Документација на играта

- Опис на играта:

Играта претставува 3Д парк со скриени богатства. Има 3 предмети: монети, ковчези и дијаманти. Целта е да се соберат 75 поени за време на 75 секунди. Секоја монета носи 2 поени, секој ковчег 5 и секој дијамант 10, заедно даваат 75. Поретките предмети се скриени во различни објекти како куќички на дрво и игралишта.

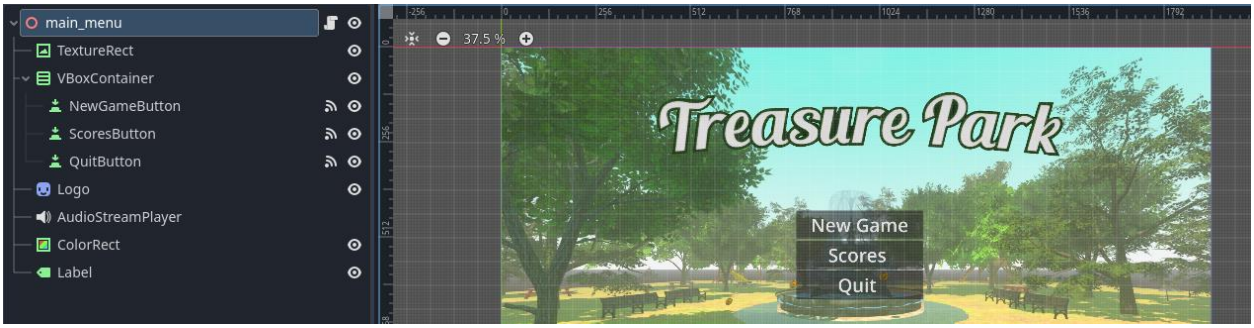
- Посебни својства:

Играта во JSON фајл ги зачувува најдобрите 5 резултати и ги прикажува после завршување на секоја партија како и во посебен екран кој се пристапува од гланото мени. Има копче за ресетирање на резултатите. Освен резултатот и времето играта покажува кои предмети и колку од нив ни фалат. Можеме сред играње да ја исклучиме играта на главниот екран.

- Уникатен голем модел: Паркот е направен со долго спојување на многу 3Д модели и нивно позиционирање на теренот кој е plane mesh со текстури на трева

- Manin menu сцена. (main_menu.tscn)

Главниот екран е направен од сцена изградена од 2д елементи. Тој содржи слика со употреба на textureRect node, лого (Sprite2d note), текст со објаснување (Label) и копчиња. Логото е самостојно изработено во photoshop и импортирано. Се користи AudioStreamPlayer за музика.



Оваа сцена е главна во едиторот преку project settings. Таа прва почнува и преку неа кон сите одиме.

За главниот root node е прикачина скрипта main_menu.gd. Скриптата пристапува до сцените за scores и terrain

```
const SCORES = preload("res://scores.tscn")
const TERRAIN = preload("res://ttterain.tscn")
```

И потоа преку табот Node се поставува што да се случи при клик на соодветните копчиња. Ова го правиме преку стискање на connect и поставување на функционалност во функцијата. Со ова го менуваме екранто на друга сцена.

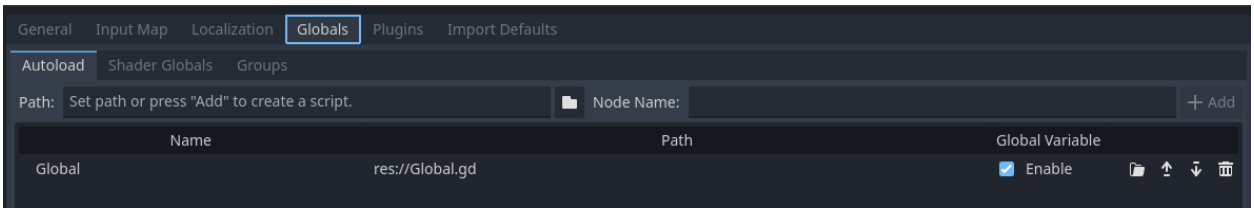
```
func _on_scores_button_pressed() -> void:
    get_tree().change_scene_to_packed(SCORES)

func _on_new_game_button_pressed() -> void:
    get_tree().change_scene_to_packed(TERRAIN)
```

Во ready функцијата која се случува при вчитување на сцената ја пуштаме музиката и го поставуваме курсерот да биде видлив.

```
func _ready() -> void:
    Input.set_mouse_mode(Input.MOUSE_MODE_VISIBLE)
    audio_player.play()  # Play sound when the scene loads
```

- Глобална скрипта (Global.gb)

Логиката на играта ја решавам со глобална скрипта која ја назначувам преку project settings.



Поставувам варијабла за резултат, json фајл во меморија на корисникот и scoreboard низа.

```
var final_score: String = "0"
const SAVE_PATH = "user://scoreboard.json"
var scoreboard = [0, 0, 0, 0, 0]
```

Преку load scores функцијата прво се проверува фали нашиот корисник го има фајлот на соодветната json локацијата. Доколку го има го отвараме и парсираме json-от. Доколку пак нема корисникот ваков фајл ја повикуваме save scores функцијата за да го генерираме.

Add score функцијата преку сортирачка и слајсирачка функција на scoreboard-от ни ги прикчува најдопбрите 5 резултати сортирани опаѓачки.

Save scores функцијата ни го отвара/креира json фајлот и ни ги ги запишува таму најдобртие 5 резултати.

Reset scores функцијата ни ги премува резултатите со scoreboard поставен на сите нули.

```
func load_scores() -> void:
    if FileAccess.file_exists(SAVE_PATH):
        var file = FileAccess.open(SAVE_PATH, FileAccess.READ)
        var content = file.get_as_text()
        file.close()
```

- Сцената со 3Д Паркот (ttterain.tscn)

3Д паркот го изградив скрос целосно преку долго спојување и редење на многу 3Д модели со нивните текстури.

Најпрво креирав MeshInstance3d со static body и collision shape. Формата на mesh-от е plane mesh значи ова ми е подот. Ја поставив големината. Потоа симнав текстура на жолта трева (leafy grass 1k) преку Poly Haven. На мојот mesh употребив surface material override за да ги поставам текстурите. Albedo, metalic и roughness соодветно. Преку UV1 ја скалирав големината.

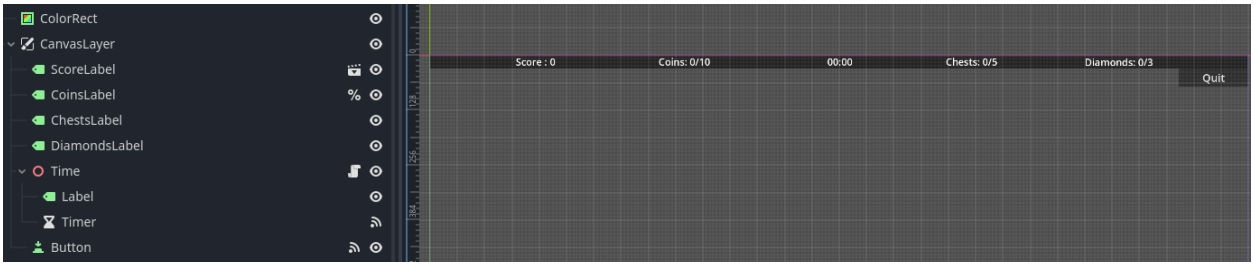
Понатаму импортирав многу 3Д модели и на ист начин текстурите им ги ставав. Моделите се различни дрвја, клупи, куќички, фонтанта и детско игралиште. Најголем дел ги преземав од sketchfab и некои од cgtrader.

Следно додадов музика и ги импортирав сцените за контролерот (player.tscn) и предметите кои и ги дуплирав и поставив насекаде на специфични места.

Откога ги позиционирав насекаде требаше да им дадам 3Д collision бидејќи никој немаше. Со десен клик -> editable children ги пристапив meshot на сите модели. Преку селекција на сите mesh-ови притиснав на опцијата create collision shape , type=trimerish, shape palcmenet = static body child.

Понатаму додадов небо, со world enviorment и directional light. Овде дефинитивно највеќе го променив изгледот на играта. Во directional light додаов shadows и directional shadows. Во World enviorment преку inspector-от ставив procedural sky mate сина боја, потоа додадов glow, fog, како и adjustments за brightness,contrast и saturation. Сево ова го даваа изгледот на светлината што се гледа.

Во однос на функционалности. Ставив на екранот overlay кој го прикажува резултатот како и колку од предметите имаме собрано. Има исто така и timer кој после 75s ќе не префли на крајната сцена. Quit копчето не префла на main_menu сцената.



Овие nodes вушност користат импортирани сцени кои преку скрипта ја вршат функционалноста.

Time.gd на следен начин го пресметува времето.

```
func time_left_to_live():
    var time_left = timer.time_left
    var minute = floor(time_left/60)
    var second = int(time_left) %60
    return [minute, second]
```

Самиот overlay со лабели со резултати има своја сцена наречена scores.gd.

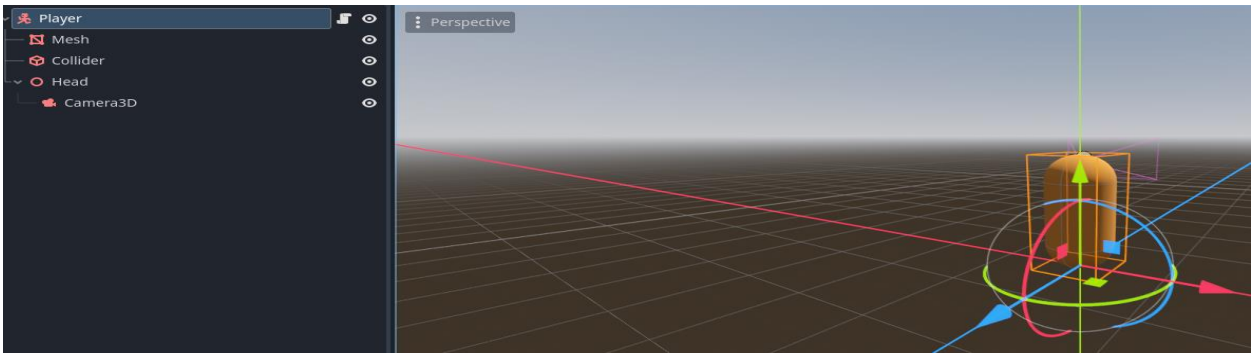
Во однос на скриптата ttterain.gd, во process функцијата која се случува секој frame го земаме резултатот од флобалната варијабла. Доколку е 75 значи максимален одиме на end_screen сцената.

```
func _process(delta: float) -> void:
    var score_number = int(Global.final_score.split(" ")[-1])
    if score_number==75:
        get_tree().change_scene_to_packed(END_SCREEN)
```

- Контолерот (player.tscn)

Контролерот е од огромна важност за играта, тој си има своја сцена и скрипта.

Сцената е составена од characterBody3d поврзан со скриптата за движење како и со своите деца mesh instance, collision shape како и camera3D.



Значи играчот претставува капсула чија камера е поставена пред неа за да гледа во first person. Телото се движи со скриптата во characterBody3d

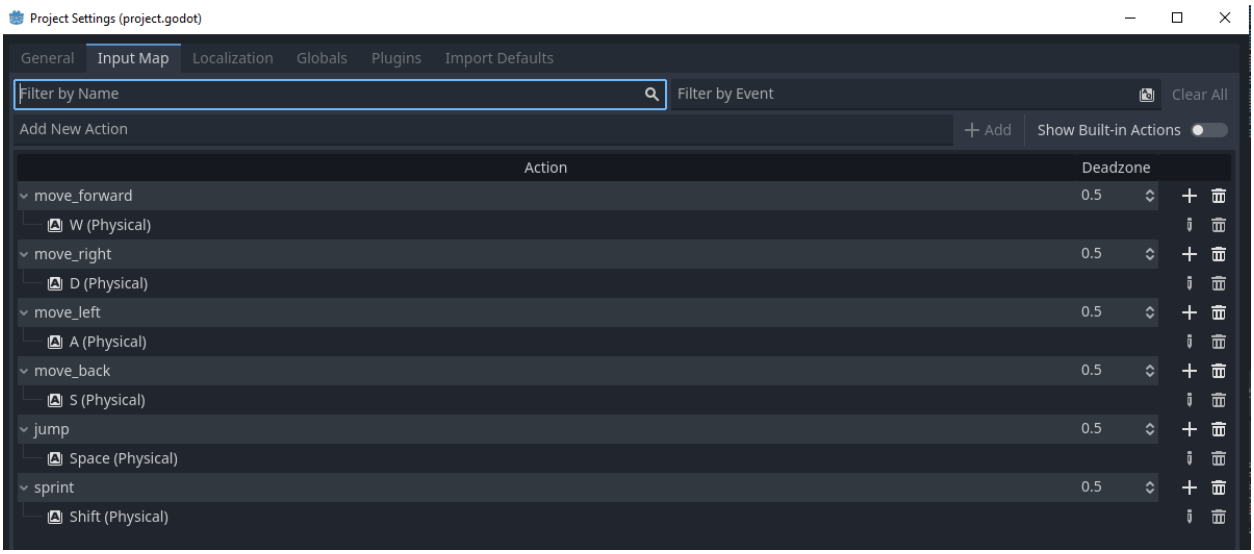
Во однос на скриптата:

Прво се земаат самите лабели од сцената со играта. Тие прикажуваат резултат и број на собрани items и се ажурираат во реално време.

```
@onready var coin_label = get_node("../CanvasLayer/CoinsLabel")
```

Во move променливите се складираат имињата на акциите за движење. Овие акциски имиња се дефинирани во Input Map од project settings, каде што можат да бидат поврзани со копчиња на тастатурата.

```
@export var move_right : String = "move_right"
@export var move_forward : String = "move_forward"
```



Во однос на функциите најважни се:

* increase score ја пристапува глобалната варијабла за резултат и го покажува нејзиниот резлутат за износ зависен од тоа кој предмет е собран. Оваа функција стои во контролеров бидејки тој секогаш ќе биде достапен каде што се движиме, потоа самиот предмет ја повикува функцијава со соодветен износ за покачување. Главната сцена преку лабела ке го покаже резултаот во real time.

* counter прима стринг кој означува кој објект бил собран и колку од тој објект се собрани а колку останууваат. За секој од нив соодветната варијабла ја зголемува за 1. Потоа главната свена ке ја прикаже варијаблава Пр. Diamonds: 1/3.

```
func increase_score(amount: int = 1):
    score += amount
    score_label.text = "Score: " + str(score)
    Global.final_score = score_label.text

func counter(type: String):
    if type == "coin":
        coins += 1
        coin_label.text = "Coins: " + str(coins) +"/10"
    elif type == "chest":
        chests += 1
        treasure_label.text = "Treasures: " + str(chests) +"/5"
    elif type == "diamond":
        diamonds += 1
        diamond_label.text = "Diamonds: " + str(diamonds) +"/3"
```

Останатите функции:

* ready ја зачувува почетната ротација на играчот и неговиот поглед. Ова овозможува правилно управување со погледот и движењето.

* unheald input го обработува кориснички внес.

* physics process. Ја поставува визиката на движење, скокање и трчање.

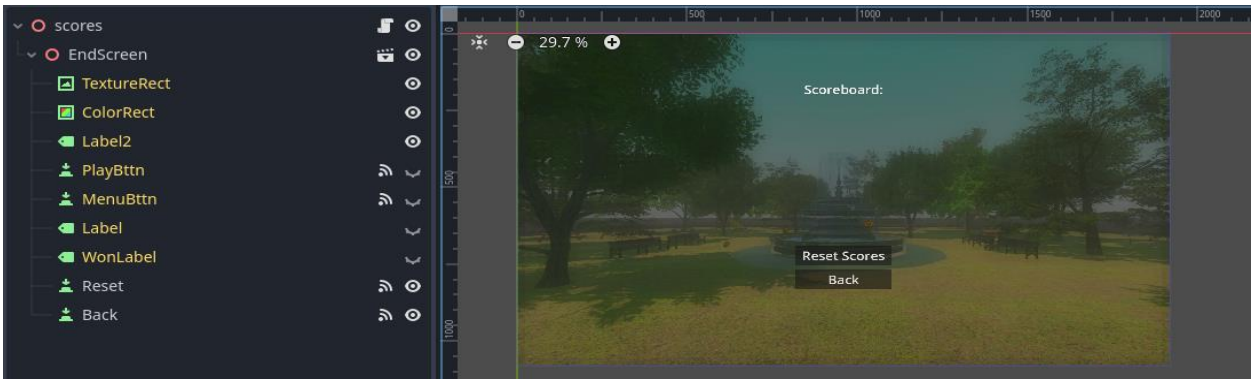
* rotate look ја контролира ротацијата на играчот според движењето на глувчето.).

* capture/release mouse курсерот од глувчето исчезнува или се појавува на

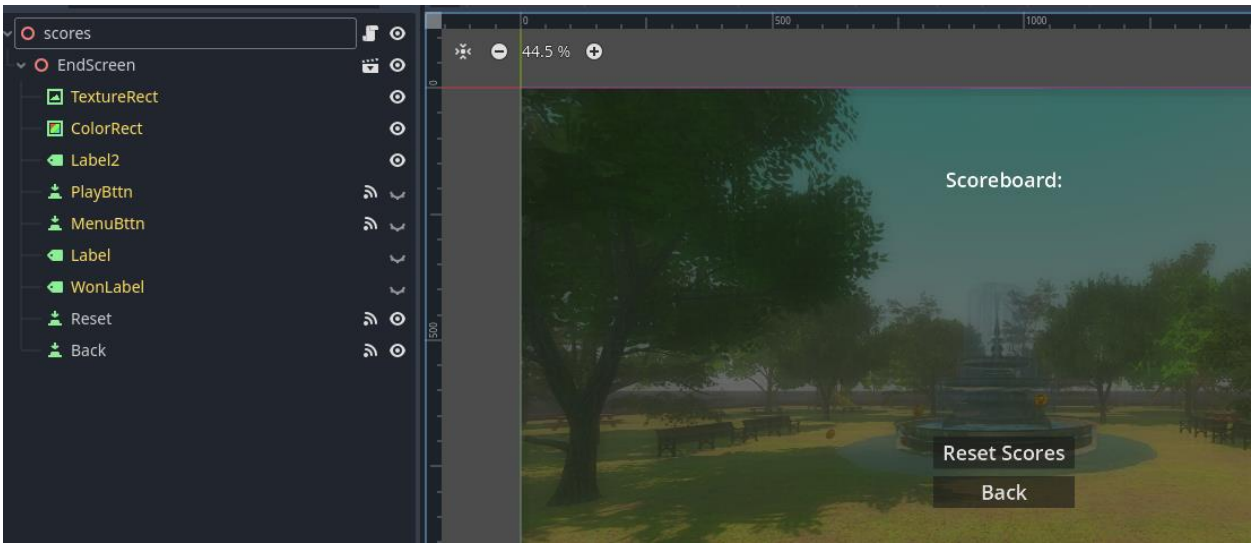
- ***Екран со резултати (scores.tscn)***

Сцената scores.tscn ги прикажува екранот со резултати. Слично е составен како почетниот екран. Овде дополнително екранот е затемнет и заматен. Ова го постигнувам преку ColorRect node. Во неговиот инспектор ја поставувам бојата на црна со opacity за да е прозирна. Заматувањето го постигнувам преку функција во shader-от која прави gaussian blur.

```
for (int x = -2; x <= 2; x++) {
    for (int y = -2; y <= 2; y++) {
        vec2 offset = vec2(float(x), float(y)) * pixel_size;
        blurred_color += texture(screen_texture, SCREEN_UV + offset) * weight[x + 2] * weight[y + 2];
    }
}
```



Главниот node е споен со скриптата scores.gd таа кога се вчитува сцената ни го генерира текстот за top scores кој го влече од глобалната варијабла. Копчињата се мапирани соодветно, копчето за reset scores исто така ја повикува ресет функцијата на глобалната варијабла.

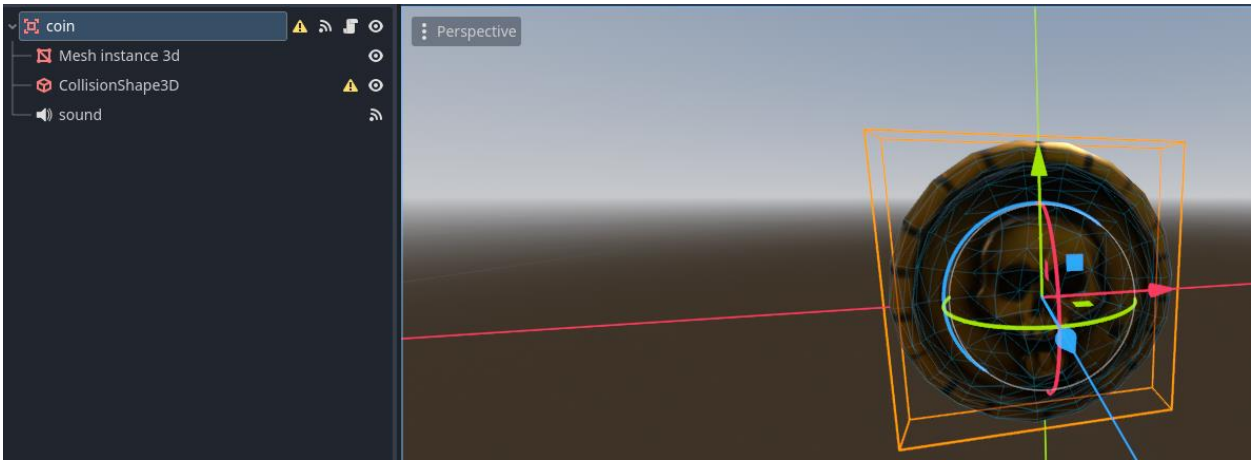


- Сцените на предметите

Секој предмет има своја сцена и своја скрипта. Coin.tscn, diamond.tscn, chest.tscn, Coin.gd, diamond.gd, chest.gd. Тие исто функциноираат и како вакви се импортитано во играта преку import child instance.

Овие предмети имаат иста функциналност, ке ја објаснам онаа на coin.

Секоја паричка е изградена од Area3d model со посебни деца Mesh instance 3d и collisionShape3D.



Area3D овозможува детекција на влез и излез на објекти во одредена зона (on body entered), MeshInstance3D го прикажува 3D моделот на паричката, а CollisionShape3D дефинира физички судир за да детектира кога играчот го допира објектот.

Исто така имаме и node за звук за играчот да слушне кога паричка е собрана.

Area3D е поврзан за скриптата, која ги има функционалностите.

Најпрво во process функцијата која се случува секој еден frame. Поставуваме вертикална ротација на објектот со брзина од 0.01. Со ова секојапаричка ке се врти околу својата оска.

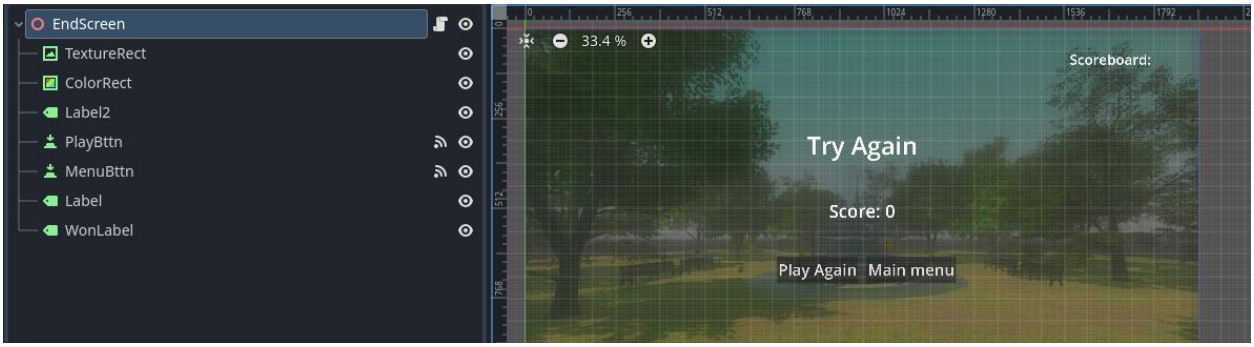
```
func _process(delta: float) -> void:
    rotate_y(0.01)
```

Како што напоменав со Area3D имаме опција да детектираме кога некој навлегов во објектот. Кога ке се случи тоа користам функција за пуштање на звукот дека објектот е собран, потоа најважно го исклучувам collisionShape3d со ова гарантирам дека корисникот нема два пати иста паричка да ја собере. Наредно ја кријам паричката од полето со hide и потоа со increase_score и со counter всушност повикувам мои custom функции кои се дефинирани во скриптата на контролерот (player.gd). Тие на сцената со играта ке ги сменат лабелите горе за да го прикажат прво соодветниот score, второ колку x-predmeti се собрани. Пр: score:2 coins: 1/10

```
func _on_body_entered(body: Node3D) -> void:
    $sound.play()
    $CollisionShape3D.disabled = true
    hide()
    body.increase_score(2)
    body.counter("coin")
```

- Завршна сцена (end_screen.tscn)

Завршната сцена го реискорисутва темниот overlay од scores сцената. Овдека го прикажуваме резултатот, scorebaord-от како и копчиња за нова игра и одење на менито. Лабелата WonLabel прикажува TryAgain но доколку сме во скриптата од играта ќе ни се смени текстот од лабелата на “You Won!”.



Освен што ни ги поставува копчињата и менува текстот, скриптата ги повикува нашите глобални функции за вчитување и поставување на резултат и со тој ни го поставува текстот на лабелите.

```
var scoreboard_text = "\nTop Scores:\n"
for score in Global.scoreboard:
    scoreboard_text += str(score) + "\n"

label.text = Global.final_score
scoreLabel.text = scoreboard_text
```

Изработил:

Марио Шекеровски 223287