

人工智能学院

课程设计说明书

|  |  |
| --- | --- |
| 题目 | 末日狂潮  ——游戏编码清单 |
| 专业 | 软件工程 |
| 班级 | 21软件工程01班 |
| 学生姓名 | 谢菁菁、谢恒艺、李世珍、叶羽霜 |
| 指导老师 | 高广尚 |
| 实验时间 | 2023/11/20——2023/12/24 |
| 实验地点 | 文综楼 |

目录

[1.开发平台与工具 3](#_Toc9482)

[1.1godot游戏引擎 3](#_Toc19040)

[2.程序设计 3](#_Toc20221)

[2.1玩家角色 3](#_Toc7995)

[2.1.1描述 3](#_Toc8305)

[2.1.2 编码设计 3](#_Toc25199)

[2.2怪物 5](#_Toc14660)

[2.2.1 描述 5](#_Toc21826)

[2.2.2 编码设计 6](#_Toc31489)

[2.3 时间UI设计 8](#_Toc19210)

[2.3.1描述 8](#_Toc19407)

[2.3.2 编码设计 9](#_Toc6353)

[2.4血量 10](#_Toc7690)

[2.4.1描述 10](#_Toc31509)

[2.4.2编码设计 10](#_Toc18816)

[2.5攻击动作 10](#_Toc27279)

[2.5.1描述 10](#_Toc27550)

[2.5.2编码设计 11](#_Toc24615)

[2.6角色状态 11](#_Toc28190)

[2.6.1描述 11](#_Toc18806)

[2.6.2编码设计 11](#_Toc4072)

# 1.开发平台与工具

## 1.1godot游戏引擎

Godot游戏引擎是Juan Linietsky [1]和Ariel Manzur [1]发起的，是一款制作游戏的软件，可以制作2D和3D游戏。通过基于节点的架构来设计游戏，3D渲染器设计可以增强3D游戏的画面。具有内置工具的2D游戏功能以像素坐标工作，可以掌控2D游戏效果。面向团队的设计从架构和工具到VCS集成，Godot专为团队中的每个人设计。编辑器可在Windows、Mac OS和Linux系统中运行，支持导出游戏到Windows、Mac OS、Linux、Android、iOS、UWP和HTML5等平台。

# 2.程序设计

## 2.1玩家角色

### 2.1.1描述

玩家角色类的控制代码。

### 2.1.2 编码设计

extends CharacterBody3D

#玩家移动速度

@export var speed = 14

#下落的速度

@export var fall\_acceleration = 75

@export var jump\_impulse = 25

var target\_velocity = Vector3.ZERO

#动画

@onready var anim = $advancedCharacter/AnimationPlayer

#状态

@onready var stats : Node = $Stats

#枪

@onready var gun = $advancedCharacter/blasterB/RayCast3D

#子弹

var bullet

func \_ready():

pass

func \_physics\_process(\_delta):

var direction = Vector3.ZERO

#键盘控制角色

if Input.is\_action\_pressed("move\_right"):

direction.x +=1;

if Input.is\_action\_pressed("move\_left"):

direction.x -=1;

if Input.is\_action\_pressed("move\_back"):

direction.z +=1;

if Input.is\_action\_pressed("move\_forward"):

direction.z -=1;

#跳跃

if is\_on\_floor() and Input.is\_action\_just\_pressed("move\_jump"):

target\_velocity.y = jump\_impulse

#控制向量的长度保持一致

if direction != Vector3.ZERO:

direction = direction.normalized()

$advancedCharacter.look\_at(position - direction,Vector3.UP)

#地面速度

target\_velocity.x = direction.x \* speed

target\_velocity.z = direction.z \* speed

#垂直速度

if not is\_on\_floor():

target\_velocity.y = target\_velocity.y - (fall\_acceleration \* \_delta)

#移动玩家

velocity = target\_velocity

move\_and\_slide()

#控制动画

if Input.is\_action\_just\_pressed("move\_back") or Input.is\_action\_just\_pressed("move\_forward") or Input.is\_action\_just\_pressed("move\_left") or Input.is\_action\_just\_pressed("move\_right"):

anim.play("walk")

if Input.is\_action\_just\_released("move\_back") or Input.is\_action\_just\_released("move\_forward") or Input.is\_action\_just\_released("move\_left") or Input.is\_action\_just\_released("move\_right"):

anim.play("RESET")

if Input.is\_action\_just\_pressed("squat"):

anim.play("squat")

if Input.is\_action\_just\_released("squat"):

anim.play("RESET")

#近身攻击

if Input.is\_action\_just\_pressed("attack"):

anim.play("attack")

if Input.is\_action\_just\_released("attack"):

anim.play("RESET")

#远程攻击

if Input.is\_action\_just\_pressed("shoot"):

anim.play("shoot")

if Input.is\_action\_just\_released("shoot"):

anim.play("RESET")

#受到攻击

func \_on\_hurt\_box\_hurt():

print("受到攻击")

stats.health -= 1

if stats.health == 0:

anim.play("dead")

## 2.2怪物

### 2.2.1 描述

怪物类的控制代码

### 2.2.2 编码设计

extends CharacterBody3D  
  
# 敌人属性  
var health = 100  # 基础血量  
var attack\_power = 10  # 基础攻击力  
var move\_speed = 5  # 移动速度  
var attack\_range = 10  # 攻击范围  
# 受击后自动寻路追击玩家  
func take\_damage(damage):  
if damage > 0:  
print("Enemy took damage:", damage)  
health -= damage  
  
# 如果受到伤害，寻找玩家并追击  
if health < 0:  
get\_tree().add\_node("NavMesh")  # 假设NavMesh是导航网格节点  
var target = get\_node("Player")  # 假设Player是玩家节点  
if target:  
move\_to(target.position)  
  
# 玩家进入攻击范围后自动寻路攻击玩家  
func \_process(delta):  
var player\_node = get\_node("Player")  # 假设Player是玩家节点  
if player\_node and is\_in\_range(player\_node.position, attack\_range):  
if not is\_facing(player\_node.position):  
face(player\_node.position)  # 如果不在攻击范围内，则面向玩家  
attack(player\_node)  # 攻击玩家  
  
# 随机生成敌人  
func \_ready():  
var spawn\_point = Vector3()  # 在某个随机位置生成敌人  
spawn\_point.x = -100 + random(-50, 50)  
spawn\_point.y = -10 + random(-5, 5)  
spawn\_point.z = -100 + random(-50, 50)  
set\_position(spawn\_point)  
print("Enemy spawned at:", spawn\_point)  
  
# 死亡后消失  
func \_on\_Player\_died():  
queue\_free()  # 当玩家死亡时，敌人消失

## 2.3 时间UI设计

### 2.3.1描述

时间UI的控制代码。

### 2.3.2 编码设计

manager：extends Node

@onready var timer = $Timer

func get\_time\_elapsed(): #经过的时间

return timer.wait\_time - timer.time\_left

UI：extends CanvasLayer

@export var time\_manager: Node

@onready var time\_label = $%TimeLabel

func \_process(delta):

if time\_manager == null:

return

var time\_elapsed = time\_manager.get\_time\_elapsed() #获得经过的时间

time\_label.text = format\_seconds\_to\_string(time\_elapsed) #修改时间标签

func format\_seconds\_to\_string(seconds: float):

var minutes = floor(seconds / 60)

var remining\_seconds = seconds - (minutes \* 60)

return str(minutes) + ":" + ("%02d" % floor(remining\_seconds)) #格式化输出时间

## 2.4血量

### 2.4.1描述

玩家角色血量的控制代码。

### 2.4.2编码设计

extends Control

@onready var stats : Node = $%Stats

@onready var health\_label = $CanvasLayer/HealthLabel

func \_process(delta):

health\_label.text = str(stats.health) #显示血量

func \_on\_hurt\_box\_hurt(): #受伤

stats.health -= 1

if stats.health == 0:

return

## 2.5攻击动作

### 2.5.1描述

攻击动作的控制代码。

### 2.5.2编码设计

extends Area3D

signal hit

func \_on\_area\_entered(hurtbox):

emit\_signal("hit")

hurtbox.emit\_signal("hurt")

print("碰撞")

pass

## 2.6角色状态

### 2.6.1描述

角色状态的控制代码。

### 2.6.2编码设计

extends Node

#最大血量

@export var max\_health = 3

#血量状态，默认为最大血量

@onready var health = max\_health :

set(v):

v = clampi(v,0,max\_health)

if health == v:

return

health = v