

Liceul Teoretic “Mihail Kogălniceanu” Vaslui

LUCRARE PENTRU OBȚINEREA ATESTATULUI PROFESIONAL

*profil matematică – informatică,
intensiv informatică*

Absolvent:
Turcu Andrei Cristian

Profesor îndrumător:
Nicu Vlad Laurentiu

- 2024 -

MASTER OF DODGE

JOC REALIZAT IN GODOT

INTRODUCERE

Creare jocurilor cu un engine este de multe ori mai simplă și mai rapidă decât cu ajutorul unui limbaj de programare și al unui API grafic, fiind mai ușoară și mai ușor de înțeles. În unele cazuri este necesară utilizarea unui astfel de unealtă deoarece sunt anumite utilități care nu pot fi realizate în timp util, un altfel de exemplu ar fi exportarea aplicației pe diferite platforme.

Un dezavantaj este viteza cu care rulează programul deoarece engine-ul nu este optimizat special pentru proiectul tău și particularitățile pe care le conține acesta.

Înțelegerea funcționării unui engine este utilă celor care vor să aprofundeze dezvoltarea de jocuri/aplicații și programarea modularizată sau persoanelor care vor să profeseze în industria jocurilor video sau a dezvoltării de aplicații.

Această lucrare își propune prezentarea nu doar a considerațiilor teoretice, dar și a unui proiect realizat într-un astfel de engine, Godot, cu explicații detaliate și a implementărilor în limbajul GDScript.

Fiind pasionat de jocuri am hotărât să prezint un joc și o modalitate de creare a acestuia prin intermediul engine-ului Godot pe care l-am considerat util în cadrul aplicației mele.

CONSIDERAȚII TEORETICE

Engine-ul Godot Godot Engine e un game engine plin de utilități, cross-platform folosit pentru crearea jocurilor și aplicațiilor 2D și 3D printr-o interfață unificată. Acesta asigură un set complet de unelte uzuale, astfel utilizatorul se poate concentra pe crearea jocurilor fără să reinventeze roata. Jocurile sau aplicațiile pot fi exportate printr-un “click” pentru diferite platforme, inclusiv platformele majore de desktop (Linux, macOS, Windows) precum și cele pentru telefoanele mobile (Android, iOS) cât și cele bazate pe web (HTML5). Godot este complet gratis și are sursa liberă sub licența permisivă MIT. Jocurile sau aplicațiile utilizatorilor sunt ale lor și numai ale lor. Dezvoltarea engine-ului e complet independentă și condusă de comunitate, lăsând utilizatorii să contureze engine-ul astfel încât să le întâlnească așteptările. Acest joc a fost scris în limbajele specifice engine-ului Godot și anume GDScript și TSCN.




GDScript este un limbaj de programare de nivel înalt, orientat pe obiecte, imperativ și scris treptat, construit pentru Godot. Folosește o sintaxă bazată pe indentare similară cu limbaje precum Python. Scopul său este de a fi optimizat și strâns integrat cu Godot Engine, permițând o mare flexibilitate pentru crearea și integrarea conținutului.

GDScript este complet independent de Python și nu se bazează pe acesta.

Keyword	Description
if	See if/else/elif .
elif	See if/else/elif .
else	See if/else/elif .
for	See for .
while	See while .
match	See match .
break	Exits the execution of the current <code>for</code> or <code>while</code> loop.
continue	Immediately skips to the next iteration of the <code>for</code> or <code>while</code> loop.
pass	Used where a statement is required syntactically but execution of code is undesired, e.g. in empty functions.
return	Returns a value from a function.
class	Defines an inner class. See Inner classes .
class_name	Defines the script as a globally accessible class with the specified name. See Registering named classes .
extends	Defines what class to extend with the current class.
is	Tests whether a variable extends a given class, or is of a given built-in type.
in	Tests whether a value is within a string, array, range, dictionary, or node. When used with <code>for</code> , it iterates through them instead of testing.
as	Cast the value to a given type if possible.
self	Refers to current class instance.
signal	Defines a signal.
func	Defines a function.
static	Defines a static function or a static member variable.
const	Defines a constant.
enum	Defines an enum.
var	Defines a variable.

Operatori

Mai jos este lista operatorilor acceptați și prioritatea acestora.

Operator	Description
<code>()</code>	Grouping (highest priority) Parentheses are not really an operator, but allow you to explicitly specify the precedence of an operation.
<code>x[index]</code>	Subscription
<code>x.attribute</code>	Attribute reference
<code>foo()</code>	Function call
<code>await x</code>	Awaiting for signals or coroutines
<code>x is Node</code>	Type checking See also <code>is_instance_of()</code>  function.
<code>x ** y</code>	Power Multiplies <code>x</code> by itself <code>y</code> times, similar to calling <code>pow()</code>  function. Note: In GDScript, the <code>**</code> operator is left-associative  . See a detailed note after the table.
<code>~x</code>	Bitwise NOT
<code>+x</code> <code>-x</code>	Identity / Negation
<code>x * y</code> <code>x / y</code> <code>x % y</code>	Multiplication / Division / Remainder The <code>%</code> operator is additionally used for format strings . Note: These operators have the same behavior as C++, which may be unexpected for users coming from Python, JavaScript, etc. See a detailed note after the table.
<code>x + y</code> <code>x - y</code>	Addition (or Concatenation) / Subtraction
<code>x << y</code> <code>x >> y</code>	Bit shifting
<code>x & y</code>	Bitwise AND
<code>x ^ y</code>	Bitwise XOR
<code>x y</code>	Bitwise OR

<pre> x == y x != y x < y x > y x <= y x >= y </pre>	<p>Comparison</p> <p>See a detailed note after the table.</p>
<pre> x in y x not in y </pre>	<p>Inclusion checking</p> <p><code>in</code> is also used with the <code>for</code> keyword as part of the syntax.</p>
<pre> not x !x </pre>	<p>Boolean NOT and its unrecommended alias</p>
<pre> x and y x && y </pre>	<p>Boolean AND and its unrecommended alias</p>
<pre> x or y x y </pre>	<p>Boolean OR and its unrecommended alias</p>
<pre> true_expr if cond else false_expr </pre>	<p>Ternary if/else</p>
<pre> x as Node </pre>	<p>Type casting</p>
<pre> x = y x += y x -= y x *= y x /= y x **= y x %= y x &= y x = y x ^= y x <<= y x >>= y </pre>	<p>Assignment (lowest priority)</p> <p>You cannot use an assignment operator inside an expression.</p>

Ca orice altceva în Godot, interfața cu utilizatorul este construită folosind noduri, în special noduri de control. Există multe tipuri diferite de controale care sunt utile pentru crearea unor tipuri specifice de GUI. Pentru simplitate, le putem separa în două grupe: conținut și aspect.

Controalele tipice ale conținutului includ:

Butoane

Etichete

LineEdit și TextEdit

Controalele tipice de aspect includ:

BoxContainers

MarginContainers

ScrollContainers

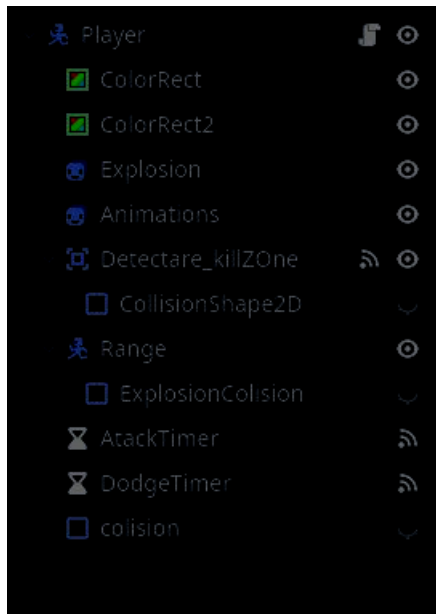
TabContainers

Ferestre pop-up

Scopul jocului este să obții un scor cât mai mare. Scorul crește cu fiecare inamic ucis. Jucătorul are două abilități: abilitatea de a se feri (acesta se mișcă cu o viteză foarte mare și este invincibil pentru foarte puțin timp) și abilitatea de a ataca. După folosirea unei abilități jucătorul trebuie să aștepte câteva secunde pentru a o mai folosi. Două abilități diferite pot fi folosite în același timp.



Jucător



Codul din spatele caracterului controlat de jucător

Variabilele folosite

```

var SPEED = 50.0
var alive = 1
@onready var ap = $Animations
@onready var exp = $Explosion
var available_atack = true
var available_dodge = true
var killable = 1

```

Pentru a face caracterul sa fie mai ușor de văzut o sa adăugam niște dreptunghiuri colorate care se invar in jurul sau

```

$ColorRect.rotation = $ColorRect.rotation+0.4
$ColorRect2.visible = true
$ColorRect2.rotation = $ColorRect.rotation+0.1

```

Pentru a controla caracterul inițial verificam daca acesta este in viață după care verificam daca au fost apăstate tastele de mișcare si in caz ca da îl mișcam

```

if alive :
    var directionx = Input.get_axis("move_left",
"move_right")
    if directionx:
        velocity.x = directionx * SPEED
        var directiony = Input.get_axis("move_up",
"move_down")
        if directiony:
            velocity.y = directiony * SPEED
            move_and_slide()

```

In caz ca jucătorul își schimba orientarea

```

if directionx!=0 :
    ap.flip_h = (directionx==-1)

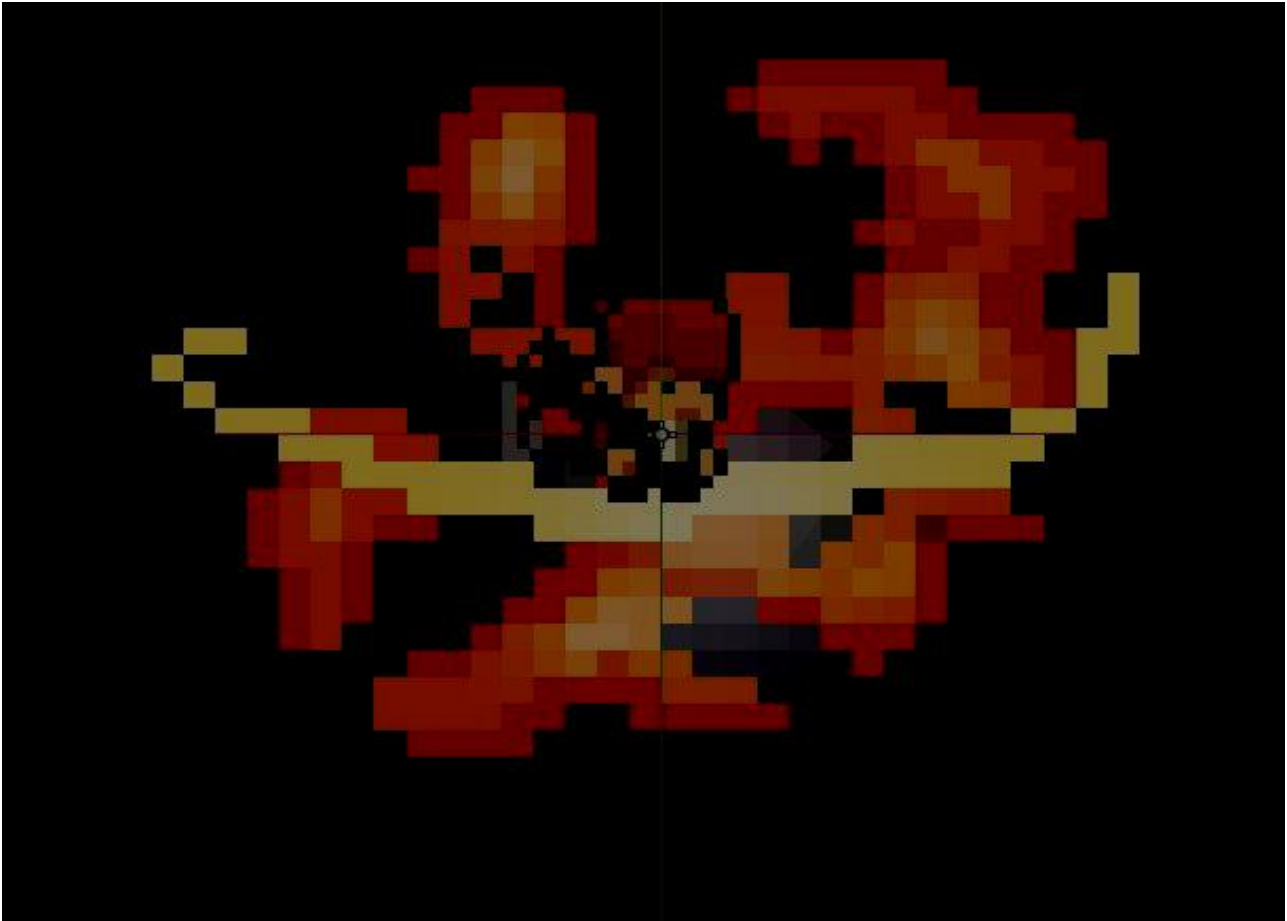
```

Pentru ca jucătorul sa se oprească din mișcat ii schimbam manual viteza la fiecare pas

```

velocity.x=0;
velocity.y=0;

```



Pentru a abilitatea de atac verificam daca este posibil. Daca da îl folosim si pornim un cronometru pentru a ne asigura ca abilitatea nu pote fi folosita constant

```
if Input.is_action_pressed("atack") and available_atack :  
    available_atack = false  
    $Range/ExplosionColision.disabled = false  
    $AttackTimer.start()  
    ap.play("Attack")  
    exp.play("Attack")
```

Daca cronometrul pentru atac a expirat jucătorul poate sa atace din nou

```
func _on_timer_timeout():  
    available_atack = true  
    pass
```

Pentru a abilitatea de ferit verificam daca este posibila efectuarea acesteia. Daca da oprim coliziunea cu inamicii, ii mărim viteza pentru un scurt timp si pornim un cronometru pentru a ne asigura ca abilitatea nu pote fi folosita constant

```
if Input.is_action_pressed("dodge") and available_dodge :
    available_dodge = false
    $DodgeTimer.start()
    $Detectare_kill1ZOne/CollisionShape2D.disabled = true
    SPEED *=4
    while( ap.rotation != 3 ):
        ap.rotation = ap.rotation+1
        await get_tree().create_timer(0.1).timeout
        ap.rotation = 0
        SPEED/=4
    $Detectare_kill1ZOne/CollisionShape2D.disabled
= false
```

Daca timerul pentru ferit a expirat jucătorul poate sa se fereasca din nou

```
func _on_dodge_timer_timeout():
    available_dodge = true
    pass
```

Pentru a ști ce animație sa afișam verificam daca jucătorul este in mișcare

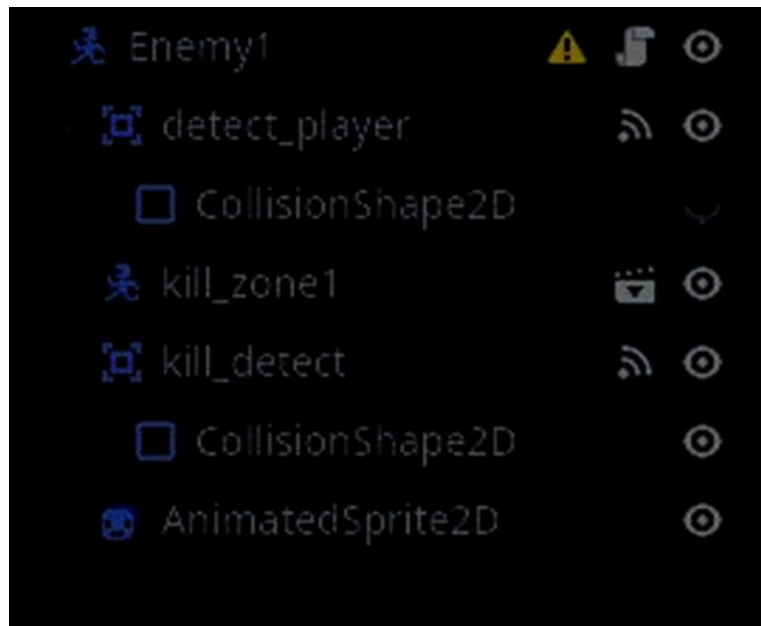
```
func update_animations(directionx,directiony):
    if directionx==0 and directiony==0:
        ap.play("Idle")
    else :
        ap.play("Run")
```

Pentru a ști dacă jucătorul este atacat verificăm dacă a intrat în contact cu un inamic. Dacă da semnalăm faptul acesta

```
func _on_detectare_kill_z_one_body_entered(body):  
    if killable :  
        ap.play("Death")  
        alive = 0  
        killable=0  
        get_parent().get_node("foreground").visible =  
true
```



Inamicul



Codul din spatele inamicului

```
variabile
const SPEED = 5.0
var player_position
var target_position
var alive = 1
@onready var player = get_parent().get_node("Player")
@onready var ap = $AnimatedSprite2D
```

În cazul în care inamicul este în viață îi facem să urmărească jucătorul

```
if alive :
    ap.play("Idle")
    player_position = player.position
    target_position = (player_position -
position).normalized() * SPEED
    velocity = Vector2(target_position * SPEED)
    move_and_slide()
```

In cazul in care inamicul detectează atacul jucătorului ii eliminam si creştem scorul

```
func _on_kill_detect_body_entered(body):  
    Global.Score += 1  
    alive=0  
    self.queue_free()
```

Codul din spatele generatorului de inamici

Ne încărcăm scena cu inamicul

```
var enemy_scene = preload("res://Enemy/enemy_1.tscn")
```

Pornim un cronometru pentru a ştii când este momentul sa mai generam un inamic

```
func _ready():  
    $Timer.start()
```

În cazul a sosit timpul sa mai adăugam un inamic la scena curenta încărcăm un inamic si pornim din nou cronometrul. In cazul in care jucătorul a fost eliminat nu mai pornim cronometrul

```
func _on_timer_timeout():  
    var ene = enemy_scene.instantiate()  
    ene.position = -pos  
    get_parent().add_child(ene)  
    if get_parent().get_node("Player").alive ==0 :  
        $Timer.stop()  
    pass
```

Afișăm animația specifică generatorului

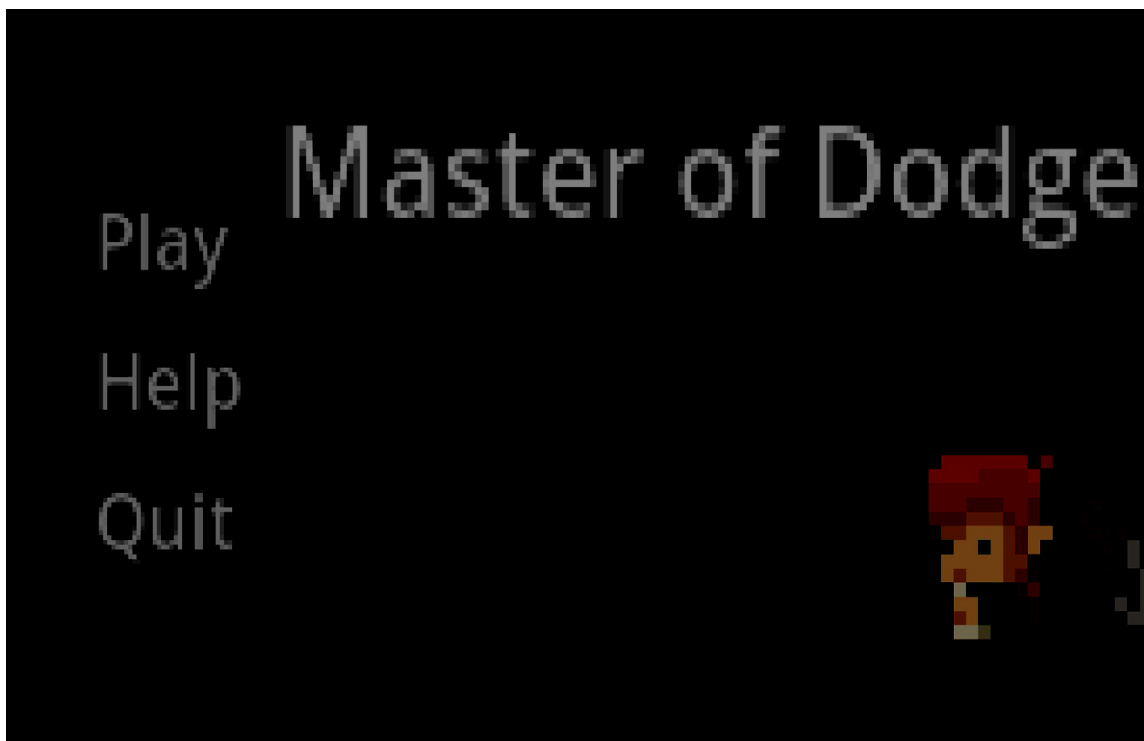
```
func _process(delta) :  
    play("default")  
    pass
```

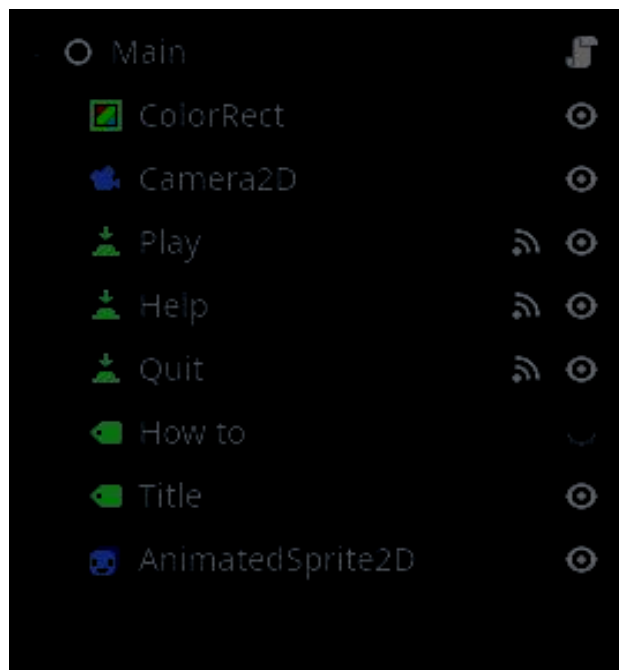
La începutul rulării

Încărcăm o scena specială care conține doar muzica jocului
Adăugăm o variabilă globală de scor

Interfață

Meniul





Pentru a face meniul mai interesant adăugam o animatie

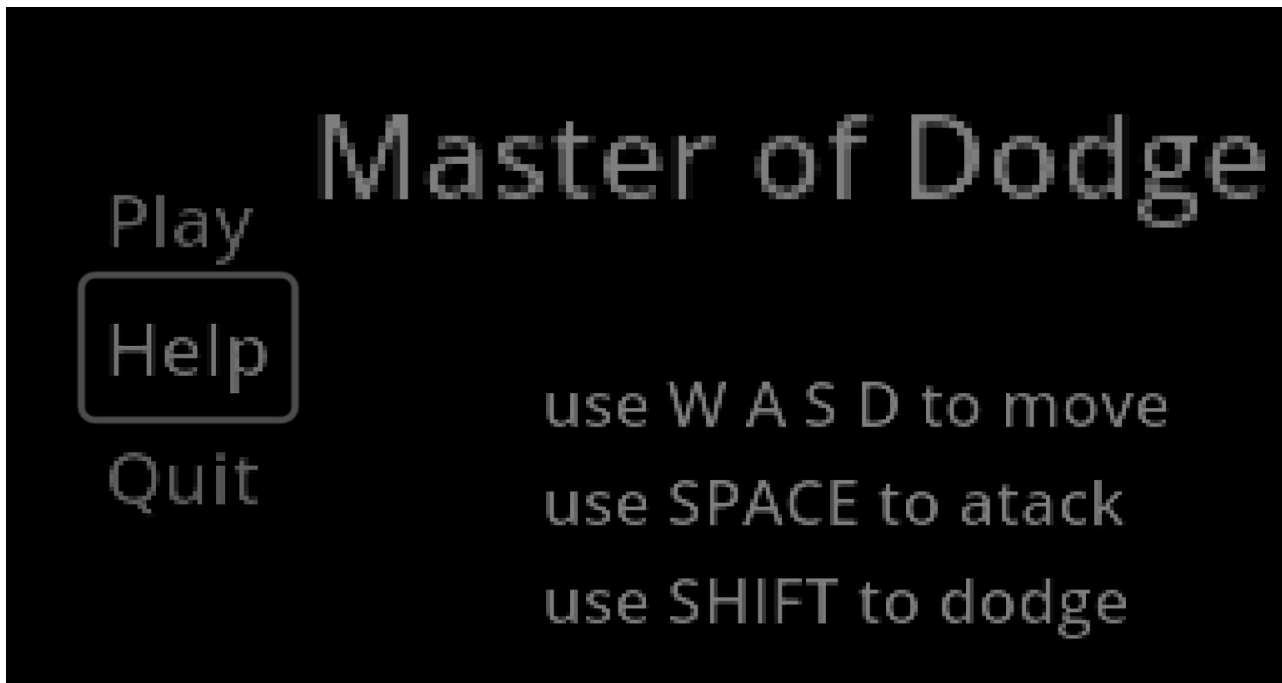
```
$AnimatedSprite2D.play("default")
```

In cazul apăsării butonului Play schimbam scena

```
func _on_play_pressed():  
    get_tree().change_scene_to_file("res://Scenes/fight_room.  
tscn")  
    pass
```

In cazul apăsării butonului Help afişam un text cu indicații și ascundem animația inițiala

```
func _on_help_pressed():  
    ht.visible = !ht.visible  
    $AnimatedSprite2D.visible = !$AnimatedSprite2D.visible  
    pass
```



În cazul apăsării butonului Quit închidem jocul

```
func _on_quit_pressed():  
    get_tree().quit()  
    pass
```

Interfața din timpul jocului

În cazul apăsării butonului Back schimbăm la scena cu meniul și resetăm scorul

```
func _on_pressed():  
    get_tree().change_scene_to_file("res://Scenes/main.tscn")  
    Global.Score=0
```

Scorul o să îl afișăm convertind variabila ce îl memorează în text

```
func _process(delta):  
    text= str(Global.Score)
```

BIBLIOGRAFIE

<https://docs.godotengine.org/en/stable/>
<https://gamedevelopment.tutsplus.com/tutorials/finite-state-machines-theory-and-implementation--gamedev-11867>
<https://www.aleksandrhovhannisyan.com/blog/finite-state-machinefsm-tutorial-implementing-an-fsm-in-c/>
<https://gamefromscratch.com/>
<https://www.gdquest.com/>
https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_basics.html

Cuprins

INTRODUCERE	3
CONSIDERAȚII TEORETICE	4
Interfața cu utilizatorul.....	8
Jucător	9
Inamicul	14
Interfață	16
Meniul	16
Interfața din timpul jocului	18
BIBLIOGRAFIE	19
Cuprins	20