Sistema de Inventario - Código y Explicación

1. Archivo: `animation_controller.gd`

```
extends AnimatedSprite2D # Usamos AnimatedSprite2D para mostrar animaciones
    bidimensionales
class_name AnimationController # Esta clase se puede referenciar desde otros scripts
    por su nombre

# Diccionario que mapea animaciones de caminar a sus correspondientes animaciones en
    reposo
const MOVEMENT_TO_IDLE = {
      "back_walk": "back_idle",
      "front_walk": "front_idle",
      "right_walk": "right_idle",
      "left_walk": "left_idle"
}

# Reproduce la animacin de movimiento dependiendo de la direccin del vector de
    velocidad
func play_movement_animation(velocity: Vector2):
      if velocity.x > 0:
            play("right_walk")
      elif velocity.x < 0:
            play("left_walk")

      if velocity.y > 0:
            play ("front_walk")
      elif velocity.y < 0:
            play("back_walk")

# Reproduce la animacin en reposo segn la ltima animacin de movimiento
func play_idle_animation():
      if MOVEMENT_TO_IDLE.keys().has(animation):
            play(MOVEMENT_TO_IDLE[animation])
```

Este script define el controlador de animaciones del personaje.

2. Archivo: `inventory.gd`

```
extends Node # Clase base general sin funcionalidad visual
class_name Inventory # Clase global para gestionar el inventario

# Referencia al nodo de UI del inventario (CanvasLayer)
@onready var inventory_ui: CanvasLayer = $"../InventoryUI"

# Arreglo exportado para contener los tems
@export var items: Array[InventoryItem] = []
```

```
# Escucha eventos de entrada, como teclas
func _input(event: InputEvent) -> void:
      if Input.is_action_just_pressed("toggle_inventory"):
            inventory_ui.toggle()

# Aade un tem al inventario, considerando si es apilable
func add_item(item: InventoryItem, stacks: int):
      if stacks && item.max_stacks > 1:
            add_stackable_item_to_inventory(item, stacks)
      else:
            items.append(item)
            inventory_ui.add_item(item)

# Aade tems que se pueden apilar, combinando o creando nuevos stacks
func add_stackable_item_to_inventory(item: InventoryItem, stacks: int):
      var item_index = -1
      for i in items.size():
            if items[i] != null and items[i].name == item.name:
                  item_index = i

      if item_index != -1:
            var inventory_item = items[item_index]
            if inventory_item.stacks + stacks <= item.max_stacks:
                  # Si cabe en el stack actual, solo se suman
                  inventory_item.stacks += stacks
                  items[item_index] = inventory_item
                  inventory_ui.update_stack_at_slot_index(inventory_item.stacks,
                      item_index)
            else:
                  # Si no cabe, se crea un nuevo stack con el sobrante
                  var stacks_diff = inventory_item.stacks + stacks - item.max_stacks
                  var additional_inventory_item = inventory_item.duplicate(true)
                  inventory_item.stacks = item.max_stacks
                  inventory_ui.update_stack_at_slot_index(inventory_item.max_stacks,
                      item_index)
                  additional_inventory_item.stacks = stacks_diff
                  items.append(additional_inventory_item)
                  inventory_ui.add_item(additional_inventory_item)
      else:
            # Si el tem no existe, se agrega como nuevo
            item.stacks = stacks
            items.append(item)
            inventory_ui.add_item(item)
```

Controla el sistema de inventario y cómo se agregan los objetos.

3. Archivo: `inventory_item.gd`

```
extends Resource # Recurso que no es nodo; puede guardarse como archivo

class_name InventoryItem # Clase global para referenciar tems

var stacks = 1 # Cantidad de unidades del tem

# Define el tipo de slot donde puede equiparse
@export_enum("Right_Hand", "Left_Hand", "Potions", "NotEquipable")
var slot_type: String = "NotEquipable"
```

```
# Forma de colisin cuando el tem est en el suelo
@export var ground_collision_shape: RectangleShape2D

# Nombre del tem
@export var name: String = ""

# Textura principal del tem
@export var texture: Texture2D

# Textura lateral (opcional)
@export var side_texture: Texture2D

# Mximo de stacks permitidos para este tem
@export var max_stacks: int

# Precio del tem
@export var price: int
```

Define la clase base de los ítems, con sus propiedades y comportamientos.

4. Archivo: `inventory_ui.gd`

```
extends CanvasLayer # Nodo que se dibuja sobre el resto de la escena (UI)

class_name InventoryUI # Clase global

# Referencia al contenedor de la cuadrcula de slots
@onready var grid_container: GridContainer = %GridContainer

# Escena precargada de un slot de inventario
const INVENTORY_SLOT_SCENE = preload("res://Scenes/UI/iventory_slot.tscn")

# Tamao del inventario (nmero total de slots)
@export var size = 8

# Nmero de columnas en la cuadrcula
@export var columns = 4

func _ready() -> void:
      # Configura el nmero de columnas y crea los slots
      grid_container.columns = columns

      for i in size:
            var inventory_slot = INVENTORY_SLOT_SCENE.instantiate()
            grid_container.add_child(inventory_slot)

# Alterna la visibilidad del inventario
func toggle():
      visible = !visible

# Agrega un tem al primer slot vaco disponible
func add_item(item: InventoryItem):
      var slots = grid_container.get_children().filter(func (slot): return slot.
          is_empty)
      var first_empty_slot = slots.front() as InventorySlot
      first_empty_slot.add_item(item)

# Actualiza la cantidad en un slot especfico
```

```
func update_stack_at_slot_index(stacks_value: int, inventory_slot_index: int):
      if inventory_slot_index == -1:
            return
      var inventory_slot: InventorySlot = grid_container.get_child(
          inventory_slot_index)
      inventory_slot.stacks_label.text = str(stacks_value)
```

Interfaz visual del inventario con lógica para añadir objetos y actualizar cantidades.

5. Archivo: `inventory_slot.gd`

```
extends VBoxContainer # Contenedor vertical que organiza los elementos del slot

class_name InventorySlot
var is_empty = true # Indica si el slot est vaco
var is_selected = false

# Propiedades exportadas para personalizar el slot desde el editor
@export var single_button_press = false
@export var starting_texture: Texture
@export var start_label: String

# Referencias a los subcomponentes del slot
@onready var texture_rect: TextureRect = $NinePatchRect/MenuButton/CenterContainer/
    TextureRect
@onready var name_label: Label = $NameLabel
@onready var stacks_label: Label = $NinePatchRect/StacksLabel
@onready var on_clickbutton: Button = $NinePatchRect/OnClickbutton
@onready var price_label: Label = $PriceLabel
@onready var menu_button: MenuButton = $NinePatchRect/MenuButton

# Slot de destino para equipar el objeto
var slot_to_equip = "NotEquipable"

func _ready() -> void:
      # Inicializa texturas y etiquetas si estn definidas
      if starting_texture != null:
            texture_rect.texture = starting_texture

      if start_label != null:
            name_label.text = start_label

      # Configura botones segn el modo
      menu_button.disabled = single_button_press
      on_clickbutton.disabled = !single_button_press
      on_clickbutton.visible = single_button_press

      # Conecta la opcin del men contextual
      var popup_menu = menu_button.get_popup()
      popup_menu.id_pressed.connect(on_popup_menu_item_pressed)

func on_popup_menu_item_pressed(id: int):
      print_debug(id) # Para depuracin

# Agrega un tem al slot y actualiza sus elementos visuales
func add_item(item: InventoryItem):
      if item.slot_type != "NotEquipable":
            var popup_menu: PopupMenu = menu_button.get_popup()
```

```
            var equip_slot_name_array = item.slot_type.to_lower().split("_")
            var equip_slot_name = " ".join(equip_slot_name_array)
            slot_to_equip = item.slot_type
            popup_menu.set_item_text(0, "Equip to " + equip_slot_name)

      is_empty = false
      menu_button.disabled = false
      texture_rect.texture = item.texture
      name_label.text = item.name
      if item.stacks < 2:
            return
      stacks_label.text = str(item.stacks)
```

Representa un espacio visual del inventario con interacción y menú contextual.

6. Archivo: `pickup_item.gd`

```
extends Area2D # Nodo que detecta colisiones bidimensionales (como para recoger
    objetos)

class_name PickUpItem

# Referencia al tem que representa este objeto recogible
@export var inventory_item: InventoryItem

# Referencias a los nodos grficos y de colisin
@onready var sprite_2d: Sprite2D = $Sprite2D
@onready var collision_shape_2d: CollisionShape2D = $CollisionShape2D

# Cantidad de stacks de este objeto en el mundo
@export var stacks: int = 1

func _ready() -> void:
      # Configura textura y colisin basadas en los datos del tem
      sprite_2d.texture = inventory_item.texture
      collision_shape_2d.shape = inventory_item.ground_collision_shape
```

Objeto que aparece en el mundo y puede recogerse para el inventario.

7. Archivo: `player.gd`

```
extends CharacterBody2D # Nodo para personajes con movimiento fsico en 2D

class_name Player

# Referencias a la animacin y al inventario del jugador
@onready var animated_sprite_2d: AnimationController = $AnimatedSprite2D
@onready var inventory: Inventory = $Inventory

const SPEED = 5000.0 # Velocidad del personaje

# Lgica de movimiento y animacin
func _physics_process(delta: float) -> void:
      var direction = Input.get_vector("left", "right", "up", "down")

      if direction:
            # Movimiento con aceleracin proporcional al delta
```

```
            velocity = direction * SPEED * delta
      else:
            # Suavizado del movimiento al detenerse
            velocity.x = move_toward(velocity.x, 0, SPEED * delta)
            velocity.y = move_toward(velocity.y, 0, SPEED * delta)

      # Cambio de animacin segn velocidad
      if velocity != Vector2.ZERO:
            animated_sprite_2d.play_movement_animation(velocity)
      else:
            animated_sprite_2d.play_idle_animation()

      move_and_slide() # Aplica el movimiento con deslizamiento

# Detecta colisin con un objeto recogible
func _on_area_2d_area_entered(area: Area2D) -> void:
      if area is PickUpItem:
            inventory.add_item(area.inventory_item, area.stacks)
            area.queue_free() # Elimina el objeto del mundo
```

Maneja al jugador, sus movimientos, animaciones y la recolección de objetos.