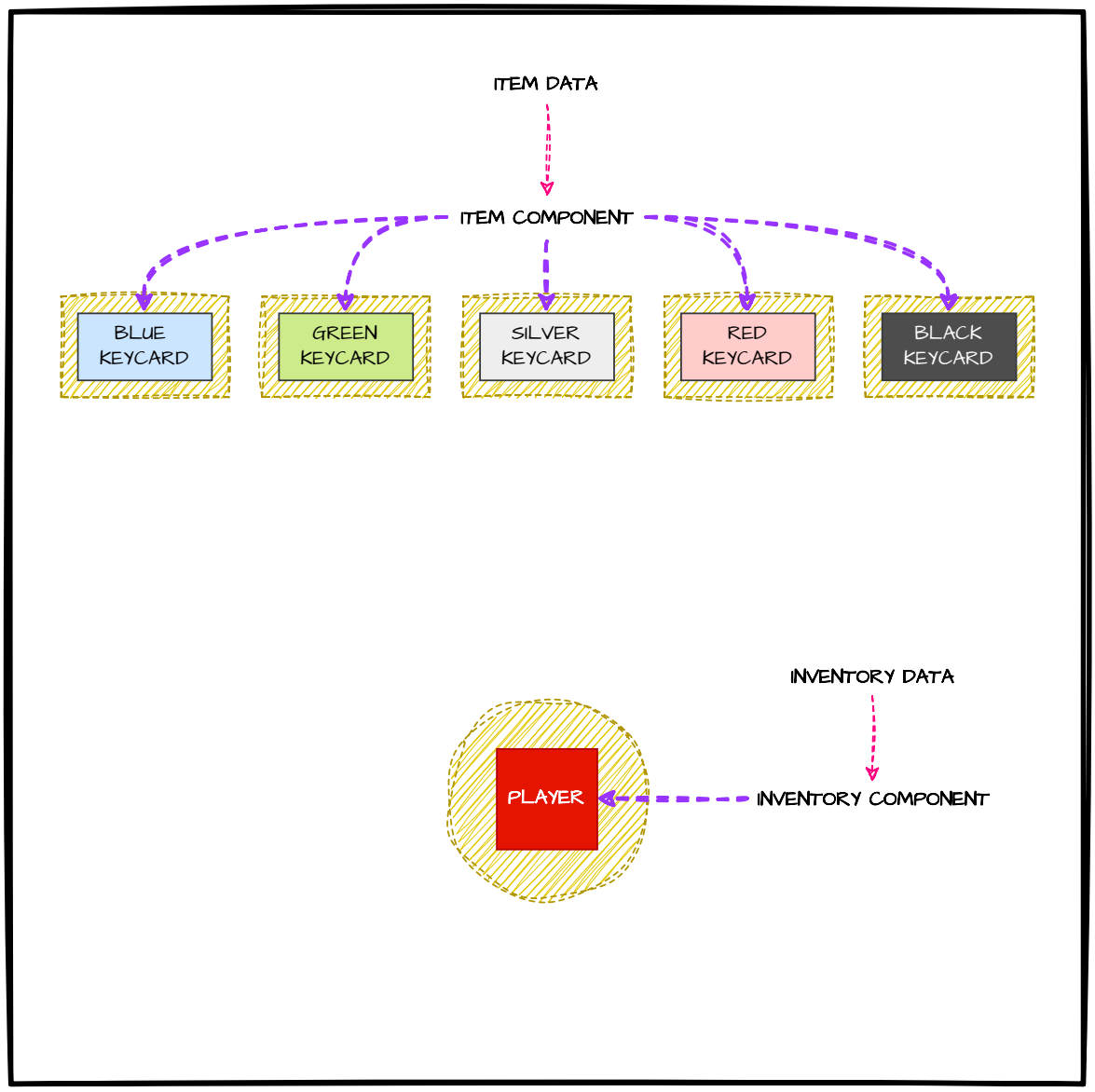
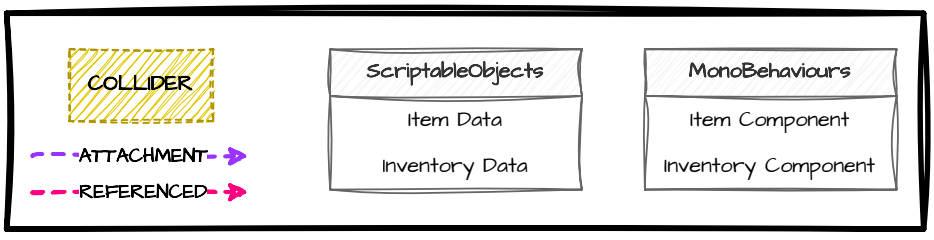
INVENTORY SYSTEM

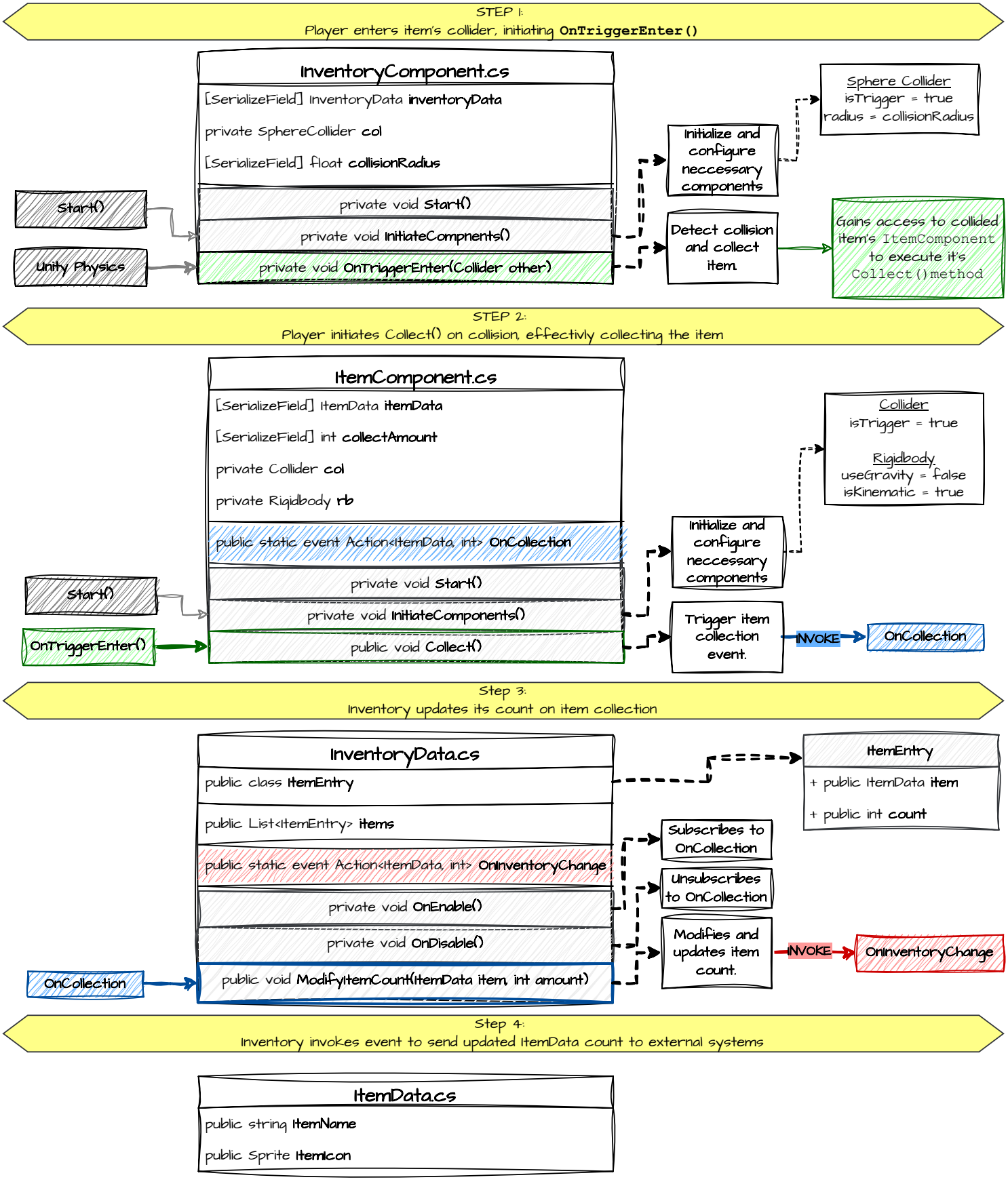
ITEM COLLECTION

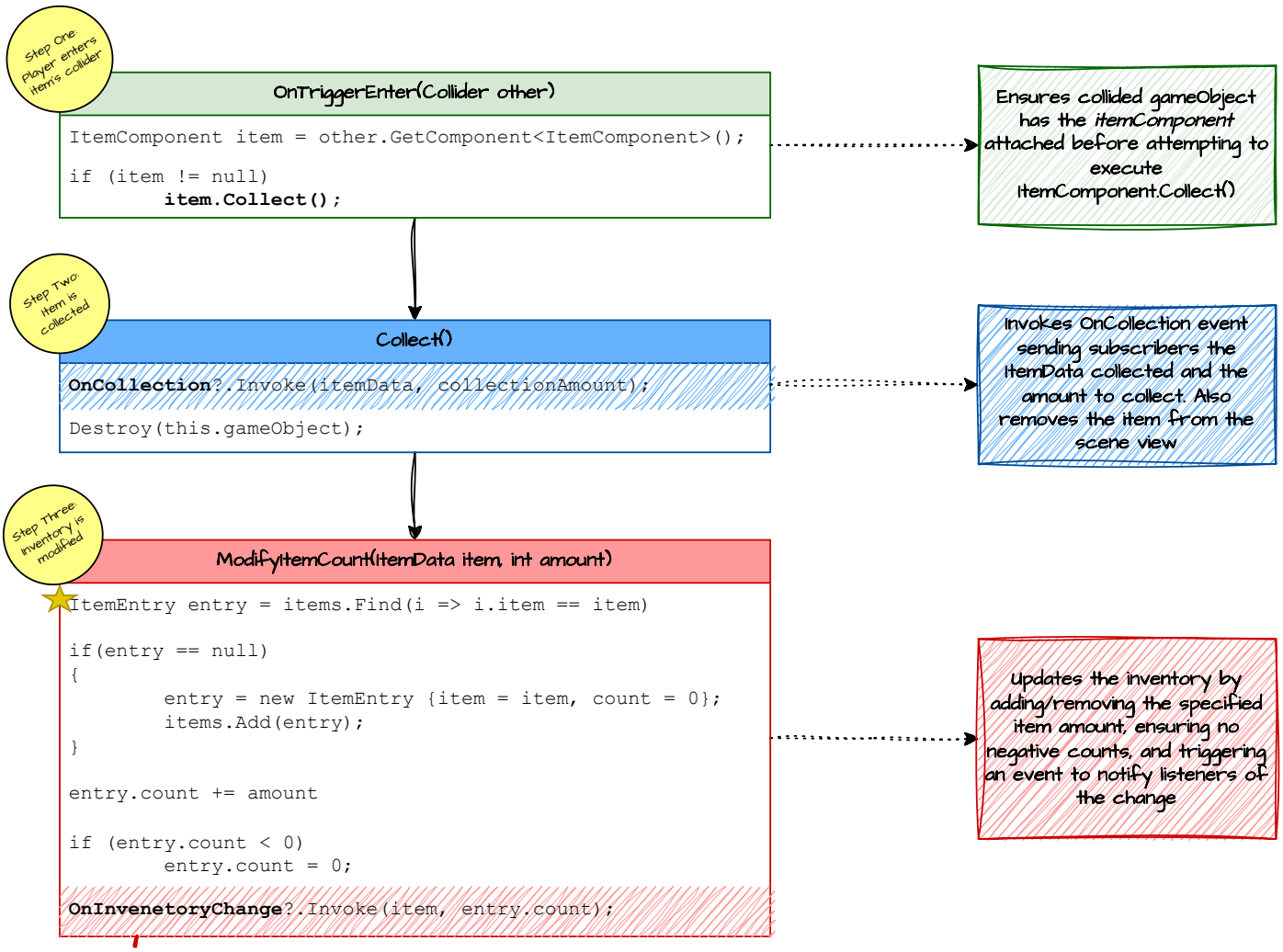


KEY



TASK LIST
Player enters Item's Collider
Item is added to player's Invenetory


STEP 1:
Player enters item's collider, initiating OnTriggerEnter()
InventoryComponent.cs
[SerializeField] InventoryData inventoryData
private SphereCollider col
[SerializeField] float collisionRadius
private void Start()
private void InitiateCompnents()
private void OnTriggerEnter(Collider other)
Start()
Unity Physics
Initialize and configure neccessary components
Sphere Collider
isTrigger = true
radius = collisionRadius
Detect collision and collect item.
Gains access to collided item's ItemComponent to execute it's Collect()method
STEP 2:
Player initiates Collect() on collision, effectivly collecting the item
ItemComponent.cs
[SerializeField] ItemData itemData
[SerializeField] int collectAmount
private Collider col
private Rigidbody rb
public static event Action<ItemData, int> OnCollection
private void Start()
private void InitiateComponents()
public void Collect()
Start()
OnTriggerEnter()
Initialize and configure neccessary components
Collider
isTrigger = true
Rigidbody
useGravity = false
isKinematic = true
Trigger item collection event.
INVOKE
OnCollection
Step 3:
Inventory updates its count on item collection
InventoryData.cs
public class ItemEntry
public List<ItemEntry> items
public static event Action<ItemData, int> OnInventoryChange
private void OnEnable()
private void OnDisable()
public void ModifyItemCount(ItemData item, int amount)
OnCollection
ItemEntry
+ public ItemData item
+ public int count
Subscribes to OnCollection
Unsubscribes to OnCollection
Modifies and updates item count.
INVOKE
OnInventoryChange
Step 4:
Inventory invokes event to send updated ItemData count to external systems
ItemData.cs
public string ItemName
public Sprite ItemIcon



`ItemEntry entry = items.Find(i => i.item == item)`

This line searches for an **`ItemEntry`** inside the **`items`** list where the `item` field matches the specified `item`. If found, it assigns the result to `entry`; otherwise, `entry` will be `null`.

Code Part	Explanation
`ItemEntry entry`	Declares a variable `entry` of type `ItemEntry`
`items`	A list of `ItemEntry` objects (e.g., `List<ItemEntry> items`)
`.Find(...)`	Searches the `items` list for the first element that meets a condition
`(i => i.items == item)`	A **lambda expression** that defines the search condition: "Find an entry where `i.item` (from the list) equals `item` (the searched item)."
`entry =`	Stores the found `ItemEntry` in `entry`, or `null` if not found