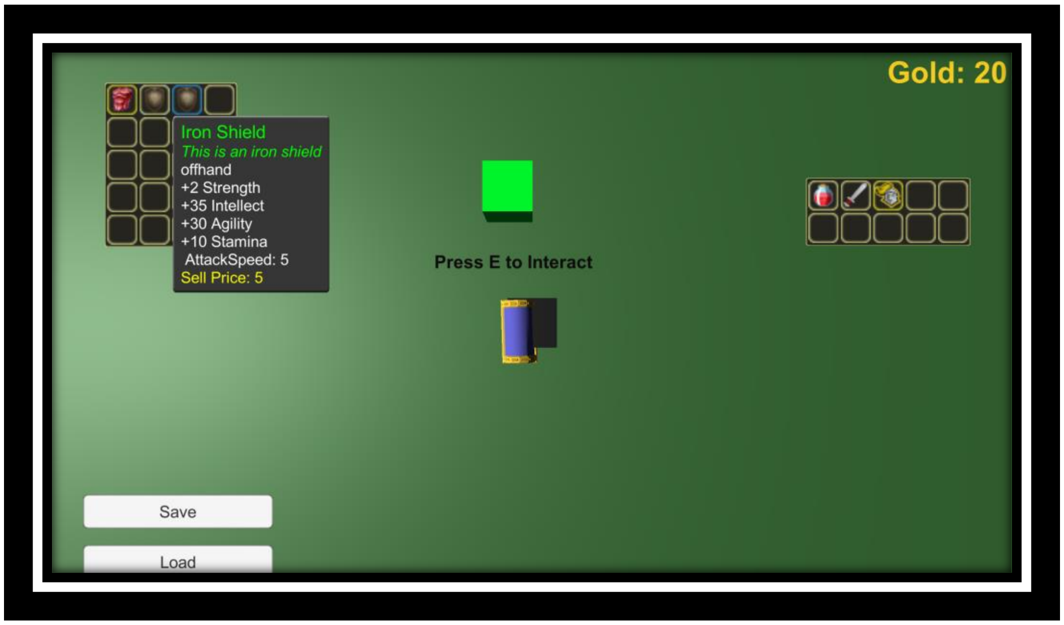
Prototype_Sim_v1

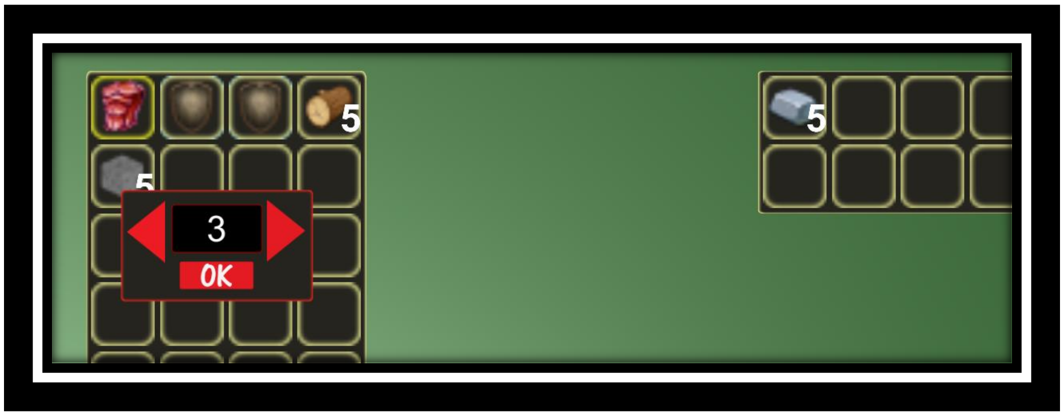
The tasks were very specific, I hit all of them besides one (Show item on player), but I thought to myself that I have to make it as complex as possible. The game logic is fairly simple:

- Player can move around in the mini game world and interact with it; (The image below shows interaction with an “NPC”);



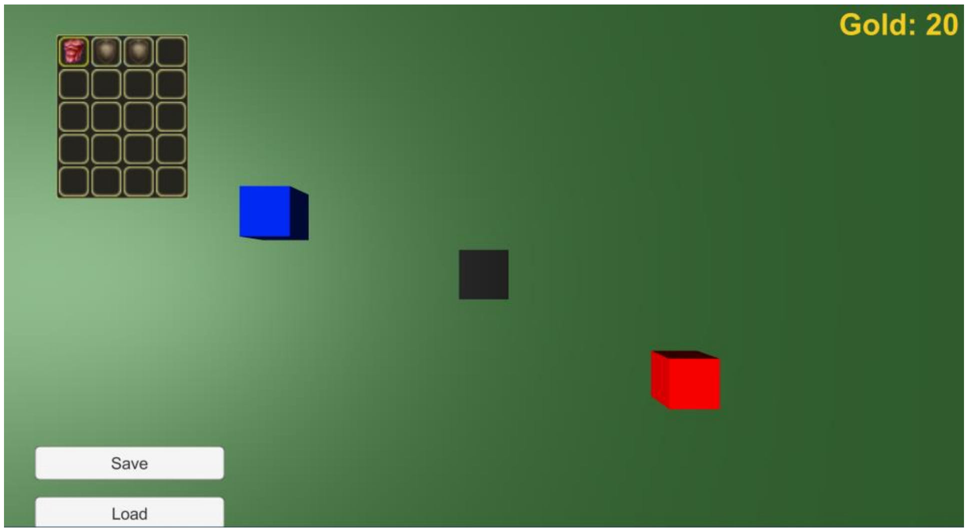
- You can sell and buy items, go get more items (I probably messed up here because I went more towards the RPG style, but hey those can be sims too. Tried to replicate your game a bit)

- The items that you get from specific points in the gameworld are usable, splitable and savable. You can even stockpile them in 2 “wardrobes” that are present in the world;

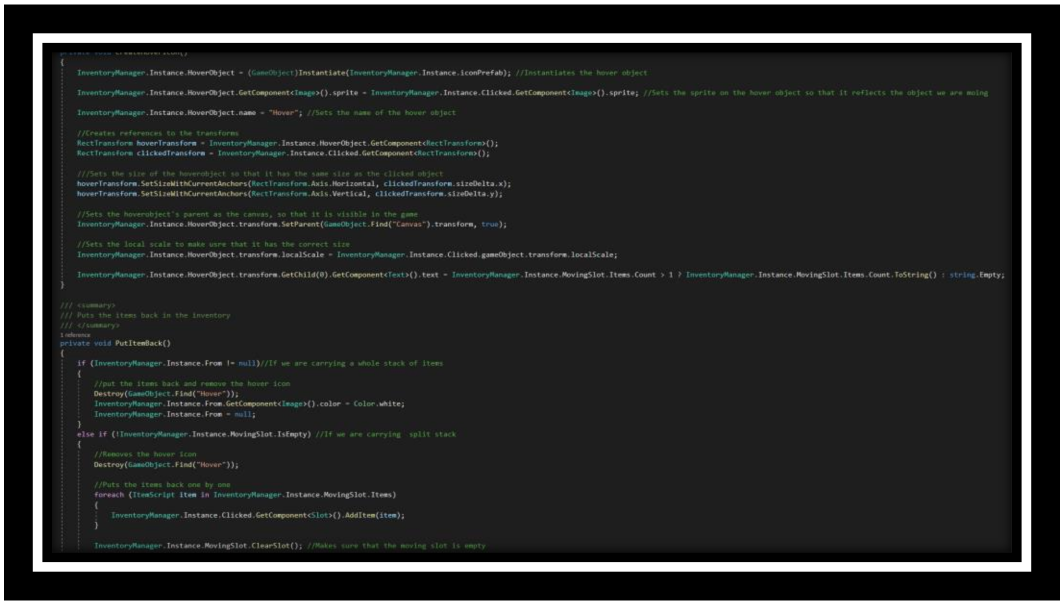


- As presented each item comes with its own functionality, can be equipped (does not show on the player though, IT’S A CUBE) can be used, the materials that you pick up from the “Green Cube” by the “NPC” can be used to craft set items at the “Blue Cube” (secrets to be discovered). The red one spawns random items in your inventory once triggered. The items themselves are coming from a generated XML file (you can find it in the project files). The items can be dropped, merged and equipped;
- The inventory is scripted, the game will build the layouts according to specified values, fairly

simple calculations;



I have separated the project into coverage areas so I can work efficiently and create the logic nicely and also have time to document the scripts in the most detailed way. Every function and every property are going to tell you exactly what it does. I will not complain about the documentation I made,



```
{
    InventoryManager.Instance.HoverObject = (GameObject)Instantiate(InventoryManager.Instance.iconPrefab); //Instantiates the hover object

    InventoryManager.Instance.HoverObject.GetComponent<Image>().sprite = InventoryManager.Instance.Clicked.GetComponent<Image>().sprite; //Sets the sprite on the hover object so that it reflects the object we are moing

    InventoryManager.Instance.HoverObject.name = "Hover"; //Sets the name of the hover object

    //Creates references to the transforms
    RectTransform hoverTransform = InventoryManager.Instance.HoverObject.GetComponent<RectTransform>();
    RectTransform clickedTransform = InventoryManager.Instance.Clicked.GetComponent<RectTransform>();

    ///Sets the size of the hoverobject so that it has the same size as the clicked object
    hoverTransform.SetSizeWithCurrentAnchors(RectTransform.Axis.Horizontal, clickedTransform.sizeDelta.x);
    hoverTransform.SetSizeWithCurrentAnchors(RectTransform.Axis.Vertical, clickedTransform.sizeDelta.y);

    //Sets the hoverobject's parent as the canvas, so that it is visible in the game
    InventoryManager.Instance.HoverObject.transform.SetParent(GameObject.Find("Canvas").transform, true);

    //Sets the local scale to make usre that it has the correct size
    InventoryManager.Instance.HoverObject.transform.localScale = InventoryManager.Instance.Clicked.gameObject.transform.localScale;

    InventoryManager.Instance.HoverObject.transform.GetChild(0).GetComponent<Text>().text = InventoryManager.Instance.MovingSlot.Items.Count > 1 ? InventoryManager.Instance.MovingSlot.Items.Count.ToString() : string.Empty;
}

/// <summary>
/// Puts the items back in the inventory
/// </summary>
1 reference
private void PutItemBack()
{
    if (InventoryManager.Instance.From != null)//If we are carrying a whole stack of items
    {
        //put the items back and remove the hover icon
        Destroy(GameObject.Find("Hover"));
        InventoryManager.Instance.From.GetComponent<Image>().color = Color.white;
        InventoryManager.Instance.From = null;
    }
    else if (!InventoryManager.Instance.MovingSlot.IsEmpty) //If we are carrying  split stack
    {
        //Removes the hover icon
        Destroy(GameObject.Find("Hover"));

        //Puts the items back one by one
        foreach (ItemScript item in InventoryManager.Instance.MovingSlot.Items)
        {
            InventoryManager.Instance.Clicked.GetComponent<Slot>().AddItem(item);
        }

        InventoryManager.Instance.MovingSlot.ClearSlot(); //Makes sure that the moving slot is empty
```

however, I must say, touching the “how good do you think you did”, project wise I think I’ve done over my expectations, but lower than the required exact tasks. I am not good at judging myself but after 50 hours of being awake I think I did an overall good job.

It was a very interesting experience and I think I will expand on it in time. Thank you for this opportunity.