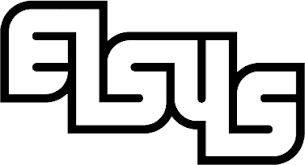
* 1. **ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ**

**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ДИПЛОМНА РАБОТА**

**по професия код 481020 „Системен програмист“**

**специалност код 4810201**  **„Системно програмиране“**

Тема: Ходово-ролева игра

Дипломант: Дипломен ръководител:

*Нам Хоанг Чиеу Борис Димитров*

СОФИЯ

2 0 2 5

**TЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ -СОФИЯ**

Дата на заданието: 28.10.2024 г. Утвърждавам:..............................

Дата на предаване: 28.01.2025 г. /проф. д-р инж. П. Якимов/

**ЗАДАНИЕ**

**за дипломна работа**

**ДЪРЖАВЕН ИЗПИТ ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ**

**по професия код 481020**  **„Системен програмист“**

**специалност код 4810201**  **„Системно програмиране“**

на ученика Нам Хоанг Чиеу от 12 б клас

1. Тема: Ходово-ролева игра
2. Изисквания:

2.1 Имплементиране на ходове за действията на героите

2.2 Да се реализират различни видове атаки на героите и опонентите

2.3 Да се реализира запазване на прогреса на играта при излизане от нея

2.4 Имплементация на 2 или повече опоненти

2.5 Да се имплементират 2 или повече герои

1. Съдържание:
   1. Теоретична част
   2. Практическа част
   3. Приложение

Дипломант :...........................................

/ Нам Хоанг Чиеу /

Ръководител:..........................................

/ Борис Димитров /

ВРИД Директор:...............................................

/ ст. пр. д-р Веселка Христова

# УВОД

**Днес, индустрията на видеоигрите е изправена пред нови предизвикателства и възможности. Според данни от 2024 година, глобалният пазар на видеоигри се очаква да нарасне с 2.1% на годишна база, достигайки стойност от 187.7 милиарда долара. Този растеж е по-бавен в сравнение с предходни години, отчасти поради намалени продажби на конзоли и промени в потребителските навици след пандемията от COVID-19. Освен това, внедряването на технологии с изкуствен интелект (AI) променя начина, по който се разработват и играят видеоигрите. AI позволява създаването на по-реалистични и адаптивни игрови среди, което подобрява потребителското изживяване и отваря нови възможности за развитие на индустрията.  
  
Избрах дипломната ми работа да бъде игра защото от малък имам интерес към видео игрите. Наближавайки времето да избирам каква кариера да поема реших да опитам с гейм девелопмент. Избрах този жанр защото изглеждат интересни и не много трудни. Целта на този проект е да е да се реализира малка ходова игра.**

# 

# Първа Глава

# Преглед на подобни продукти и на известни развойни среди

# 1.1 Преглед на на подобни ходово-ролеви игри

## 1.1.1 Angry birds epic

**Angry Birds Epic беше безплатна ролева видео игра, която е деветата част от поредицата Angry Birds, разработена от Chimera Entertainment и издадена от Rovio Entertainment. Играта беше обявена на 12 март 2014 година и включва стратегически битки на ходове и система за зъздаване. Играта беше премахната от App Store и Google Play през 2019 година. Както е типично за RPG игрите, всяка птица принадлежи към определен клас: червената птица е рицар, жълтата е магьосник, бялата е лечител и т.н., и според това действат в битка. Червената птица атакува врага със своето оръжие, жълтата използва магия, за да призовава мълнии върху противниците, а бялата атакува и едновременно с това лекува другите птички с всяка атака.**

****

**фиг. 1.1 Снимка на Angry Birds Epic по време а на битка**

## 1.1.2 Honkai Star Rail

**Honkai: Star Rail (HSR) е безплатна ролева видеоигра с елементи на гача (gacha), разработена и публикувана от miHoYo.**

**Honkai: Star Rail следва стила на класически японски ролеви игри, където играчите създават отбор от персонажи и управляват група от до четирима в бой на ходов.. Освен различни статистики, които влияят на силата на героите, всеки от тях има боен тип, определящ вида щети, които нанася, и клас, определящ ролята му в битка.**

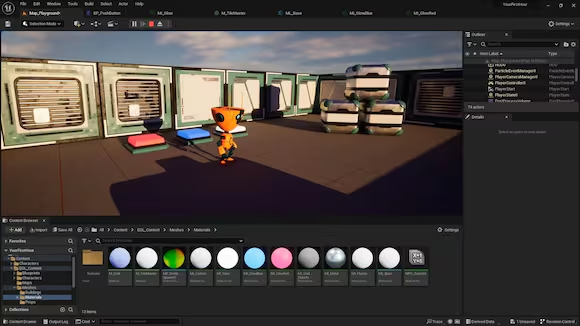


**фиг. 1.2 Снимка на Honkai Star Rail по време а на битка**

# 1.2 Технологии за създаване на игра

## 1.2.1 Unreal

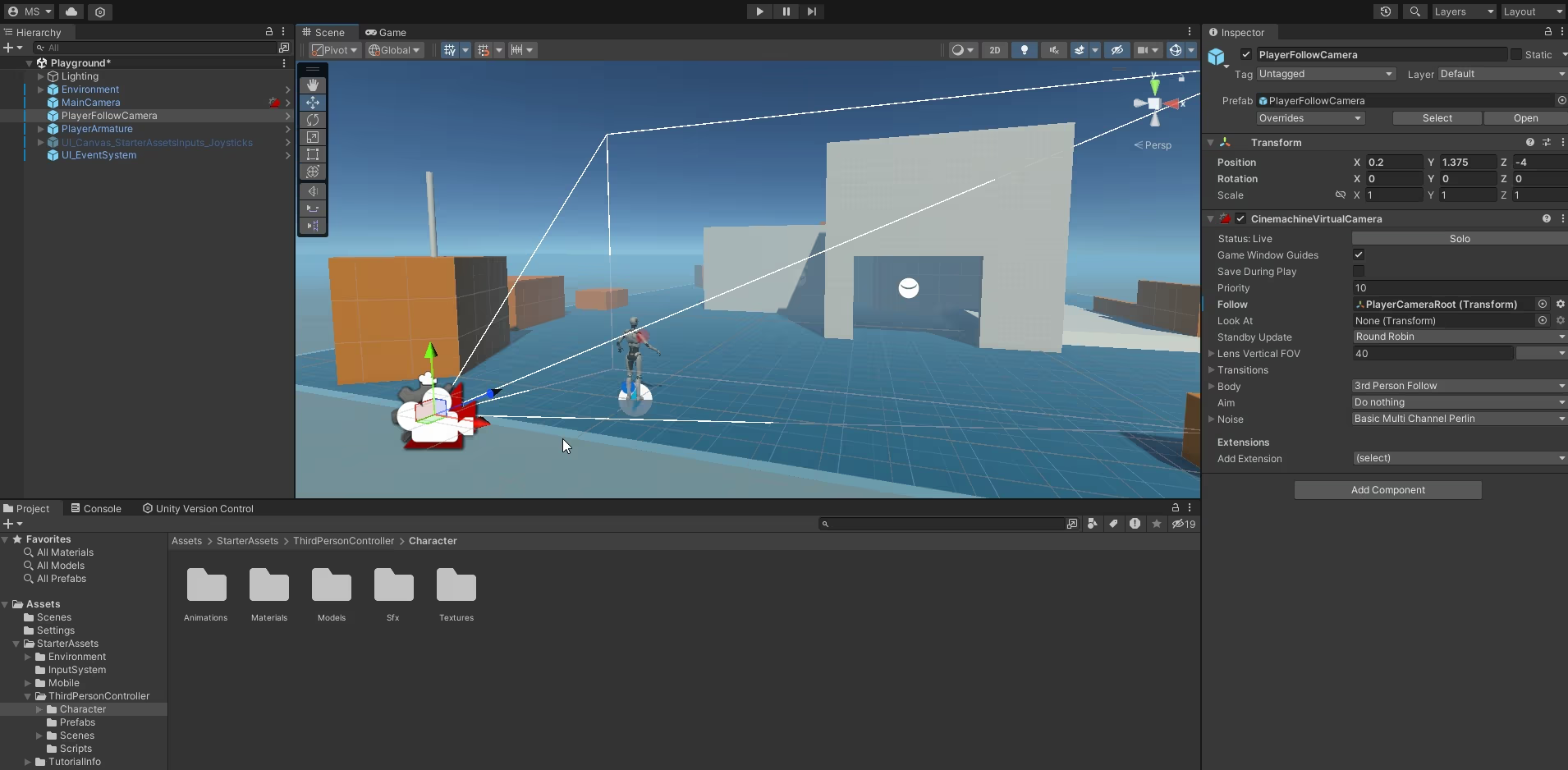
**Мощена и професионална среда, известен с висококачествена 3D графика, но поддържа и 2D проекти. Използва се за AAA игри и сложни визуални ефекти. Включва системата Blueprints за визуално програмиране, което улеснява създаването на логика без писане на код. Подходящ за амбициозни 2D игри с богати анимации и динамични светлини, но може да е прекалено сложен за малки проекти.**

****

**фиг. 1.3 Потребителски интерфейс на Unreal**

## 1.2.2 Unity

**Универсална среда, популярен сред инди разработчици и студия. Има отлични вградени инструменти за 2D разработка (Tilemap, 2D физика) и голяма библиотека от асети и разширения. Програмиране на C#, което го прави достъпен за начинаещи. Подходящ за ходови игри с баланс между лекота и функционалност, но изисква внимание към архитектурата на кода.**

****

**фиг. 1.4 Потребителски интерфейс на Unity**

## 1.2.3 Godot

**Лек, безплатен двигател с отворен код, създаден специално за 2D игри. Използва node-based система за организиране на обекти и GDScript (специален скриптов език) за бързо прототипиране. Идеален за малки студия или хобисти, търсещи гъвкавост и минимални разходи. Ограниченията му в 3D и по-малката общност са основните му слабости, но за чисти 2D проекти е изключително ефективен.**

**  
 фиг. 1.5 Потребителски интерфейс на Godot**

# Втора глава

# Функционални изисквания и аргументиран избор на развойна среда

# 2.1 Функционални изисквания

**2.1.1 Имплементация на ходове и действия на геройте чрез избора на вид атака и опонент**

**2.1.2 Нормална атака с по ниска сила и атака със средна сила, но с шанс за пропуск**

**2.1.3 При излизане на играта да се запази прогреса на битката**

**2.1.4 2 или повече опоненти , които имат различни атаки, атрибути и визия**

**2.1.5 2 или повече герои, които имат различни атаки, атрибути и визия**

# 2.2 Аргументация за избора на развойната среда и технология

**Защо Godot ?**

**Неговият интуитивен интерфейс и системата с "нодове" (визуални елементи) правят проектирането на нива, герои и битки лесно дори за тези без програмен опит. GDScript — специализираният език на Godot, подобен на Python — е прост за научаване и позволява бързо прототипиране на основни механики като ходови битки.**

**За разлика от Unity и Unreal, които изискват по-задълбочени познания по C# или Blueprints, Godot е създаден специално за 2D проекти. Вградените инструменти като Tilemap редактор и 2D анимационна система спестяват часове рутинна работа. Например, можеш да проектираш карта за битка с няколко клика, да анимираш герои с drag-and-drop функционалност и да импортираш безплатни асети от онлайн хранилища като itch.io.**

**Докато Unity предлага голяма общност и асети, а Unreal впечатлява с графика, тези двигатели са прекалено сложни за малки проекти.**

**Вградената разработна среда (IDE) на Godot е идеално решение за създаване на игри, особено за начинаещи или за проекти като 2D ходова RPG. Тя предлага пълен набор от инструменти в един интегриран интерфейс, което премахва нуждата от външни програми или сложни настройки. Node-базираният дизайн е в основата на Godot и прави създаването на логика интуитивно. Всеки елемент в играта — герой, бутон, враг — е "нод", който може да се влачи, настройва и свързва с други нодове. Така например можеш да построиш система за ходови битки, като свържеш нод за менюто с избор на действия, нод за анимациите на атаките и нод за управление на ходовете. Всичко това се визуализира в реално време, а отделните компоненти могат да се запазват като "сцени" (напр. "Битка", "Герой") и да се преизползват лесно. GDScript, специализираният език на Godot, е проектиран да бъде прост и четим, подобно на Python.**

# Трета глава

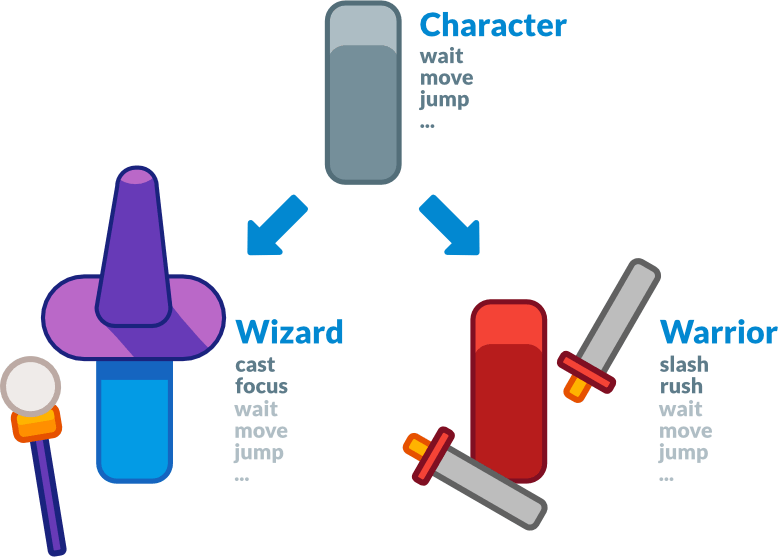
# Философски дизайн Godot

# Реализация на проекта

# 

# 3.1 Godot наследяване

Godot приема обектно-ориентиран дизайн в основата си чрез гъвкавата си система за сцени и йерархията на Node-ове. Той се опитва да се отклони от стриктните програмни модели, за да предложи интуитивен начин за структуриране на играта ви. Освен това, можете да наследявате от всяка сцена. Сцената в Godot може да бъде оръжие, герой, предмет, врата, ниво, част от ниво. Тя работи като клас в чист код, с това, че сме свободни да я проектирате чрез редактора, само с код или чрез комбинация от двете.



фиг. 3.1 пример

## 3.1.1 Наследяване на Node-ове

Godot предлага много различни видове обекти наречени Node, всеки със своя специфична цел. Node-овете са част от дърво и винаги наследяват от своите родители до класа Node. Въпреки, че енджинът включва някои нодове, като collision shape, които родителския Node physics body ще използва, повечето нодове работят независимо един от друг.



фиг 3.2 Деца на Node

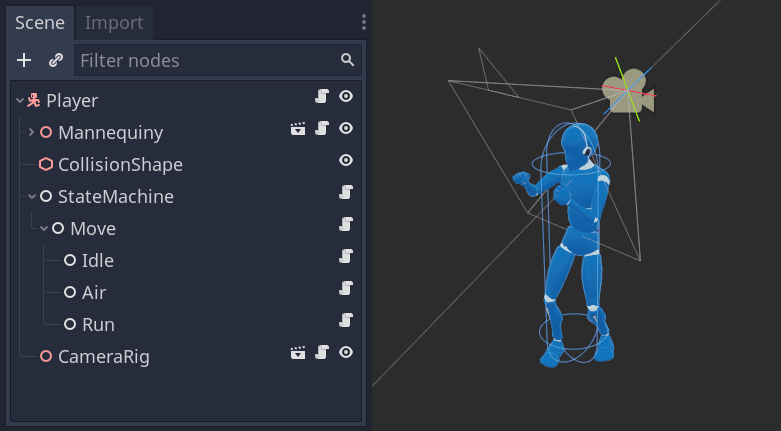
С други думи, Node-овете на Godot не работят като компоненти в някои други игрови енджини.

Sprite2D е Node2D, CanvasItem и Node. Той притежава всички свойства и характеристики на трите си родителски класа, като трансформации или възможността да чертае персонализирани форми и да рендерира с персонализиран шейдър.

## 3.1.2 Scenes

Когато организираме Node в дърво, както при нашия герой, тази конструкция наричаме Scene. След като бъде запазена, Scene функционират като нови типове Node-ове в редактора, където можем да ги добавим като дете на съществуващ нод. В този случай инстанцията на Scene се появява като един единствен Node с вътрешните си елементи скрити.

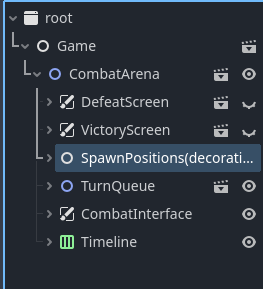
Scenes ни позволяват да структурирате кода на играта си по начина, по който желаем. Можем да композирате нодове, за да създадем персонализирани и сложни типове Node-ове, като игрален герой, който тича и скача, индикатор за живот, сандък, с който можем да взаимодействаме, и още.



фиг 3.3 Player Scene

# 3.2 Реализация на проекта

## 3.2.1 Структура на проекта



фиг 3.4

Главния Scene Game Node е наследен от няколко други Scene Nodes и нормални Node-ове. CombatArena управлява битката и визуализацията на ходове, TurnQueue управлява реда на ходовете, CombatInterface управлява избора на различни видове атаки и избора на противник, Timeline не съдържа логика той е контейнер за визуално показване на ходове, SpawnPosition не съдържа логика той е спомагателен Node за визуализация за къде да се поставят геройте.

## 3.2.2 Game Scene



фиг 3.5 Кода на Game Node

Game Scene чрез @onready анотацията инициализира референции към обекта CombatArena. С \_ready функцията извикваме enter\_battle и след това тази функция извиква initialize функцията на обекта CombatArena, после прави един къс таймер за да предотврати грешки и накрая извиква battle\_start функцията на CombatArena.

#### 

## 3.2.3 CombatArena

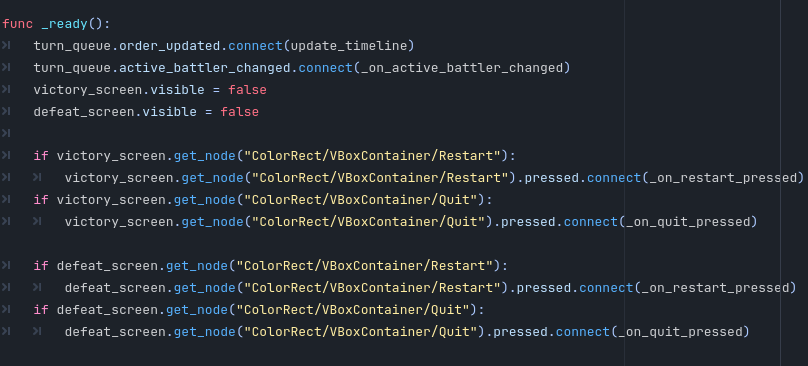
#### Инициализация на референции сочещи към различни Node-ове в scene-на. Active променлива за показване дали играта е свършила

#### 

фиг 3.6 инициализация на променливи

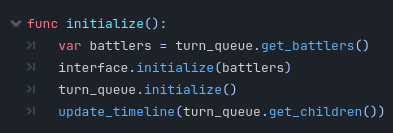
Свързване на сигнали. При промяна в реда на ходовете (order\_updated) се обновява таймлайн.

При смяна на активния участник (active\_battler\_changed) се извиква \_on\_active\_battler\_changed. Скрива екраните за победа/загуба при стартиране. Свързва бутоните "Restart" и "Quit" от екраните с функциите за рестарт и изход. If-овете проверяват когато се опитваме да вземем референция към бутоните дали са Null ако не са тогава свързва бутоните



фиг 3.7 свързване на сигнали и скриване на други екрани

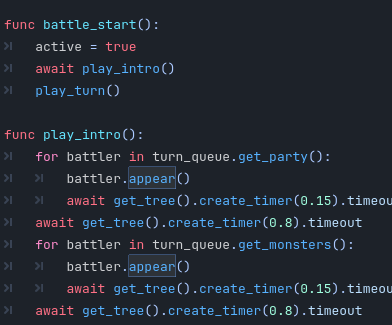
Получава участниците от turn\_queue. Извиква initialize функцията на interface. Стартира queue-то за ходове и обновява timeline.



фиг 3.8 функция за вземане на участници от queue

battle\_start започва битката като променя променливата актива на true. Изчаква анимацията play\_intro и започва първия ход play\_turn.

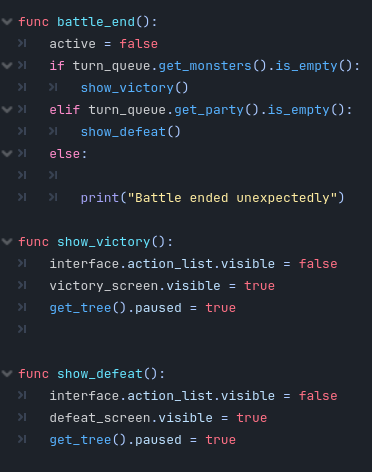
play\_intro анимира появата на героите и чудовищата последователно с паузи.



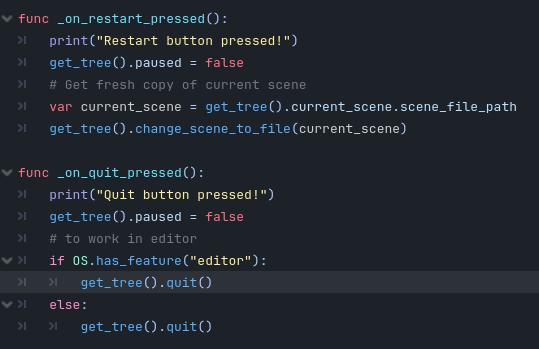
фиг 3.9 функция за започване на битка и поява на участниците

battle\_end проверява условия за край, ако няма чудовища → победа. Ако няма герои → загуба. Ако нито едно от двете → принтира, че битката е свършила по неочакван начин.

interface.action\_list.visible = false скрива интерфейса за избор на действия.В зависимост от резултата при battle\_end се показва екрана за победа/загуба и в двата случая get\_tree().paused = true спира играта



фиг 3.10 функции за край на играта



фиг 3.11 Функционалност на бутоните Restart и Quit

get\_active\_battler връща текущия участника, който е на ход.

selected = true маркира участника като "активен". get\_targets връща списък с възможни избори за атака:

Ако активният е герой → изборите са противници.

Ако е противник → целите са герои.

Ако няма цели (всички противници са умрели), битката приключва.

За герои party\_member == true:

interface.update\_actions(battler): Показва действията на героя в интерфейса.

await interface.select\_target изчаква играча да избере противник от списъка.

action взема избраното действие от интерфейса.

За противниците choose\_target(targets) избира герой.

pick\_random избира случайно действие от списъка с действия на противника.

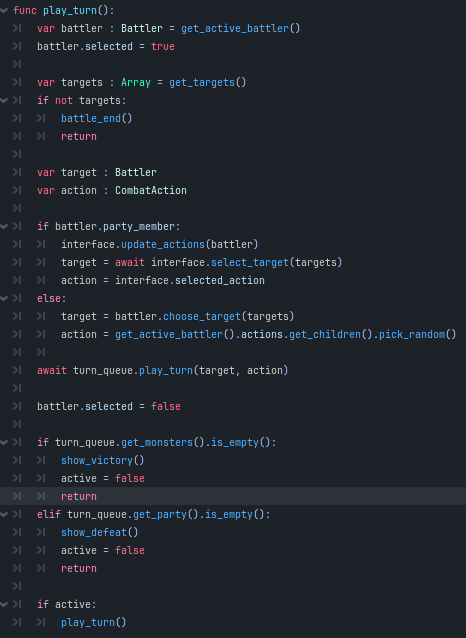
turn\_queue.play\_turn() изпълнява действието.

await изчаква приключването логиката на хода.

selected = false премахва визуалната маркировка на участника.

Проверките проверява дали няма противници или герои. Ако няма

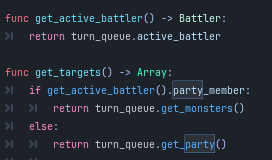
герои показва екран за загуба, ако няма противници показва екран за победа. Ако първите 2 проверките не са успешни, има 3та проверка за active дали е ако е true, кода вика рекурсивно play\_turn.



фиг 3.12 функция за играене на ход

get\_active\_battler връща текущия герой/противник който е на ход.

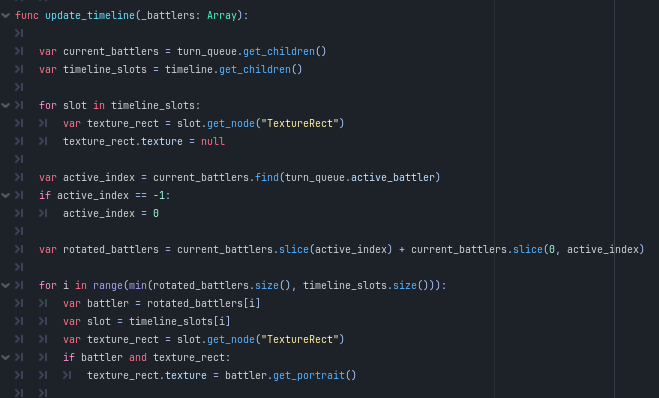
get\_targets връща списък с цели в зависимост от герой/противник



фиг 3.13

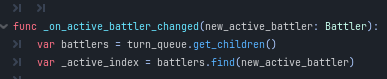
Показва портрети на участниците в реда на ходовете.

Активният участник е в началото на списъка.



фиг 3.14

актуализира

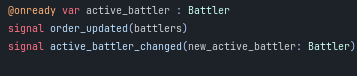


фиг 3.15

## 3.2.4 TurnQueue

TurnQueue държи всички противници и герои като деца и работи като масив.

чрез който определя реда на ходовете.



фиг 3.16 инициализация на active\_battler и сигнали за промяна на реда на ходовете и смяна на активния участник.

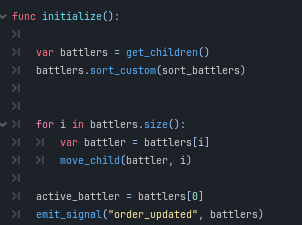
Взима всички деца (Battler нодове).

Сортира ги по скорост чрез sort\_battlers.

Пренарежда децата в сцената според сортирания ред.

Задава първия участник като активен.

Сигнализира за актуализиран ред.



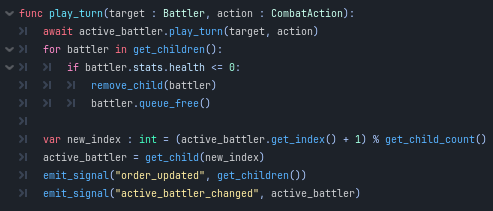
фиг 3.17 инициализация на началния ред на ходове

Изчаква изпълнението на хода play\_turn на активния Battler.

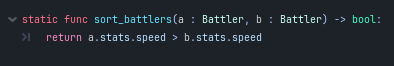
Премахва мъртви участници с health ≤ 0.

Определя следващия активен участник.

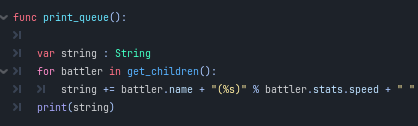
Сигнализира за нов ред и нова активна единица.



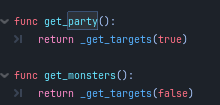
фиг 3.18 изпълняване на хода на активния играч и преминаване към следващия



фиг 3.19 сортиране на участниците по скорост

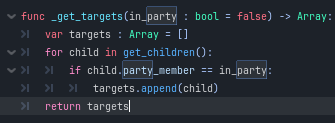


фиг 3.20 дебъг функция



фиг 3.21 връща списък с участници

in\_party – филтрира герои (true) или противници (false).

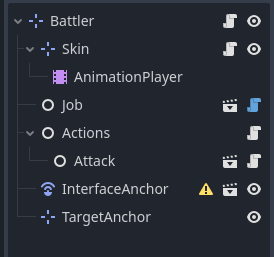


фиг 3.22 филтрация на опоненти и герои



фиг 3.23 взимане на всички герои и противници без филтрация

## 3.2.5 Battler



фиг 3.24 структура на базовия клас на герои и опоненти

TARGET\_OFFSET\_DISTANCE разстояни от TargetAnchor за позициониране.

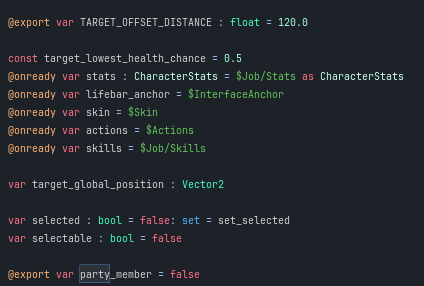
target\_lowest\_health\_chance шанс AI да атакува най-слабия (50%)

target\_global\_position крайна позиция в битката.

selected: дали Battler е избран за действие (променя визуален ефект).

selectable дали може да бъде избран.

party\_member: Флаг за герой/противник.

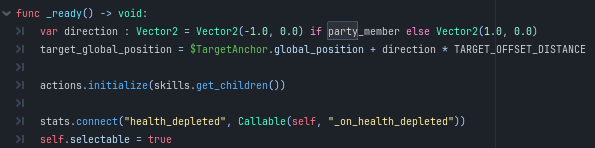


фиг 3.25 инициализация на променливи

Позициониране Battler се местят наляво, врагове - надясно.

Инициализира уменията.

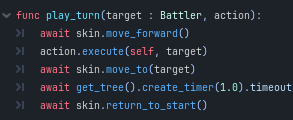
Свързва сигнал за смърт



фиг 3.26

Движение към целта → изпълнение на действие → връщане.

Изчакване между стъпките за визуална яснота



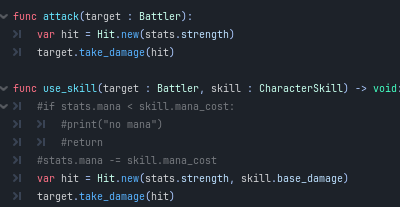
фиг 3.27

мигане за героя които е наред

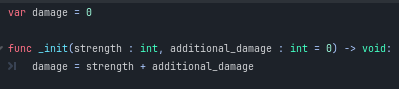


фиг 3.28

създава обект за атака



фиг 3.29



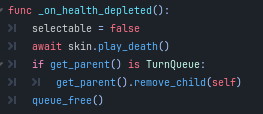
фиг 3.30 базов клас за атака

Функция за поемане на щети и пускане на анимация



фиг 3.32

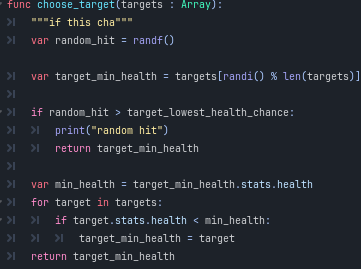
Премахване на героя/опонента при смърт



фиг 3.33

Опонентите имат шанс да атакуват героя с най-малко кръв.

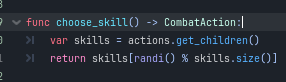
В друг случай избира случайна цел



фиг 3.34 логика на опоненти за избор



фиг 3.35 връщане на икона за UI



фиг 3.36 избор на случаен вид атака

#### 

### 3.2.5.1 Skin

Занимава се с анимирането на геройте

TURN\_START\_MOVE\_DISTANCE разстояние за движение в началото на ход.

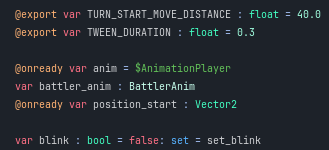
TWEEN\_DURATION време за анимациите на движение.

anim референция към анимационния плейър.

battler\_anim референция към нода с анимациите на героя.

position\_start начална позиция за връщане след движение.

blink контролира мигащия.



фиг 3.37 променливи

Скриване на обекта при стартиране.

Търсене на BattlerAnim нод сред децата (за анимациите).

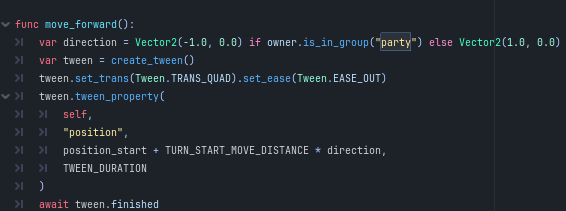


фиг 3.38

Движение напред в началото на ход:

Съюзниците се движат наляво, врагове - надясно.

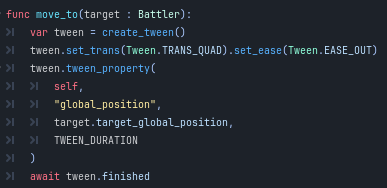
Използва Tween за плавна анимация.



фиг 3.39

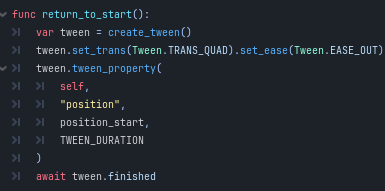
Движение към цел при атака:

Премества се до позицията на целта (target.target\_global\_position).



фиг 3.40

Връщане в начална позиция след атака.



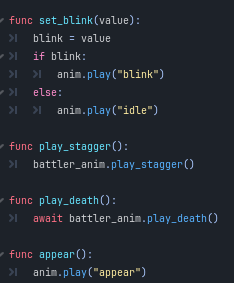
фиг 3.41

set\_blink Активиа мигане, когато героя е ход

play\_stagger анимация когато е поета щета

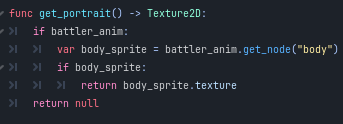
plat\_death анимация когато Battler остане без кръв

appear анимация за поява в началото на играта



фиг 3.42 визуални ефекти

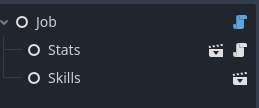
взимане на портрети за timeline



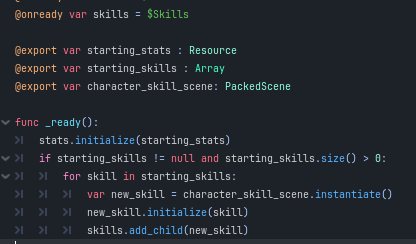
фиг 3.43

### 3.2.5.2 Job

Управление на характеристики и различните видове атаки



фиг 3.44 структура на job



фиг 3.45 инициализация на характеристиките и видовете атаки

#### 3.5.2.2.1 Stats

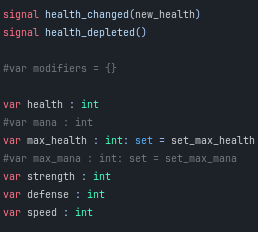
health\_changed при промяна на кръв (новият остатък)

health\_depleted при достигане на 0 кръв

health: текуща кръв

max\_health максимална кръв (със сетър за валидация).

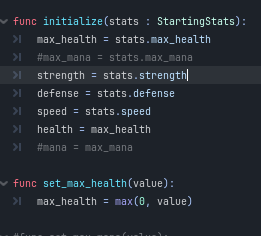
strength, defense, speed бойни характеристики



фиг 3.46 сигнали и променливи

Задаване на начални стойности от StartingStats ресурс (дефиниран в редактора).

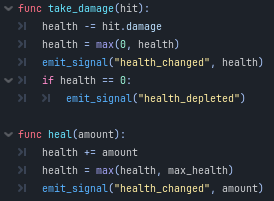
set\_max\_health гарантира, че max\_health никога не е отрицателно число.



фиг 3.47

take\_damage намаля кръвта според щетите.

heal възстановява здравето, но не може да надвишава max\_health



фиг 3.48 логика за щети и лечение

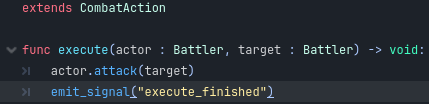
### 3.5.2.3 Action

Съхранява различните видове атаки на героите



фиг 3.49 базов клас за атаки

#### 3.5.2.3.1 Attack

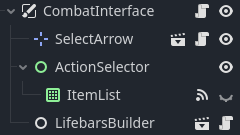


фиг 3.50 нормална атака

### 3.5.2.4 InterfaceAnchor и TargetAnchor

InterFaceAnchor се използва за позициониране на UI елементи, а TargerAnchor се използва за движенията на героите

## 3.2.6 CombatInterface



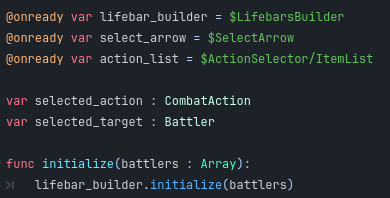
фиг 3.51 структура на CombatInterface

lifebar\_builder референция към нод, който създава ленти за живот.

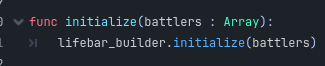
select\_arrow курсор за избор на цели в битката.

action\_list списък с умения/атаки (ItemList).

selected\_action и selected\_target запазват избраните действие и цел



фиг 3.52 .



фиг 3.53 инициализира UI елементите в началото на битката

select\_arrow се активира и чака избор .

връща избрания Battler като цел.

### 

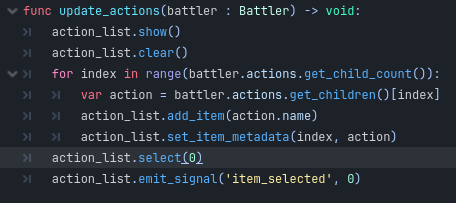
фиг 3.54 показва курсор за избор на противник

Показва action\_list.

Изчиства старите елементи.

Добавя имената на уменията от battler.actions.

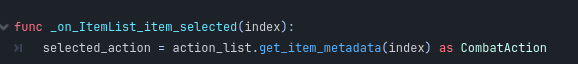
Запазва самите умения като metadata и избира първия елемент по подразбиране



фиг 3.55 показва списък с атаки и умения

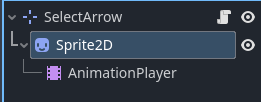
При избор на елемент от action\_list, взема метаданните (самия обект CombatAction).

Записва го в selected\_action за използване в хода.



фиг 3.56

### 3.2.6.1 SelectArrow



фиг 3.57 структура на SelectArrow

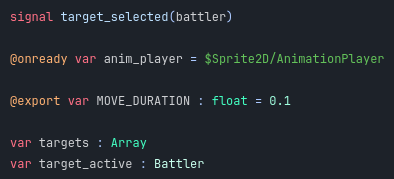
target\_selected за избор на цел.

anim\_player референция към анимационния плейър за ефекти.

MOVE\_DURATION време за движение между цели.

targets списък с налични цели (Battler нодове).

target\_active текущо избрана цел.



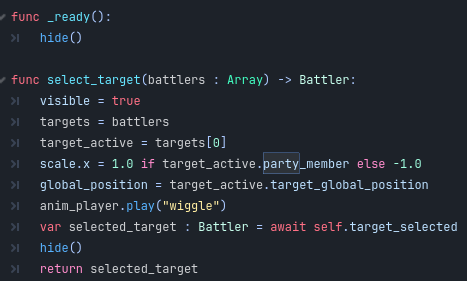
фиг 3.58

\_ready скрива стрелката, докато не се активира избор на цел.

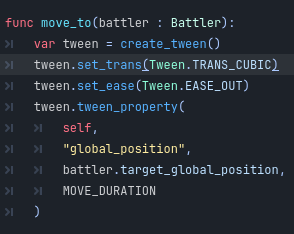
select\_target показва стрелката след това задава начална цел (първия в списъка) обръща стрелката хоризонтално за врагове (scale.x = -1) и мести стрелката до целта и пуска анимация.

Накрая чака избор (await target\_selected) и

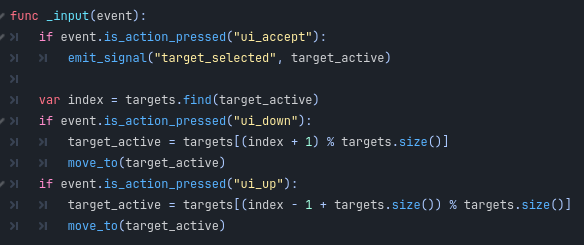
скрива стрелката и връща избраната цел



фиг 3.59



фиг 3.60 функция за плавно местене на курсора



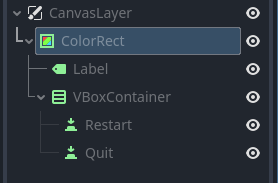
фиг 3.61 функция за обработка на вход

### 3.2.6.2 LifebarsBuilder

фиг 3.62 инициализация и създаване на лентите за кръв

## 

## 3.2.7 Meнюта



фиг 3.63 структура на менюта

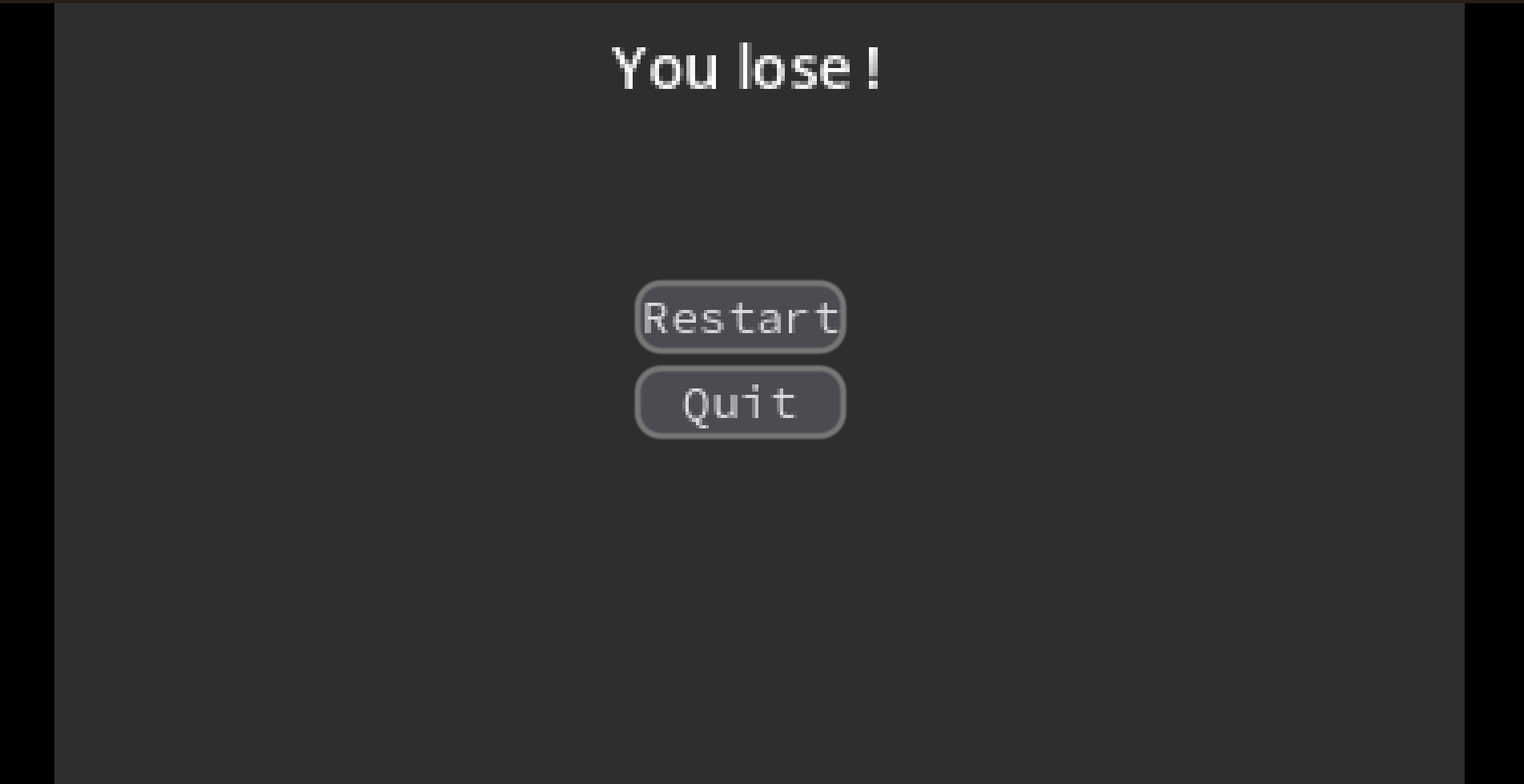
Когато всички противници са победени се появява VictoryScreen.

Restart бутона започва играта на ново, а quit бутона кара човека да напусне играта

фиг 3.64 меню при победа

Когато всички герои са победени се появява DefeatScreen.

Сходно на VictoryScreen DeafeatScreen има два бутона със същите функционалности.



фиг 3.65 меню при загуба

# Четвърта глава. Ръководство на потребителя

# 

## 4.1 Инсталация и системни изисквания

За стартиране на играта отворете Game Of Mid.exe след

като разархивирате Game Of Mid.rar

Препоръчителни изисквания :

OS: Windows 10 (64-bit)

CPU: Intel Core i7-1065G7

GPU: NVIDIA GeForce MX330

RAM: 8 GB

Дисково пространство: 250MB

## 4.2 Описание на играта и контроли

### 4.2.1 Контроли

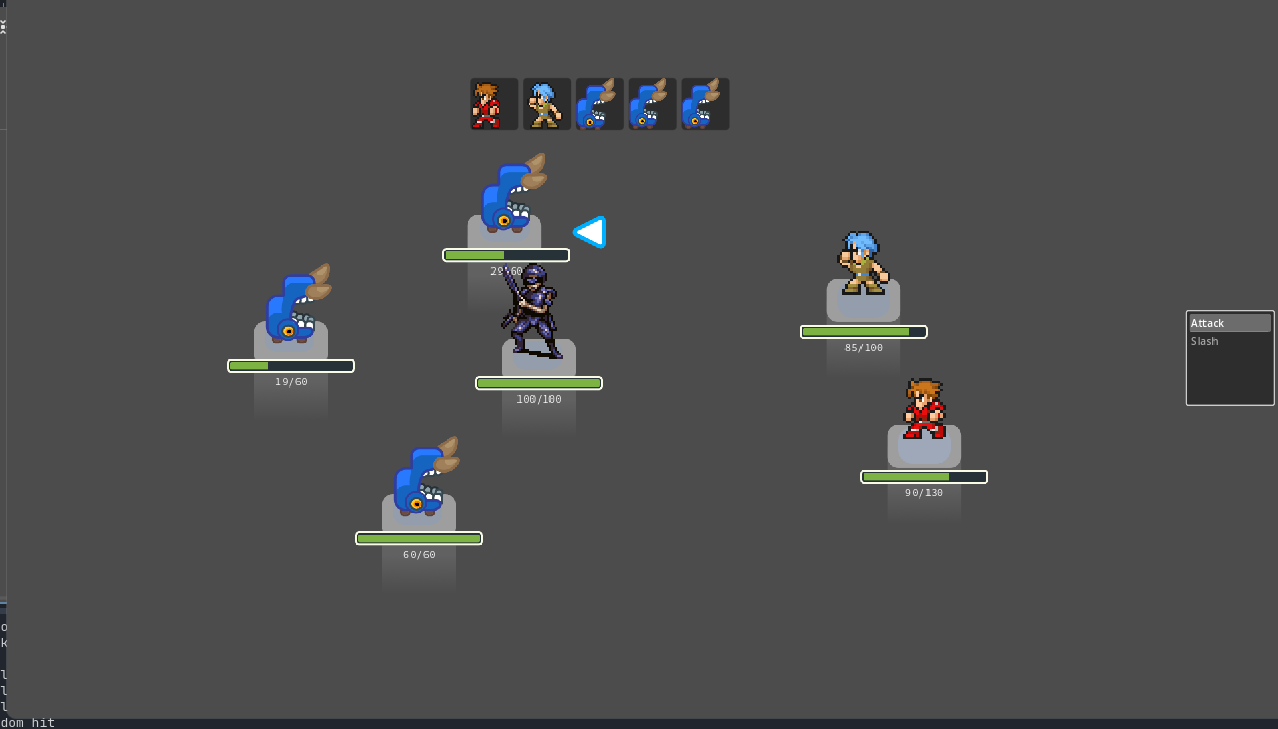
Местене на курсор за избор на противник - “ ↑”/”↓”

Кликване с мишка за избор на умение

За потвърждаване на избора - “Enter”

### 4.2.2 Описание на играта

При влизане в играта играчът веднага започва битка целта на играча е да победи всички опоненти.



фиг 4.1 Gameplay

# Заключение

Постигнати цели:

-Реализация на 2 вида противници

-Реализация на 2 героя

-Реализация на различни видове атаки

-Наличие на анимации

-Различни видове атаки

-Ходове на действие на героите

Бъдещо развитие:

* добавяне на система за запазване на прогреса
* стартово меню
* меню за пауза
* меню за настройки
* повече различни видове умения
* подобряване на менюто за избор на умения

# Използвана литература

1.[https://manager.bg](https://manager.bg/%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8/industriata-za-videoigri-e-izpravena-pred-baven-rastez-prez-2024-g-na-fona-na-slabi-prodazbi-na-konzoli) - увод

2. [https://delnik.bg/](https://delnik.bg/business/tehnologiite-s-izkustven-intelekt-promenyat-industriyata-za-videoigri/147032/) - увод

3. [https://blog.gsmarena.com](https://blog.gsmarena.com/angry-birds-epic-ios-android-windows-phone-game-review/) - angry birds epic

4. <https://en.wikipedia.org/wiki/Angry_Birds_Epic> - angry birds epic

5. <https://en.wikipedia.org/wiki/Honkai:_Star_Rail> - honkai star rail

6. <https://www.pockettactics.com/honkai-star-rail/review> - honkai star rail

7. <https://www.unrealengine.com/en-US> - unreal

8.<https://unity.com/use-cases> - unity

9.[https://docs.godotengine.org](https://docs.godotengine.org/en/stable/getting_started/introduction/first_look_at_the_editor.html) - godot

10.[https://docs.godotengine.org](https://docs.godotengine.org/en/stable/getting_started/introduction/godot_design_philosophy.html-) - godot

11.[https://www.reddit.com](https://www.reddit.com/r/godot/comments/17vq6kk/can_someone_explain_the_ready_function_to_me_like/) - \_ready

12.[https://gamedev.stackexchange.com](https://gamedev.stackexchange.com/questions/207050/what-is-the-difference-between-onready-and-ready) - \_ready vs @onready

13. [Programming a tactical strategy game in Godot 4](https://www.youtube.com/watch?v=w6jXgaWHo1s)

14. [Godot RPG videos (Turn-Based Combat) - YouTube](https://www.youtube.com/playlist?list=PLhqJJNjsQ7KHaAQcGij5SmOPpFjrDTHUq)

15 [https://github.com/GDQuest/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbnpFSUpfZ1FGWXZ3QnE4V29LXzJqUjJwOC1IUXxBQ3Jtc0tsVy1KOFd1ckxsSmNkY0J3YjFQWjNRTHEwSjFqVjNuc2cxeUF0RmtYcE9PRVBjeHlYM25FSDl0SE5wanA3T0hMUWlRU2lEdFUxakJIaGY2R3VnN1VwYlpDdGkyRGdhTFYyUnR1VE9tQUdtZ09SbUJkQQ&q=https%3A%2F%2Fgithub.com%2FGDQuest%2Fgodot-turn-based-rpg%2F&v=FV4JkwI4OF4)

16. [How to make a Video Game - Godot Beginner Tutorial](https://www.youtube.com/watch?v=LOhfqjmasi0&t=1222s)

17. [JRPG Turn Based Combat Tutorial in Godot 4](https://www.youtube.com/watch?v=HEexLmt7enc)

18.[https://github.com/16BitDev](https://github.com/16BitDev?tab=overview&from=2025-02-01&to=2025-02-08)

19.[How Conditional Turn Based Combat Works? | Godot 4](https://www.youtube.com/watch?v=Envh07viSOY&t=173s)

20.<https://www.youtube.com/playlist?list=PL0swe3EwWBiJcD5AVPt58ecqTrBDr1F5n>

21.[Sprite Animation Tutorial | By LucasRPDJ](https://www.youtube.com/watch?v=jzXoT7fXlp4)

22.[https://opengameart.org/](https://opengameart.org/art-search-advanced?keys=characters&title=&field_art_tags_tid_op=or&field_art_tags_tid=&name=&field_art_type_tid%5B0%5D=9&field_art_type_tid%5B1%5D=14&sort_by=count&sort_order=DESC&items_per_page=24&Collection=&page=5)

23.[Introduction to Custom Resources in Godot!](https://www.youtube.com/watch?v=s-BqbdY5dZM)

24. [Godot 4 Animation Player Tutorial](https://www.youtube.com/watch?v=V9kshvtj6s0&list=PLqF5LscxmBZWD3Ezk_PKw5w4QQ5VQzbtB&index=23&pp=iAQB)

# Съдържание

[**УВОД 4**](#_b2gxbphjt6k9)

[**Първа Глава 5**](#_g3uxolub3rdb)

[**Преглед на подобни продукти и на известни развойни среди 5**](#_kptszhhuneuq)

[1.1 Преглед на на подобни ходово-ролеви игри 5](#_ccoqx8t5aiay)

[1.1.1 Angry birds epic 5](#_idbe89x3ss72)

[1.1.2 Honkai Star Rail 6](#_fc4m062j7gnk)

[1.2 Технологии за създаване на игра 7](#_7my5iigt1kkb)

[1.2.1 Unreal 7](#_lz4psgdeo4pk)

[1.2.2 Unity 8](#_8a6ng547p3yv)

[1.2.3 Godot 9](#_bto9vhxkjst2)

[**Втора глава 11**](#_jp03ia68c4uc)

[**Функционални изисквания и аргументиран избор на развойна среда 11**](#_t49xtp9vwl0d)

[**2.1 Функционални изисквания 11**](#_fbqn1myi94af)

[2.2 Аргументация за избора на развойната среда и технология 12](#_1ci93xb)

[**Трета глава 14**](#_3whwml4)

[**Философски дизайн Godot 14**](#_sgmeqex65s9x)

[**Реализация на проекта 14**](#_vl37cyfxy6q7)

[3.1 Godot наследяване 14](#_e91zwqb4m4bc)

[3.1.1 Наследяване на Node-ове 15](#_i3cfqkpyo3hg)

[3.1.2 Scenes 16](#_v4sk0dnz4qqm)

[**3.2 Реализация на проекта 17**](#_vdkjss6msbch)

[3.2.1 Структура на проекта 17](#_pk6v2276uplc)

[3.2.2 Game Scene 18](#_t9znebsg3crr)

[3.2.3 CombatArena 18](#_oofc9zgftzz7)

[Инициализация на референции сочещи към различни Node-ове в scene-на. Active променлива за показване дали играта е свършила 18](#_e9cdbx4g91er)

[3.2.4 TurnQueue 26](#_ls13vfwl8ec4)

[3.2.5 Battler 28](#_zeiq5nbrsnhr)

[3.2.5.1 Skin 32](#_xwovnrbrevav)

[3.2.5.2 Job 34](#_5hfcf5epurjl)

[3.5.2.2.1 Stats 35](#_s91ob1271hd4)

[3.5.2.3 Action 37](#_xmwnpmqa102v)

[3.5.2.3.1 Attack 37](#_jzsret91zkgh)

[3.5.2.4 InterfaceAnchor и TargetAnchor 38](#_osqae6hgyk75)

[3.2.6 CombatInterface 38](#_9jo9r04tli1b)

[3.2.6.1 SelectArrow 39](#_aeeq56t4us8u)

[3.2.6.2 LifebarsBuilder 42](#_gdi5w49kbxsx)

[3.2.7 Meнюта 43](#_pwi7ip4eqf6x)

[**Четвърта глава. Ръководство на потребителя 45**](#_rpodavw3zlet)

[4.1 Инсталация и системни изисквания 45](#_wks9cau72uy)

[4.2 Описание на играта и контроли 45](#_chhwpor4npxm)

[4.2.1 Контроли 45](#_s5h3pv9c0f8m)

[4.2.2 Описание на играта 46](#_3g1zyfq94pjo)

[**Заключение 47**](#_7lhub1v5v533)

[**Използвана литература 48**](#_erw8cvt2emlz)

[**Съдържание 49**](#_2ffd0v3pdwq4)