SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

Dokumentace

Get out: Druids escape hra v Godot Engine

Jeroným Baron

Obor:	18-20-M/01 INFORMAČNÍ TECHNOLOGIE se zaměřením na počítačové sítě a programování
Třída:	IT4
Školní rok:	2024/2025

Poděkování

- *Rád bych vyjádřil své upřímné poděkování panu Mgr. Markovi Lučnému za cennou podporu a konzultace, které mi poskytoval před i během tvorby této závěrečné práce.*

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

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

V Opavě 09. 01. 2025

podpis autora práce

ABSTRAKT

Hlavním cílem bylo vytvoření hratelné hry s funkcionalitou, která zahrnuje pohyb nepřátel s využitím jednoduchého AI a pathfinding systémem v ručně navržených úrovních. Vývoj se soustředil na implementaci herních mechanik, jako je pohyb postav a interakce s prostředím a to s ohledem na plynulost a zábavnost hry. Během vývoje bylo nutné řešit různé technické výzvy, například optimalizaci herního Enginu a efektivní správu úrovní. Proces zahrnoval programování herní logiky, práci s moderními nástroji pro 3D modelování a texturování a také testování, které pomohlo vyladit detaily a zajistit správné fungování hry.

ABSTRACT

The main goal was to create a playable game with functionality, including enemy movement in hand-designed levels. The development focused on implementing game mechanics such as character movement and interaction with the environment, with an emphasis on gameplay fluidity and enjoyment. Throughout the development, various technical challenges had to be addressed, such as optimizing the game engine and efficiently managing levels. The process involved programming game logic, working with modern 3D modeling and texturing tools, and testing, which helped fine-tune the details and ensure the game worked as intended.

OBSAH

ÚVOD

Hry v dnešní době představují jednu z nejpopulárnějších forem zábavy, která spojuje technologické inovace s kreativním vyprávěním příběhů. Moderní hry sahají od jednoduchých mobilních aplikací až po komplexní tituly s otevřeným světem, které hráčům nabízejí stovky hodin zábavy. Díky vývoji technologií, jako je virtuální realita, ray tracing a cloudové hraní, se zážitek z her stává stále více pohlcujícím a přístupným. Herní průmysl také přináší silný sociální prvek, ať už prostřednictvím online multiplayerových her, nebo komunit na streamovacích platformách, kde hráči sdílejí své zážitky.

Cílem mého projektu bylo naučit se pracovat a používat Godot engine, seznámit se a porozumět jazyku GDScript. Zdokonalit se v práci a vytváření 3D modelů v Blendru a během toho vytvořit hru která bude příjemně hratelná a dokáže zabavit. Mým stanoveným cílem bylo vytvořit hratelnou hru, ve které budou nepřátelé s funkčním AI a pathfinding systémem ve vlastních úrovních.

Tento dokument může posloužit jako inspirace a zdroj informací pro začínající vývojáře her, kteří se chtějí naučit pracovat s Godot Enginem, vytvářet vlastní 3D projekty nebo pochopit základní principy vývoje herních mechanik.

1 ZÁMĚR A CÍLE PROJEKTU

1.1 Cíle projektu

Hlavním cílem projektu bylo vytvoření plně funkční 3D hry žánru dungeon crawler, která by byla příjemně hratelná a dokázala zaujmout hráče. Při práci na projektu jsem se zaměřil na několik klíčových oblastí: seznámení a práce s herním enginem Godot, osvojení programovacího jazyka GDScript a zdokonalení dovedností při tvorbě 3D modelů v softwaru Blender. Dalším důležitým cílem byla implementace funkčního AI systému pro nepřátele, který zahrnuje pathfinding systém pro pohyb nepřátel v prostředí.

1.2 Herní prvky

Hra byla tvořena pro žánr dungeon crawler znamenaje, že cílem hry je procházet temnými katakombami a nasbírat co nejvíce zlata a esencí ideálně bez toho, aby hráč upozornil jakéhokoli nepřítele. Hlavní mechanikou je míra nebezpečí, která varuje hráče. Protože čím déle bude hráč uvnitř, tím více věcí může najít ale i tím nebezpečnější, rychlejší a pozornější se stávají nepřátelé.

Druhou mechanikou je využití balíčku karet, který nabízí různé bonusy, které pomáhají hráči při postupu hlouběji do katakomb.

Karty lze zakoupit za získané esence v obchodě před vstupem do katakomb stejně jako pasivní vylepšení které lze koupit za mince.

1.3 Get out: Druids escape v budoucnu

Hru hodlám dále vyvíjet i po odevzdání. V budoucích verzích by mělo být podstatně více úrovní, silnější nepřátelé a rozvinutější mechanika balíčku karet, která by měla dávat unikátní dočasné bonusy na zjednodušení procházení katakomb. Dále pasivní předměty které budou zase zvyšovat základní atributy hráče a jeho schopnost procházet danými úrovněmi.

2 VYUŽITÉ TECHNOLOGIE

2.1 Seznam technologii

Během vývoje hry byly použity následující technologie a nástroje:

- **Godot Engine**: Hlavní herní Engine pro tvorbu hry. Godot nabízí robustní systém scén, integrované nástroje pro tvorbu 2D a 3D her a podporu pro GDScript, VisualScript i programování v C#.
- **GDScript**: Programovací jazyk specifický pro Godot Engine. Jedná se o vysokoúrovňový orientovaný jazyk inspirovaný Pythonem, který usnadňuje rychlé prototypování a tvorbu herních mechanik.
- **Blender**: Software pro tvorbu 3D modelů, animací a texturování. Blender byl využit k vytváření herních objektů, postav a dalšího obsahu.
- **OpenGL**: Renderovací framework, který Godot využívá pro vykreslování 2D a 3D grafiky.

2.2 Výhody práce v Godot Enginu

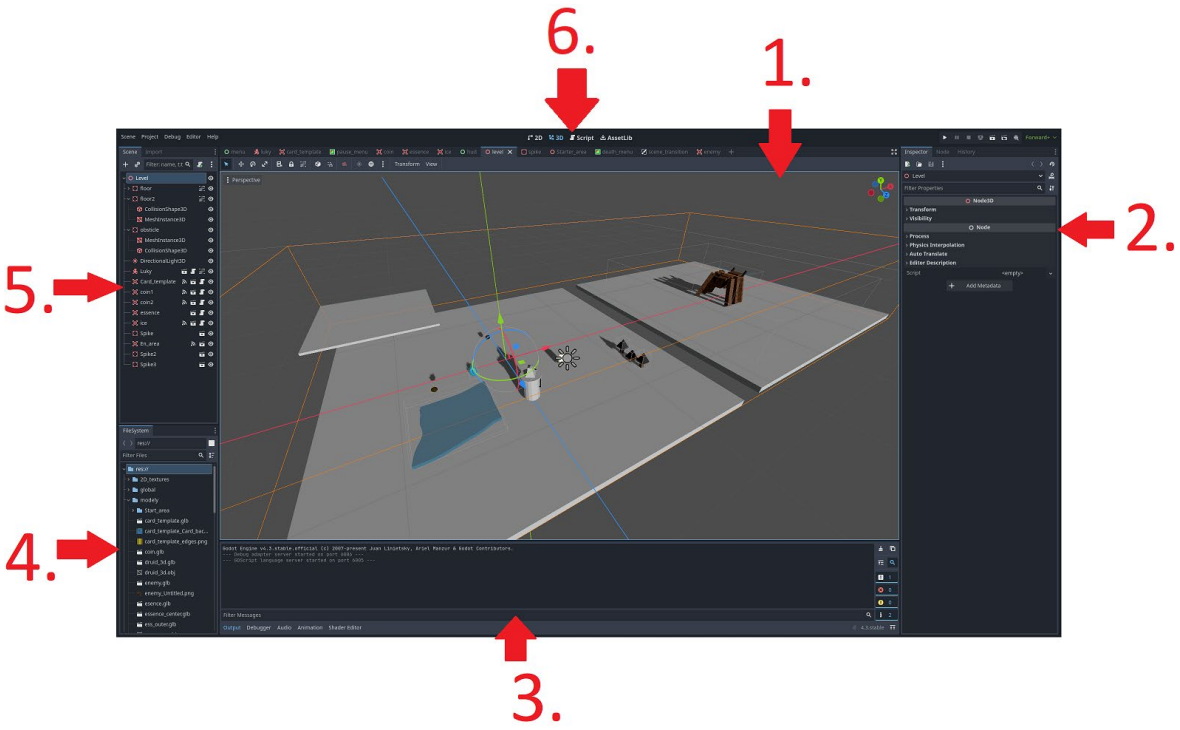
1. **Otevřený zdrojový kód**: Godot je open-source, což znamená, že je zdarma k dispozici a lze ho modifikovat podle potřeb projektu.
2. **Jednoduché rozhraní**: Intuitivní a uživatelsky přívětivé rozhraní umožňuje rychlý začátek i pro začátečníky.
3. **Podpora pro 2D i 3D**: Godot poskytuje vysoce optimalizované nástroje pro vývoj 2D i 3D her, což umožňuje flexibilitu při tvorbě různorodých projektů.
4. **Integrovaný skriptovací jazyk:** GDScript je snadný na naučení a velmi efektivní pro rychlý vývoj prototypů.
5. **Lehká instalace a nízké nároky**: Godot je nenáročný na hardware, což umožňuje vývoj i na starších zařízeních.
6. **Silná komunita a dokumentace**: Komunita kolem Godotu je aktivní a poskytuje řadu tutoriálů, řešení problémů a doplňkových pluginů.

2.3 Nevýhody práce v Godot Enginu

1. **Méně rozšířené oproti jiným engineům**: Ve srovnání s Unity nebo Unreal Enginem nemá Godot tak rozsáhlou podporu třetích stran a integrací.
2. **Nižší výkon v komplexních 3D projektech**: I když se Godot zlepšuje v oblasti 3D, není tak optimalizovaný jako konkurence při tvorbě velmi rozsáhlých nebo graficky náročných projektů.
3. **Menší míra předpřipraveného obsahu**: Godot obsahuje méně vestavěných šablon a assetů, což může vyžadovat více práce při vývoji od nuly.
4. **Specifický jazyk GDScript**: I když je GDScript snadný na použití, nemusí vyhovovat všem vývojářům, kteří jsou zvyklí na univerzálnější jazyky, jako je C++ nebo C#.
5. **Menší zázemí pro komerční projekty**: Pro velké studio může být Godot omezenější v oblasti marketingové podpory a profesionálních služeb.

Godot Engine je ideální volbou pro jednotlivce a menší týmy, kteří chtějí vytvořit kreativní projekty s minimálními náklady a učit se moderním postupům herního vývoje.

3 DOPORUČENÁ ČASOVÁ OSA VÝVOJE HER



Obrázek 1: Godot editoru

1. Pracovní plocha (3D viewport):

Hlavní prostor, kde se upravuje scéna. Umožňuje zobrazit a manipulovat s objekty ve 3D nebo 2D prostoru pomocí nástrojů jako pohyb, rotace, nebo škálování. Nebo psaní scriptů ve skriptovacím okně.

2. Inspektor:

Tento panel umožňuje uživateli upravovat vlastnosti vybraného objektu. Obsahuje možnosti jako Transform, Visibility a další nastavení specifické pro uzel nebo objekt, který je aktuálně označen.

3. Konzole / Výstupy:

Oblast pro zobrazení výstupů, ladění nebo chybových hlášení během práce v editoru či při spuštění aplikace.

4. Správce souborů (Filesystem):

Zobrazuje všechny soubory a assety projektu, které jsou k dispozici. Slouží k rychlému přístupu k modelům, texturám, skriptům a dalším souborům.

5. Seznam uzlů (Scene tree):

Stromová struktura všech objektů (uzlů) ve scéně. Umožňuje vybírat, přidávat, odstraňovat a organizovat objekty ve scéně.

6. Horní panel (hlavní nabídka):

Zahrnuje hlavní nabídky jako 3D, 2D, Script a další nástroje pro přepínání pohledů a funkcí editoru. Slouží k přístupu k obecným funkcím, přepínání mezi různými módy a nástrojům.

3.1 Začátek projektu

Začátek projektu v Godotu je o převodu nápadu a své vize do konkrétní podoby. S flexibilními nástroji tohoto enginu je možné rychle prototypovat a testovat různé koncepty. To nám umožní rozvíjet kreativní myšlenky do funkčního výsledku v celkem rozumném čase. V téhle kapitole ukážu doporučený postup projektu na příkladu této hry.

3.2 Tvorba hráče

V tomhle příkladu dělám 3D hru a tím pádem si vytvoříme Godot projekt pro graficky náročnější projekty. V první scéně je třeba vytvořit hráčskou postavu tak, že přidám objekt characterBody3D. Tomuto objektu se musí jako potomci přidat dva objekty a to MeshInstance3D, aby náš hráč měl viditelný tvar. Druhým je CollisionShape3D aby náš hráč mohl interagovat s prostředím.

Dalším klíčovým objektem je kamera, takže přidáme objekt Camera3D.

Pokud chceme mít hru z první osoby jako je můj projekt, tak dáme kameru jakožto potomka našeho hráče.

3.3 GDScript

GDScript je skriptovací jazyk specificky navržený pro herní engine Godot. Svou syntaxí se podobá Pythonu, což z něj činí snadno srozumitelný a čitelný jazyk, zejména pro začátečníky. GDScript je přímo integrovaný do Godotu, což znamená, že umožňuje efektivní práci s herními objekty a scénami bez potřeby externích knihoven. Mezi hlavní výhody patří jednoduchost, rychlý vývojový cyklus díky vestavěné podpoře v editoru a optimalizace pro

výkon při práci v herním prostředí. Díky tomu je GDScript ideální pro rychlé prototypování i plnohodnotný vývoj her.

3.3.1 Příklad jednoduchého GDScriptu

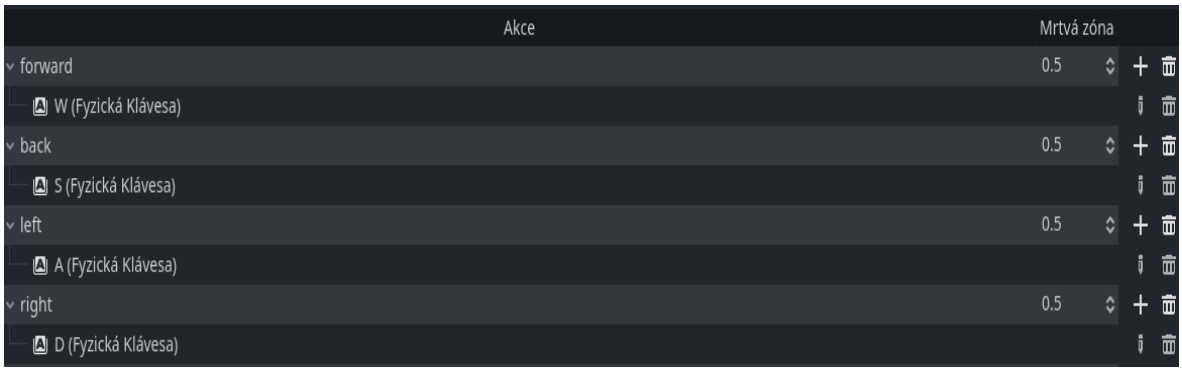
Toto je jednoduchý příklad GDSriptu pro levitaci mince:

```
extends Area3D

#Deklarace proměnných
const ROT_SPEED = 100
var move_speed = 0.005
var move_y = 0
var add = false

# Funkce   ready se volá pouze jednou, a to tehdy když se daný objekt poprvé načte v dané scéně
func _ready() -> void:
	pass

# Funkce   process se volá každý frame. Proměnná delta, kterou příjme jako parametr ‘delta‘ Je čas mezi dvěma herními frame
func _process(delta: float) -> void:
# Tenhle for loop zjišťuje, zda uvnitř kolize mince se nachází hráč a pokud ano tak jí smaže ze scény a připočte hráči jednu minci
	for body in get_overlapping_bodies():
		if body.is_in_group("player"):
			queue_free()
			Global_Vars.coins = Global_Vars.coins + 1

	#Rotace mince okolo své osy počítané pomocí času delta
	rotate_y(deg_to_rad(ROT_SPEED * delta))

	#Zajištění pohybu mince v daném rozmezí vytvářející pohyb nahoru a dolů
	if move_y <= -0.35:
		add = true
	if move_y >= 0.35:
```

```
            add = false
        if add == true:
            move_y += move_speed
        if add == false:
            move_y -= move_speed


        position.y += move_y * delta * 0.5
```

3.4 Mapování a programování pohybu

Další částí je zprovoznit pohyb našeho hráče tak, že si vytvoříme pro příklad soubor: ‘Hráč.gd‘, ve kterém potřebujeme naprogramovat pohyb. Nejdřív si v nastavení projektu definujeme hráčské vstupy v nastavení projektu a následně mapování vstupů jako v příkladu:



Obrázek 2: Mapování vstupů

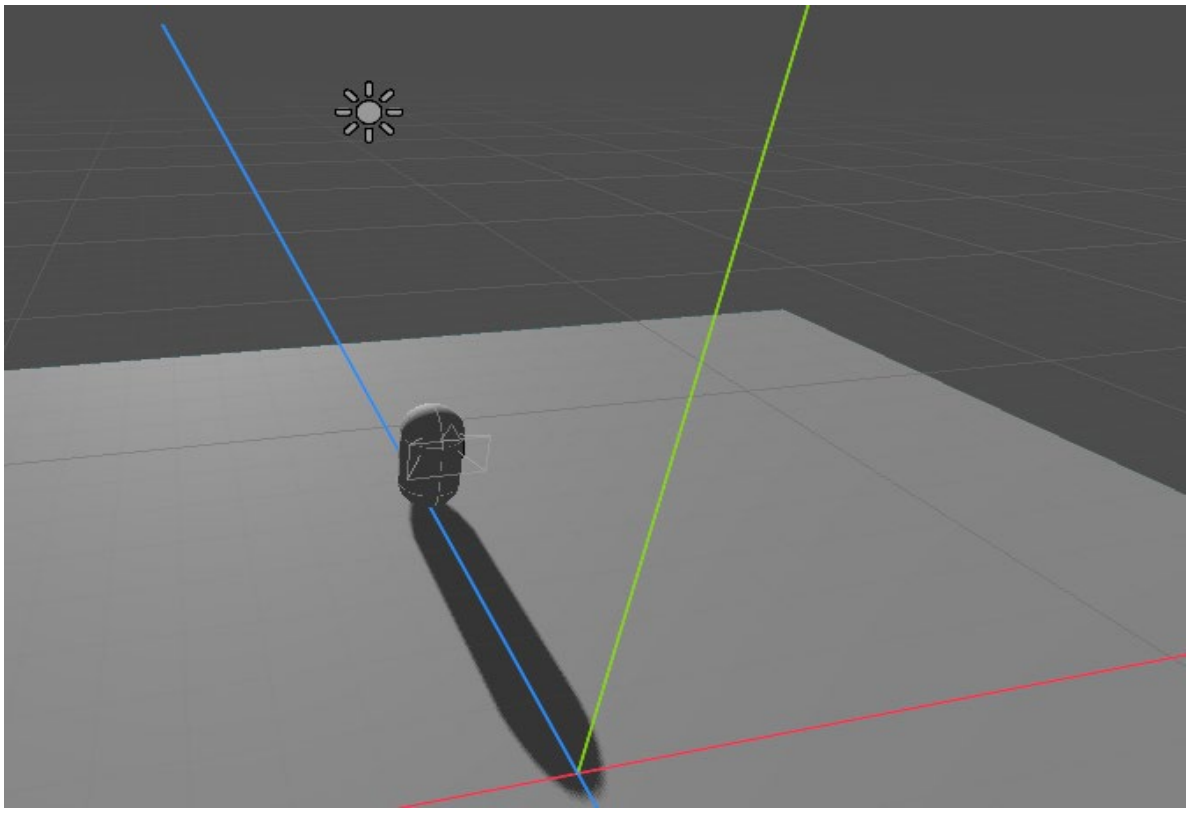
Potom si připojíme náš hráčský script k hráčskému objeku a přidáme kód pro hráčské ovládání závislé na tom jak jste si pojmenovali namapované vstupy:

```
var input_dir = Input.get_vector("left", "right", "forward", "back")
var direction = (transform.basis * Vector3(input_dir.x, 0, input_dir.y)).normalized()
        if direction:
            velocity.x = direction.x * curent_speed
            velocity.z = direction.z * curent_speed
        else:
            velocity.x = move_toward(velocity.x, 0, curent_speed)
            velocity.z = move_toward(velocity.z, 0, curent_speed)
```

3.5 Tvorba prostředí

Přidáme si nějaké základní prostředí tak, že objekt StaticBody3D s dvěma potomky MeshInstance3D a CollisionShape3D. Poté přidáme directionalLight3D aby šlo něco vidět. A už se

může spustit projekt a máme hráče s funkčním ovládáním, což je největší základ pro rozvíjení svého projektu.



Obrázek 3: Godot prostředí

3.6 Hierarchie scén

Hierarchické stavění scén v Godot enginu je klíčovým konceptem, který usnadňuje organizaci a správu herních prvků. Každá scéna je tvořena stromovou strukturou uzlů, kde každý uzel má svůj specifický typ a funkci. Například zobrazení, fyziku nebo logiku. Díky hierarchii mohou uzly dědit vlastnosti od svých rodičů, což zjednodušuje opakované používání komponentů. Tato organizace umožňuje, aby změny provedené na rodičovském uzlu automaticky ovlivnily všechny jeho potomky, například transformace, viditelnost nebo aktivace.

Scény lze snadno vnořovat, což umožňuje vytvoření modulárních a znovupoužitelných částí hry, jako jsou postavy, prostředí nebo UI prvky. Tento přístup výrazně zvyšuje efektivitu vývoje a přehlednost projektu.

3.6.1 Příklady Hierarchií

Obrázek 4: Příklad hierarchie

Hierarchie vypadá jako na obrázku výše a ukazuje nám které objekty jsou rodičovské pro které a taky atributy daných objektů.

Př. MeshInstance3D je potomkem objektu StaticBody s názvem obsticle. Přičemž objekt obsticle je potomkem objektu Node3D s názvem level.

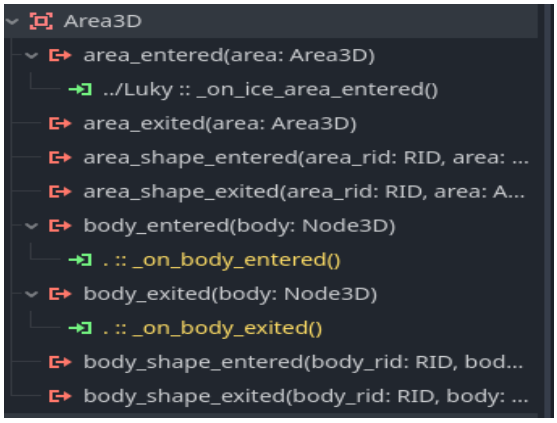
1. ikona signálu – Tato ikona značí, že uzel má připojené nebo vysílané signály.
2. ikona instance třídy – Tento symbol ukazuje, že uzel je odvozený z externí scény.
3. ikona scriptu – Tato ikona naznačuje, že je k uzlu připojen skript (kód).
4. ikona viditelnosti – Tato ikona značí, zda je uzel viditelný nebo skrytý.

3.7 Signály

Signály v Godotu umožňují uzlům komunikovat pomocí událostí jako je kliknutí nebo změna stavu, aniž by byly pevně propojeny. Díky tomu je kód modulární, přehledný a snadno udržovatelný. Umožňují například upozornit rodičovský uzel na akci potomka, aniž by mezi nimi byla přímá závislost, což je klíčové pro událostmi řízené systémy.

3.7.1 Příklad signálů

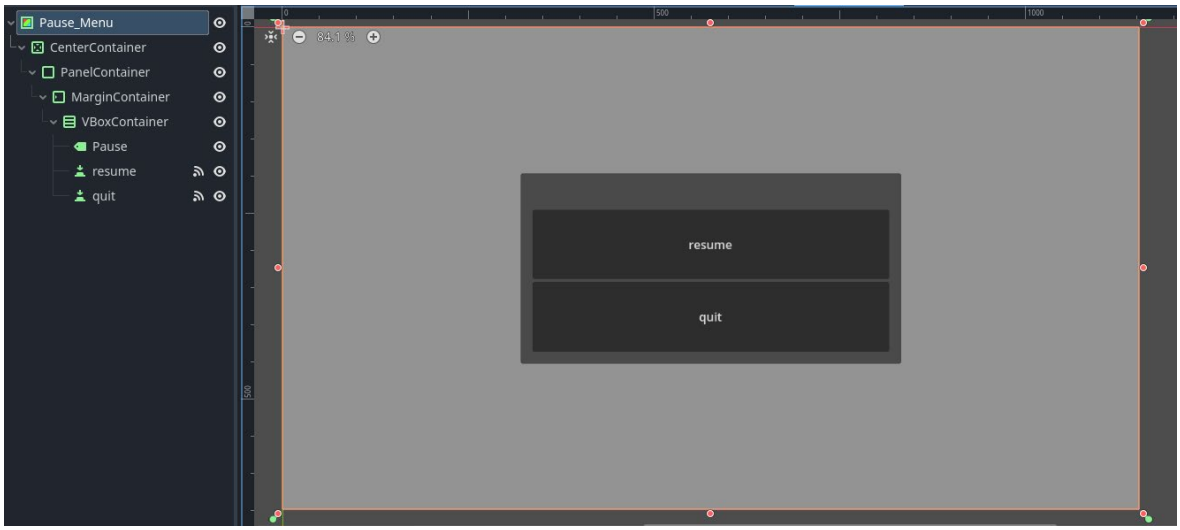
Příklad vestavěných signálů pro objekt Area3D



Obrázek 5: Příklad signálů

3.8 Menu pozastavení

Dalším logickým krokem bylo vytvoření menu pozastavení, aby hráč mohl hru pozastavit a nebyl nucen hrát bez přestávek. Prvně jsem vytvořil scénu s průhledným pozadím a dvěma tlačítky ’resume’ pro pokračování a ’quit’ pro ukončení hry.



Obrázek 6: Scéna pozastavení

Připojení scriptu pro dané menu:

```
extends ColorRect

@onready var luky = $"../../"

func _on_resume_pressed() -> void:
        luky.pauseMenu()
        Input.set_mouse_mode(Input.MOUSE_MODE_CAPTURED)
```

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

Do scriptu hráče dopsat funkci PauseMenu():

```
func pauseMenu():
	if paused:
		pause_menu.hide()
		Engine.time_scale = 1
	else:
		pause_menu.show()
		Engine.time_scale = 0

	paused = !paused
```

3.9 HUD (heads up display)

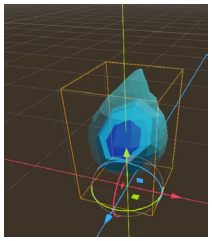
Dalším krokem bylo vytvořit HUD, který zobrazuje hráči důležité informace jako je zdraví, skóre nebo čas přímo na obrazovce. Umožňuje rychlou orientaci ve hře, aniž by rušil zážitek. Dobrý HUD je přehledný a poskytuje hráči potřebné údaje bez zbytečného zdržování.

Obrázek 7: Scéna průhledového displeje

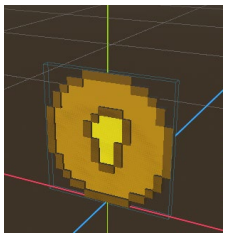
Samozřejmě tím že máme různé velikosti monitorů, musíme zapracovat i na responsivitě všech spritů na obrazovce HUD displeje.

3.10 Sběratelné předměty

Další bylo vytvoření herní měny, naprogramování a vytvoření sebratelného objektu. To bylo dokázáno vytvořením objektu Area3D. V momentu, kdy se hráč ocitne v téhle oblasti, se předmět smaže ze scény a přičte do globální proměnné hráčových mincí. Pro esence je stejný proces.



Obrázek 8 : Objekt esence



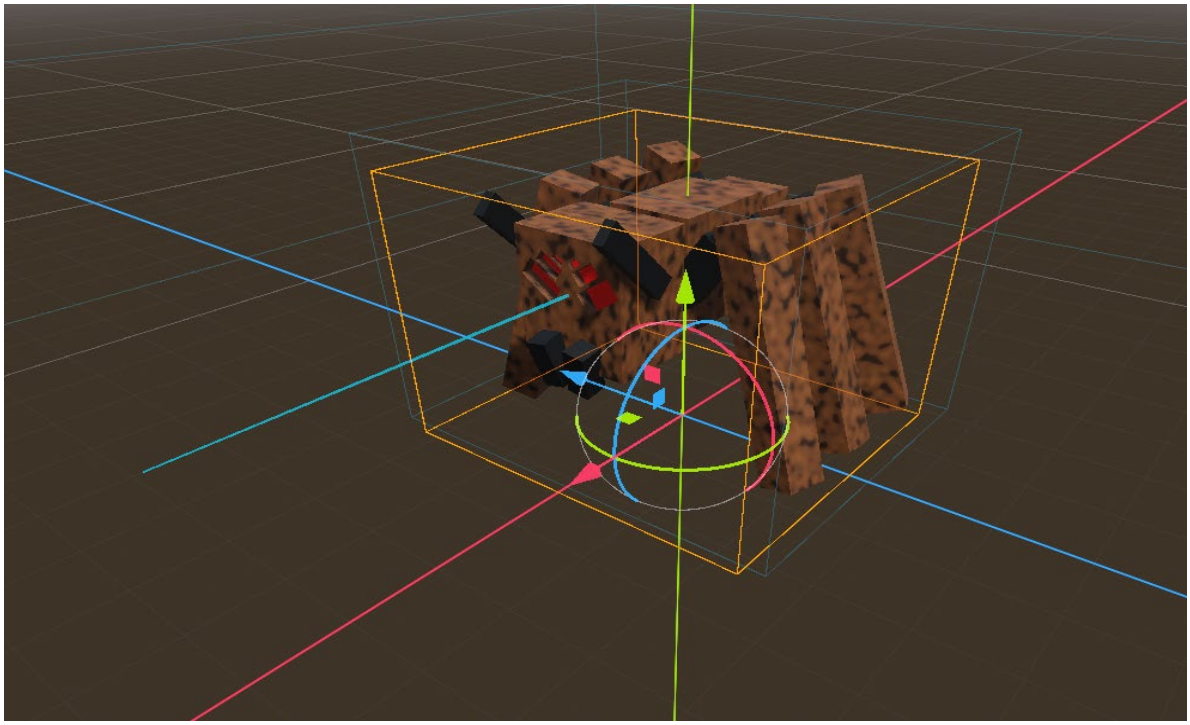
Obrázek 9: Objekt mince

3.11 Nepřátelé

Posledním krokem pro základní hru a také pro tenhle doporučený postup jsou nepřátelé, které lze vytvořit tak, že vytvoříme stejně jako u hráče characterBody3D s potomky CollisionShape3D a MeshInstance3D. Ale pro nepřítele ještě vložíme NavigationAgent3D a do oblasti ve které chcete, aby se nepřítel pohyboval, vložíte NavigationRegion3D. Když následně vypečete integrovaný pathfinding systém v Godotu, máte vytvořeného jednoduchého nepřítele. Když k němu připojíte tenhle script, bude fungovat.

```
extends CharacterBody3D

@onready var nav = $NavigationAgent3D
const SPEED = 5.0
var I_frame = 0.8

func _physics_process(delta: float) -> void:
	# Add the gravity.
	if not is_on_floor():
		velocity += get_gravity() * delta
```

```
func _process(delta: float) -> void:
	var next_location = nav.get_next_path_position()
	var current_location = global_transform.origin
	var new_velocity = (next_location - current_location).normalized() * SPEED
	hit()

	velocity = velocity.move_toward(new_velocity, delta)
	move_and_slide()

func hit():
	for index in range(get_slide_collision_count()):
		var collision = get_slide_collision(index)
		if collision.get_collider().is_in_group("Player"):
			var coll_p = collision.get_collider()
			if coll_p.has_method("got_hit"):
				coll_p.got_hit(40)

func target_position(target):
	nav.target_position = (target)
```



Obrázek 10: Objekt nepřítele

Já jsem však ve svém projektu přidal svému nepříteli tři raycasty:

3.11.1 Raycasting

Raycast je technika v počítačové grafice a programování, která simuluje cestu paprsku z určitého bodu v prostoru za účelem detekce kolizí nebo interakce s objekty. Používá se například v herním vývoji k určení, zda paprsek zasáhne objekt, nebo při renderování k simulaci světelných efektů. Raycasting je rychlejší a jednodušší než složitější metody, jako je ray tracing.

A vytvořil jsem jednoduchou neuronovou síť (AI). Ta se učí pomocí těchto raycastů a souřadnic objektů v dané oblasti které dostává od NavigationAgent3D. Pomocí těchto vstupů se učí vyhýbat objektům a následování hráče.

ZÁVĚR

Cílem této práce bylo navrhnout a vytvořit 3D dungeon crawler hru pomocí Godot Enginu, která demonstruje klíčové principy vývoje her, jako je hierarchická správa scén, interakce mezi herními objekty a využití signálů. Výsledkem je plně funkční prototyp, který obsahuje základní herní mechaniky, jako je pohyb hráče, sbírání předmětů a interakce s nepřáteli.

Toto řešení má praktické uplatnění jako výukový materiál pro začínající vývojáře nebo základ pro další rozvoj herního projektu. Díky modularitě a přehledné architektuře lze hru snadno rozšiřovat o nové prvky, jako jsou složitější nepřátelské AI, propracované bojové mechaniky nebo systémy generování náhodných dungeonů.

Do budoucna by bylo možné zaměřit se na optimalizaci výkonu pro větší a komplexnější dungeony, integraci pokročilých grafických efektů nebo přidání multiplayerových funkcí. Tímto způsobem může projekt sloužit nejen jako základ pro další vývoj, ale také jako inspirace pro tvorbu podobných her v Godot enginu.

https://github.com/yrej/Final-Project

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] *Godot fórum*

Dostupné z: https://godotforums.org/

[2] *Oficiální dokumentace godot enginu*

Dostupné z: https://docs.godotengine.org/en/stable/

[3] *Lukky, Godot 4.X : Ultimate First Person Controller Tutorial (2023)*

Dostupné z: https://www.youtube.com/watch?v=xIKErMgJ1Yk

[4] *Lukky, Godot 4.X : Ultimate First Person Controller Tutorial Part 2(2023)*

Dostupné z: https://www.youtube.com/watch?v=WF7d21zOD0M&t