformy nebo pozadí) sestává z opakujících se dlaždic.

1.5.1 Hlavní vlastnosti Tilesetteru

- Automatické generování dlaždic
- Podpora dlaždicových map
- Jednoduché uživatelské rozhraní
- Export pro různé herní enginy
- Podpora dlaždic s přechody

1.6 Historie herniho vývoje

Historie herního vývoje sahá od 50. let, kdy vznikaly první experimentální hry, jako OXO nebo Tennis for Two, přes zlatý věk arkádových her 70. let, kdy hry jako Pong a Space Invaders získaly obrovskou popularitu, až po 80. léta, kdy konzole jako NES s titulem Super Mario Bros. zachránily herní průmysl po jeho krizi. 90. léta přinesla 3D grafiku, ikonické hry jako Doom

a Super Mario 64, a nástup CD-ROM. Od roku 2000 začaly dominovat online multiplayer hry (např. World of Warcraft) a nezávislé projekty, jako Minecraft, ukázaly sílu kreativního designu. Herní technologie dnes zahrnují pokročilou grafiku (např. ray tracing), cloudové hraní a virtuální realitu, zatímco distribuce přes digitální platformy umožňuje přístup širokému publiku. Herní průmysl tak prošel od jednoduchých pixelových her až po komplexní projekty s rozpočty konkurenčními hollywoodským filmům.

2 TECHNOLOGIE

2.1 GODOT

Godot Engine je open-source herní engine, který nabízí širokou škálu funkcí pro vývoj 2D i 3D her. Mezi jeho klíčové vlastnosti patří:

- **Podpora více platforem:** Umožňuje export na Windows, macOS, Linux, Android, iOS, web (HTML5) a konzole.
- **Vizualní editor:** Intuitivní grafické uživatelské rozhraní pro návrh scén, animací a uživatelského rozhraní.
- **GDScript:** Vlastní jednoduchý skriptovací jazyk podobný Pythonu, s podporou dalších jazyků jako C#, VisualScript a dalších.
- **Node-based systém:** Hierarchická struktura uzlů usnadňuje správu herních objektů a scén.
- **2D i 3D podpora:** Nabízí pokročilé funkce pro oba režimy, včetně 2D fyziky, světel, tilemap a 3D renderování s ray tracingem.
- **Asset pipeline:** Integrovaná podpora pro import a správu 2D/3D assetů a integraci externích nástrojů.
- **Modularita:** Flexibilní systém umožňuje snadné rozšíření funkcionality prostřednictvím pluginů.
- **Open-source:** K dispozici pod licencí MIT, což zajišťuje plnou kontrolu nad kódem a bezplatné využití pro komerční účely.

Godot Engine je oblíbený zejména mezi nezávislými vývojáři díky své přístupnosti, výkonu a širokým možnostem pro tvorbu her všech žánrů.

2.2 KRITA

Krita je open-source grafický editor zaměřený na digitální malbu a ilustraci. Mezi její hlavní vlastnosti patří:

- **Podpora vrstev:** Práce s různými typy vrstev (bitmapové, vektorové, textové).
- **Pokročilé štětce:** Více než 100 přednastavených štětců a nástroje pro jejich přizpůsobení.
- **Podpora animace:** Vytváření frame-by-frame animací přímo v editoru.
- **Barevné nástroje:** Správa barevných palet, podpora HDR a různých barevných modelů (RGB, CMYK).
- **Správa formátů:** Export a import široké škály formátů, včetně PSD, PNG a TIFF.
- **Open-source:** Zdarma a pod licencí GPL, přístupná pro Windows, macOS a Linux.

Krita je oblíbená mezi ilustrátory, komiksovými umělci a animátory díky své výkonnosti a přizpůsobivosti.

2.3 TILESETTER

Tilesetter je nástroj pro tvorbu a správu dlaždicových map v 2D hrách. Mezi jeho hlavní vlastnosti patří:

- **Automatické generování dlaždic:** Vytváří různé kombinace dlaždic na základě základních tvarů.
- **Podpora různých formátů:** Umožňuje export do formátů kompatibilních s herními enginy jako Godot, Unity nebo GameMaker.
- **Podpora tilemap:** Pomáhá při vytváření map, které využívají opakující se dlaždice (např. terén, cesty).
- **Intuitivní editor:** Uživatelsky přívětivé rozhraní pro navrhování a úpravu dlaždic.
- **Přechody mezi dlaždicemi:** Automaticky generuje plynulé přechody mezi různými typy terénů (tráva, písek, voda).

Tilesetter je oblíbený pro svou efektivitu při tvorbě herních úrovní a snadnou integraci s populárními herními enginy.

3 GRAFICKÁ ČÁST PROJEKTU (FRONTEND)

Grafická část projektu zahrnuje návrh herních postav, prostředí a animací, které jsou klíčové pro vizuální zážitek hráče. Pro tvorbu spritek a tilemap používáme nástroje jako Krita pro malbu a Tilesetter pro správu dlaždicových map. Animace jsou vytvořeny buď pomocí frame-by-frame techniky, nebo pomocí kostrových animací pro komplexní pohyby postav.

3.1 KRESBA TEXTUR

Kresba 2D pixel art textur zahrnuje vytváření grafiky, kde každý pixel je pečlivě umístěn, aby tvořil vizuálně atraktivní a funkční obraz. Používají se nástroje jako Krita nebo Aseprite k vytváření pixelových spritek a textur, které jsou následně aplikovány na herní objekty nebo tilemapy.

3.2 TEXTUREMAP

V této 2D hře jsou 2D Texture Mapy vytvořeny pomocí nástroje Tilesetter, který umožňuje efektivní správu dlaždicových map. Tilesetter generuje opakující se textury pro terénní prvky (tráva, cesty) a objekty (stromy, budovy), a zajišťuje plynulé přechody mezi dlaždicemi. Výsledné tilemapy jsou použity k vytvoření vizuálně atraktivního herního prostředí.

3.3 VYTVÁŘENÍ SCÉN A NODE

V Godot Engine je tvorba herních scén založena na systému uzlů (nodes). Scény jsou složeny z hierarchie uzlů, kde každý uzel reprezentuje konkrétní prvek nebo objekt v hře, jako je postava, kamera, nebo tlačítko v uživatelském rozhraní.

3.3.1 Vytváření scén

Scéna v Godotu je kombinací několika uzlů, které spolupracují a vytvářejí herní objekt nebo celou úroveň. Může to být například 2D nebo 3D prostředí, postava nebo herní menu. Každý uzel může mít svou funkci, jako je vykreslování grafiky, detekce kolizí, nebo ovládání animací.

3.3.2 Node (Uzel)

Každý uzel v Godotu je základní stavební prvek scény a má specifickou roli. Například:

- Sprite (2D objekt): Zobrazuje obrázek na obrazovce.
- CollisionShape2D: Definuje tvar pro detekci kolizí.
- Camera2D: Umožňuje sledování pohybu kamery.
- Area2D: Používá se pro detekci interakcí s okolními objekty, například pro spuštění událostí, když hráč vstoupí do oblasti.
- atd.

Uzel může mít i poduzly, což umožňuje organizovat složité objekty, jako jsou animované postavy nebo dynamické objekty. Systém uzlů v Godotu je flexibilní a modulární, což zjednodušuje vytváření komplexních herních scén.

4 BACKEND

Backend 2D hry zahrnuje veškerou logiku a funkcionalitu, která podporuje herní mechaniky a interakce mezi objekty ve hře. Zahrnuje implementaci pohybu postav, detekci kolizí, umělou inteligenci pro nepřátele, správu dat jako je ukládání herního pokroku a generování úrovní. Backend je zodpovědný i za propojení s frontendem pomocí skriptů, které řídí interakce mezi hrou a herním prostředím. Dále může zahrnovat síťové funkce pro multiplayer, které zajišťují komunikaci mezi hráči. Celkově backend zajišťuje, že všechny herní mechaniky fungují správně a interakce s hrou jsou plynulé a realistické.

4.1 POHYB A ANIMACE CHARAKTERU HRÁČE

Pohyb charakteru hráče je řízen vstupy hráče (klávesy nebo ovladač) a zahrnuje akce jako běh, skákání a interakce s prostředím. Animace jsou spojeny s těmito pohyby, přičemž každá akce (běh, skok) má svou specifickou animaci, která se přehrává podle aktuálního pohybu postavy. V enginech jako Godot jsou pohyb a animace propojeny, aby plynule reagovaly na akce hráče.

4.1.1 Kode charakteru

4.2 PŘÍKLAD KÓDU

Kód 4.1: Ukázka kódu charakter$_p$*ixel*.*gd*

```
extends CharacterBody2D
class_name Player
const TARGET_FPS = 60
const ACCELERATION = 8.0
const MAX_SPEED = 64.0
const FRICTION = 10.0
const AIR_RESISTANCE = 1.0
const GRAVITY = 4.0
const JUMP_FORCE = 140.0
const MAX_FALL_SPEED = 500.0
```

```
var motion = Vector2.ZERO

@onready var sprite = $Sprite2D
@onready var animation_player = $AnimationPlayer

func _physics_process(delta):
	var x_input = Input.get_action_strength("ui_right") -
		Input.get_action_strength("ui_left")

	if x_input != 0:
		motion.x += x_input * ACCELERATION * delta *
			TARGET_FPS
		motion.x = clamp(motion.x, -MAX_SPEED, MAX_SPEED)
		sprite.flip_h = x_input < 0
		if is_on_floor():
			animation_player.play("Run")
	else:
		if is_on_floor():
			motion.x = lerp(motion.x, 0.0, FRICTION * delta)
			animation_player.play("Stand")
		else:
			motion.x = lerp(motion.x, 0.0, AIR_RESISTANCE * delta)

	motion.y += GRAVITY * delta * TARGET_FPS

	if motion.y > MAX_FALL_SPEED:
		motion.y = MAX_FALL_SPEED

	if is_on_floor():
		if Input.is_action_just_pressed("ui_up") or
			Input.is_action_just_pressed("ui_select"):
			motion.y = -JUMP_FORCE
			animation_player.play("Jump")
	else:
		if Input.is_action_just_released("ui_up") and
			motion.y < -JUMP_FORCE / 2:
			motion.y = -JUMP_FORCE / 2
		animation_player.play("Fall")

	velocity = motion
```

```
        move_and_slide()
```

4.2.1 Animovani pohybu

Pro animace jsem použil pole charakterů v různých pozicích pro vytvoření animace 'Stand', 'Fall', 'Jump' a 'Run'.

4.3 INTRO

Kód 4.2: Ukázka kódu pro přechod mezi scénami

```
extends Node2D

# Called when the node enters the scene tree for the first time.
func _ready() -> void:
    $AnimationPlayer.play("Fade in")
    await get_tree().create_timer(6).timeout
    $AnimationPlayer.play("Fade out")
    await get_tree().create_timer(3).timeout
    get_tree().change_scene_to_file("res://Scenes/dialog.tscn")
```

4.4 MENU

Základní uzivatelsé prostředí pro výběr možností

4.4.1 Settings

Stránka s nastavením hlasitosti zvuku, vypnutím zvuku a zmněnou rozlišení okna Aplikace.

Kód 4.3: Ukázka kódu pro ovládání zvuku a rozlišení

```
extends Control

func _on_volume_value_changed(value: float) -> void:
    AudioServer.set_bus_volume_db(0, value)
```

```
func _on_mute_toggled(toggled_on: bool) -> void:
    AudioServer.set_bus_mute(0, toggled_on)

func _on_resolutions_item_selected(index: int) -> void:
    match index:
        0:
            DisplayServer.window_set_size(Vector2i(1920, 1080))
        1:
            DisplayServer.window_set_size(Vector2i(1600, 900))
        2:
            DisplayServer.window_set_size(Vector2i(1280, 720))

func _on_back_menu_pressed() -> void:
    get_tree().change_scene_to_file("res://Scenes/menu.tscn")
```

4.4.2 Hlavní menu

Přemístí pomocí tlačítek uživatele do Tutorial Levelu, Settings nebo ukonci aplikaci pomoci "quit".

Kód 4.4: Ukázka kódu pro hlavní menu

```
extends Control

func _ready() -> void:
    $VBoxContainer/START.grab_focus()

func _on_start_pressed() -> void:
    get_tree().change_scene_to_file("res://Scenes/LevelOne.xml.tscn")

func _on_options_pressed() -> void:
    get_tree().change_scene_to_file("res://Scenes/options.tscn")

func _on_quit_pressed() -> void:
    get_tree().quit()
```

4.4.3 Úvodní dialog

Zobrazí uživateli příběh herní postavy ve formě dialog boxu

Kód 4.5: Ukázka kódu pro dialog v hře

```
extends RichTextLabel

var dialog = [
    Text dialogu postav
]

var page = 0
var timer : Timer


func _ready() -> void:
    set_process_input(true)
    text = ""  # Clear text to start fresh
    visible_characters = 0

    # Add a Timer node dynamically if not already in the scene
    timer = Timer.new()
    add_child(timer)
    timer.connect("timeout", Callable(self, "_on_timer_timeout"))
    timer.wait_time = 0.05  # Adjust the speed of character reveal
    timer.start()

    show_page()  # Show the first page

# Function to handle showing a new page
func show_page() -> void:
    text = dialog[page]
    visible_characters = 0
    timer.start()

func _input(event: InputEvent) -> void:
    if event is InputEventMouseButton and event.pressed:
        if visible_characters >= get_total_character_count():
            if page < dialog.size() - 1:
                page += 1
                show_page()
            else:
                print("End of dialog.")
                change_scene_to_menu()
        else:
```

```
            # Skip to show the full text instantly
            visible_characters = get_total_character_count()

func _on_timer_timeout() -> void:
    if visible_characters < get_total_character_count():
        visible_characters += 1
    else:
        timer.stop()  # Stop the timer when all characters are visible

# Function to change the scene
func change_scene_to_menu() -> void:
    get_tree().change_scene_to_file("res://Scenes/menu.tscn")
```

ZÁVĚR

V této práci jsem se zaměřil na vytvoření základního herního konceptu zahrnujícího mechaniku skákání a systém výběru úrovní. Tyto prvky představují klíčové komponenty pro zábavný a uživatelsky přívětivý herní zážitek. Implementace herní fyziky, plynulého pohybu a přehledného uživatelského rozhraní pro výběr úrovní byly hlavními výzvami, které se mi podařilo úspěšně překonat.

Mechanika skákání byla optimalizována tak, aby byla intuitivní a přístupná pro hráče různých úrovní dovedností. Systém výběru levelů umožňuje hráčům snadno navigovat mezi úrovněmi, což podporuje plynulý průběh hry.

MOŽNÁ VYLEPŠENÍ

- Rozšíření herních mechanik
- Vylepšení vizuální stránky
- Multiplayer a sociální prvky
- Oprava chyb(Bugu)
- Přidání hudby/zvuků do pozadí

LITERATURA

[1] OpenAI, ChatGPT, Jazykový model umělé inteligence. Dostupné z: https://openai.com/chatgpt, [cit. 2025-01-02].

[2] YouTube, Godot Engine - Complete Game Dev Tutorials, Playlist vytvořená uživatelem Godot Tutorials. Dostupné z: https://www.youtube.com/playlist?list=PL9FzW-m48fn1iR6WL4mjXtGi8P4TaPIAp, [cit. 2025-01-02].

[3] Godot Engine, Oficiální repozitář na GitHubu. Dostupné z: https://github.com/godotengine/godot, [cit. 2025-01-02].