ADVANCED INVENTORY SYSTEM GUIDE

Creating an inventory system in Godot involves managing a collection of items, creating a user interface (UI) to display them, and implementing logic to handle item interactions. Here is a guide on creating a basic inventory system.

****Please note that this is only pseudocode – please do not try and run this as actual code in Godot!*

1. **Setup Godot Project:**

Make sure you have Godot installed and create a new project if you haven't already.

2. **Create Item Data Structure:**

You should have a way to define the items that can be in the inventory. Create a script for items.

```gd
# Item.gd

extends Resource

class_name Item



export(String) var name

export(Texture) var icon

export(int) var max_stack_size = 1
```

3. **Create Inventory Data Structure:**

You need a place to store the items that the player collects. Create a script to manage the inventory.

```gd
# Inventory.gd

extends Node

class_name Inventory

var items = []



func add_item(item, amount = 1):

    for i in range(items.size()):

        if items[i].item == item and items[i].amount < item.max_stack_size:

            items[i].amount += amount

            return

    items.append({ "item": item, "amount": amount })



func remove_item(item, amount = 1):

    for i in range(items.size()):

        if items[i].item == item:

            items[i].amount -= amount

            if items[i].amount <= 0:
```

```
            items.remove(i)

        return```
```

4. <u>Create Inventory UI:</u>

Use Godot's `Control` nodes to create the UI. For the inventory UI, you might use a `GridContainer` to display item icons in a grid.

<u>5. Create InventoryUI Script:</u>

This script will control the inventory UI. It will handle showing, updating, and hiding the inventory.

```gd
# InventoryUI.gd

extends Control

export(NodePath) var inventory_path

onready var inventory = get_node(inventory_path)

onready var grid_container = $GridContainer



func _ready():

    hide()

    update_inventory()



func toggle_inventory():

    visible = !visible
```

```
func update_inventory():
    grid_container.queue_free()

    for item_info in inventory.items:
        var item_button = Button.new()
        item_button.text = str(item_info.amount)
        item_button.texture_normal = item_info.item.icon
        grid_container.add_child(item_button)
```

5. **Handle Input:**

Implement input handling to toggle the inventory UI.

```gd
# InventoryUI.gd (continued)
func _input(event):
    if event.is_action_pressed("toggle_inventory"):
        toggle_inventory()
```

Make sure you have an input action set up for "toggle_inventory" or use any other input action of your choice.

6. **Populate Inventory for Testing:**

For testing, you can populate the inventory with some items when the game starts.

```gd
# Inventory.gd


func _ready():
    var item = preload("res://Item.gd").new()
    item.name = "Potion"
    item.icon = preload("res://path_to_icon.png")
    item.max_stack_size = 5


    add_item(item, 3)
```

7. **Run and Test:**

Run the game and test your inventory system. Make sure you can open and close the inventory, and that items are displayed correctly.

This is a basic example to get you started. Inventory systems can become very complex depending on the needs of your game. You may need to add more features such as item categories, using items, item descriptions, drag and drop functionality, etc.

8. **Saving and Loading:**

Implement saving and loading the inventory data to keep track across game sessions. You can use Godot's built-in `FileAccess` class or the `JSON` class for saving and loading inventory data.

This is a basic outline of how to set up a inventory system in Godot. Depending on your game, you may need to add more features. The principles will remain the same; create data structures to represent the inventory, manage them efficiently, and display information to the player in an engaging way.