READ ME

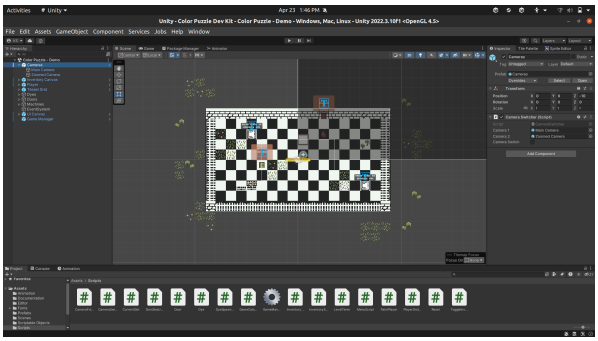
READ ME

All prefabs should also be setup correctly which a few things that can be edited from inside the inspector

All scripts should have tooltips incase you get stuck

Object/Prefab Breakdown

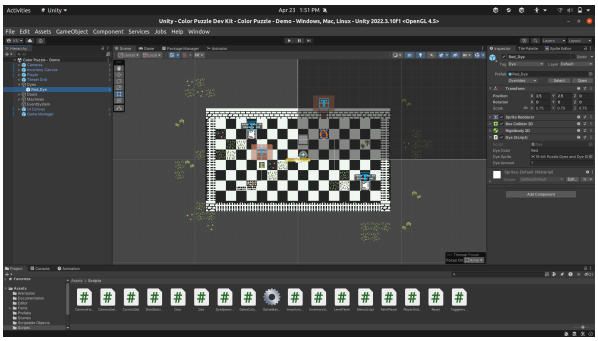
Camera



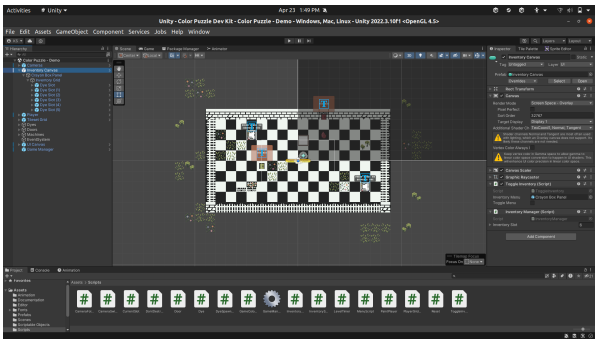
Here the two camera's are inside a parent camera object, and it has a script where the creator can choose which ever camera to be the default one and the switched one. Each of these cameras should also have a camerafollow.cs script, which has no draggable variables

Inventory Canvas

Dye

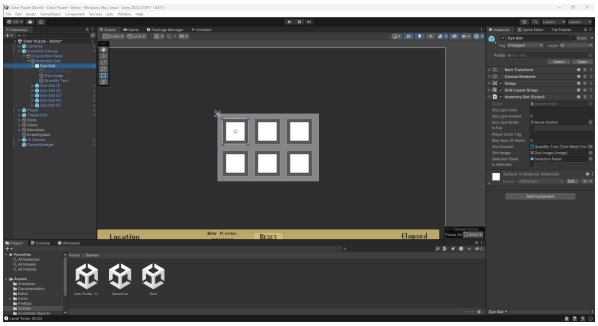


There are dye prefabs for all the colors in the kit, but the dye.cs script can still be edited. These variables are all passed to the inventory when a player picks them up. The dye amount is for the inventory each dye is worth 1 by default when the player picks it up, but this can be changed. The sprite should be set as the same sprite as the object itself. The dye color should be set to the correct color: red, green, blue, etc. but these can be changed but they need to be change in everything else provided by the script, which is not recommended.



Inventory canvas has a inventorymanager.cs script that has an array of all the dye slots inside the inventory grid. This also has a toggleinventory.cs script which should have the Canvas box panel attatched to inventory menu

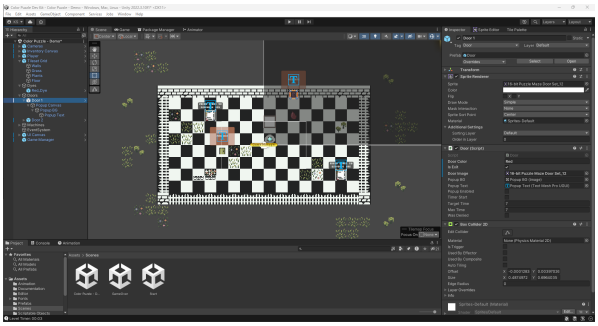
Dye Slot



Dye slots have the inventoryslot.cs script which controls the amount of dyes that can be in each slot (5), the slot text is inside this object and should be dragged to slot amount, the selection panel is also inside this object and should be dragged to selection panel, and dye image should do the same as well

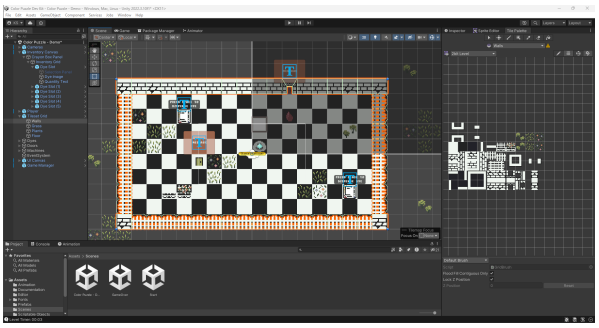
Tilesheet Grid

Door

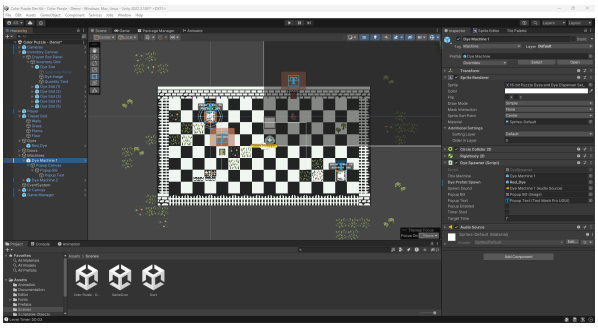


This has a door.cs script which just stores variables for checking if the door color matches with the player. There weren't doors made for every color, but a user could create a bunch for each, if they choose. The door color should be set to the color this door is supposed to check. The boolean isExit should only be check if this door is the exit. The popup text is from the within this prefab just drag over the text to popup text if it somehow got reset. Ignore the timer stuff, that works internally and should not be touched.

Dye Machine

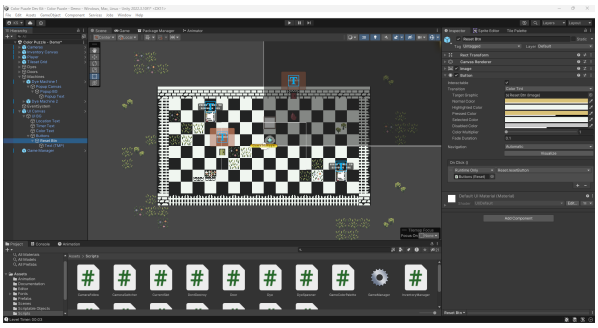


This holds all your tile sheets, which are labeled and ordered correctly. To edit it make sure that you have the tile palette window open by going to window in the top bar → selecting tile sheet → 2D → tile palette. This allows you to use the window and paint/erase the tiles on the correct tile sheet. In this it has the walls active, and it will highlight the walls. You can click on the pencil to paint in new walls, or click on the eraser to erase. I would only use the gate and brick sprites as the wall. Then switch to the plants tile sheet to paint in plant objects you don't want the player to move over. I did include paths but they don't exactly line up very well, but you may choose to have them on the floor, or keep the chess pattern. To remove the chess pattern switch to the floor tile sheet and paint over the whole thing in one of the colors. There is white, black, and dark blue.



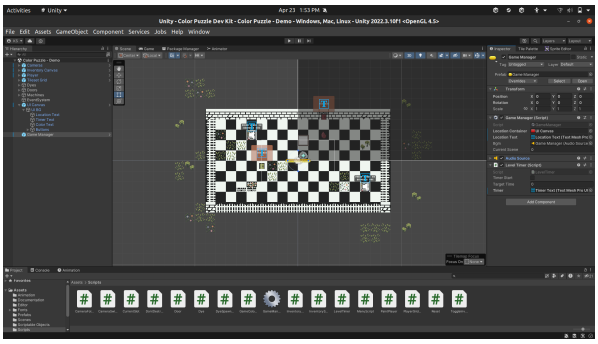
This has a dyespawner.cs script which stores all the stuff needed for it to instantiate a new dye. This machine is just there for the object to be able to grey out when it has been used. Dye prefab should have one of the dye prefabs inside the prefab folder so that the machine can spawn it. Spawn sound is dragged from this object which has an audio source set to the audio from sounds folder. You can change the sound inside this too by dropping down the audio source and replacing the current sound. The popup text is set up the same as the doors. Ignore the timer stuff, that works internally and should not be touched.

UI Canvas



This has no scripts attached except for the button parent object which has reset.cs script. This should be inside the Reset Btn's Onclick() behavior which was dragged from the button parent and select the function under Reset → resetButton ()

Game Manager



This has LevelTimer.cs applied. This should just have the time text from the UI canvas attached to timer. Ignore the other timer stuff, that works internally and should not be touched.

Scripts Breakdown

CameraFollow.cs

This makes the camera follow an object

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;

public class CameraFollow :
MonoBehaviour
{

    [Tooltip("The object th
at the camera needs to foll
ow")]
    [SerializeField] public
```

CameraSwitcher.cs

On a button press, it switches the the current camera

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;

public class CameraSwitcher
: MonoBehaviour
{
    //camera variables

    [Tooltip("The default c
amera")]
    public GameObject Camer
```

```
GameObject target;

    // Update is called onc
e per frame
    void Update()
    {
        //Camera follows ba
sed on the target position
        transform.position
= target.transform.position
+ new Vector3 (0, 1, -10);
    }
}
```

CurrentSlot.cs

This script controls adding items into the mini inventory slot that follows the player

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;
using UnityEngine.EventSyst
ems;
using TMPro;
using UnityEngine.UI;

public class CurrentSlot :
MonoBehaviour
{
    //slot data
    [Tooltip("The dye color
inside this inventory slot,
will change once the player
a1;

    [Tooltip("The secondary
camera")]
    public GameObject Camer
a2;

    [Tooltip("Checks whethe
r or not the cameras have b
een swapped, DO NOT EDIT")]
    public bool cameraSwitc
h = false;

    // Start is called befo
re the first frame update
    void Awake()
    {
        Camera1.SetActive(t
rue);
        Camera2.SetActive(f
alse);
        cameraSwitch = fals
e;
    }

    // Update is called onc
e per frame
    void Update()
    {
        //If P key pressed
switch the perspective
        if(Input.GetKeyDown
(KeyCode.P))
        {
            //if false swit
ch camera 1 with 2
```

```
picks up an item, DO NOT ED
IT")]
    public string slot_dyeC
olor;

    [Tooltip("The dye sprit
e inside this inventory slo
t, will change once the pla
yer picks up an item, DO NO
T EDIT")]
    public Sprite slot_dyeS
prite;

    [Tooltip("The dye color
tag that is passed to the
'Player Paint' script, DO N
OT EDIT")]
    public string playerCol
orTag;

    //the dye slot that sho
ws up and moves with the pl
ayer
    [Tooltip("The current i
mage of the item inside thi
s slot, DO NOT EDIT")]
    [SerializeField] public
Image currentSlotImage;

    public void AddToCurren
tSlot(string name, Sprite s
prite)
    {
        currentSlotImage.en
abled = true;
```

```
            if(cameraSwitch
== false)
            {
                Camera2.Set
Active(true);
                Camera1.Set
Active(false);

                cameraSwitc
h = true;
            }

            //otherwise swi
tch back
            else if(cameraS
witch == true)
            {
                Camera1.Set
Active(true);
                Camera2.Set
Active(false);

                cameraSwitc
h = false;
            }
        }
    }
}
```

DontDestroy.cs

For object persistence when making a full game with levels that need objects to persist across the game

```
        //update name and s
prite
        this.slot_dyeColor
= name;
        tagPlayer(this.slot
_dyeColor);

        this.slot_dyeSprite
= sprite;
        currentSlotImage.sp
rite = sprite;
    }

    //Tagging the player to
pass the color to the paint
Player function
    public void tagPlayer(s
tring colorTag)
    {
        playerColorTag = co
lorTag;
    }
}
```

Door.cs

This controls the door object. Whether or not it’s an exit door and when the player collides whether or not it lets them through based on matching the door color to player color. It enables a popup on a timer when it denies the player acess

```
using System.Collections;
using System.Collections.Ge
```

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;
using UnityEngine.SceneMana
gement;

//Use to make objects in yo
ur first level persist thro
ughout your game
public class DontDestroy :
MonoBehaviour
{
    //array has to be big e
nough to store all the obje
cts that you want to stay t
he same across different sc
enes
    private static GameObje
ct[] persistentObjs = new G
ameObject[10];

    [Tooltip("This current
object's index in the arra
y, 0 is first, 1 is second,
etc.")]
    public int objIndex;

    void Awake()
    {
        //if this array is
empty add the current objec
t
        if(persistentObjs[o
bjIndex] == null)
```

```
neric;
using UnityEngine;
using TMPro;
using UnityEngine.UI;
using UnityEngine.SceneMana
gement;

public class Door : MonoBeh
aviour
{
    [Tooltip("This door's c
olor")]
    public string doorColo
r;

    [Tooltip("Checks whethe
r or not this door is the m
aze exit.")]
    public bool isExit;

    [Tooltip("This door's s
prite, just for referenc
e")]
    [SerializeField] privat
e Sprite doorImage;

    //Ref. to the current s
lot script
    [Tooltip("The player's
mini inventory slot")]
    CurrentSlot currentSlo
t;

    //Door text popup
    [Tooltip("The Popup Bac
kground image object")]
        {
            persistentObjs
[objIndex] = gameObject;
            //don't destroy
when the scene loads
            DontDestroyOnLo
ad(gameObject);
        }

        //if there is a dup
licate game object in the a
rray, delete it
        else if (persistent
Objs[objIndex] != gameObjec
t)
        {
            Destroy(gameObj
ect);
        }
    }
}
```

Dye.cs

Controls the dye. The dye data stored which is for printing to the inventory slot. As well as the collision with the player so they can collect them

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;

public class Dye : MonoBeha
viour
```

```
    public Image popupBG;
    [Tooltip("The Popup tex
t object")]
    public TMP_Text popupTe
xt;
    [Tooltip("Checks whethe
r or not the popup text is
showing, DO NOT EDIT")]
    public bool popupEnable
d;

    //Popup timer
    [Tooltip("Checks whethe
r or not the timer started,
DO NOT EDIT")]
    public bool timerStart
= false;

    [Tooltip("The amount of
seconds the time is going t
o count down, DO NOT EDI
T")]
    [SerializeField] privat
e float targetTime = 7.0f;
//time in seconds

    [Tooltip("The amount of
seconds the time is going t
o reset, DO NOT EDIT")]
    [SerializeField] privat
e float maxTime = 7.0f;

    [Tooltip("Checks whethe
r or not this door denied t
he player, DO NOT EDIT.")]
    public bool wasDenied;
```

```
{

    [Tooltip("The color of
this dye")]
    [SerializeField] privat
e string dyeColor;

    [Tooltip("The correct s
prite for this dye color")]
    [SerializeField] privat
e Sprite dyeSprite;

    [Tooltip("The amount th
is dye object gives when th
e player picks it up")]
    [SerializeField] privat
e int dyeAmount;

    //Ref. to inventory man
ager
    private InventoryManage
r inventoryManager;

    // Start is called befo
re the first frame update
    void Start()
    {
        //find the inventor
y manager code inside the c
anvas
        inventoryManager =
GameObject.Find("Inventory
Canvas").GetComponent<Inven
toryManager>();
    }
```

```
    // Start is called befo
re the first frame update
    void Start()
    {
        popupBG.enabled = f
alse;
        popupText.enabled =
false;

        currentSlot = GameO
bject.Find("Current Dye").G
etComponent<CurrentSlot>();
    }

    void Update()
    {
        //checks if the tim
er has started and tracks i
ts progress
        if(timerStart)
        {
            //start countdo
wn
            targetTime -= T
ime.deltaTime;

            if(targetTime <
= 0.0f)
            {
                Debug.Log
("Popup Timer has finishe
d");
                timerStart
= false;
```

```
    void OnCollisionEnter2D
(Collision2D collision)
    {
        //collision with pl
ayer adds the dye to the in
ventory
        if(collision.gameOb
ject.tag == "Player")
        {
            //calculate and
return an int
            int leftOverIte
ms = inventoryManager.AddDy
e(dyeColor, dyeSprite, dyeA
mount);

            if(leftOverItem
s <= 0)
                Destroy(gam
eObject);
            else
                dyeAmount =
leftOverItems;
        }
    }
}
```

DyeSpawner.cs

This is for the vending machine to control when to give the player a single dye on collision and button press. It also deactivates its collision and goes grey on use. Add item and enable everything when a Dye is collected and added into the inventory

```
                //disable t
he text popup
                disablePopu
p();
                targetTime
= maxTime;
            }
        }
    }

    //enable the text popup
    public void enablePopup
()
    {
        //enable bool
        popupEnabled = tru
e;
        //enable text bg
        popupBG.enabled = t
rue;

        //update the text
        popupText.text = "Y
ou can't go through here wi
thout the right dye";
        //enable texts
        popupText.enabled =
true;

        Debug.Log("Door tex
t is visible");

        //start the popup t
imer
        timerStart = true;
    }
```

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;
using TMPro;
using UnityEngine.UI;

public class DyeSpawner : M
onoBehaviour
{
    [Tooltip("This dye mach
ine object")]
    public GameObject thisM
achine;

    [Tooltip("A dye prefab
object from the prefab fold
er")]
    [SerializeField] public
GameObject dyePrefabSpawn;

    [Tooltip("The sound sou
rce within this dye machine
object")]
    public AudioSource spaw
nSound;

    //Machine text popup
    [Tooltip("The Popup Bac
kground image object")]
    public Image popupBG;
    [Tooltip("The Popup tex
t object")]
    public TMP_Text popupTe
xt;
```

```
    //disable the text popu
p
    public void disablePopu
p()
    {
        //disable everythin
g
        popupBG.enabled = f
alse;
        popupText.enabled =
false;
        popupEnabled = fals
e;

        Debug.Log("Door tex
t is no longer visible");

        wasDenied = false;
    }

    void OnCollisionEnter2D
(Collision2D collision)
    {
        //checks for collis
ion with player
        if(collision.gameOb
ject.tag == "Player")
        {
            //checks if the
player's tag is the same co
lor as the door color
            if(currentSlot.
playerColorTag == doorColor
&& isExit)
            {
```

```
    [Tooltip("Checks whethe
r or not the popup text is
showing, DO NOT EDIT")]
    public bool popupEnable
d;

    //Popup timer
    [Tooltip("Checks whethe
r or not the timer started,
DO NOT EDIT")]
    public bool timerStart
= false;

    [Tooltip("The amount of
seconds the time is going t
o count down, DO NOT EDI
T")]
    [SerializeField] privat
e float targetTime = 7.0f;
//time in seconds

    void Start()
    {
        popupBG.enabled = f
alse;
        popupText.enabled =
false;

        spawnSound = GetCom
ponent<AudioSource>();
    }

    void Update()
    {
        //checks if the tim
er has started and tracks i
```

```
                wasDenied = false;
                Destroy(gameObject);
                Debug.Log("Door was unlocked");

                //change scene to game over screen
                SceneManager.LoadScene("GameOver");
            }
            //if it matches, but not the exit door
            else if(currentSlot.playerColorTag == doorColor && !isExit)
            {
                wasDenied = false;
                Destroy(gameObject);
                Debug.Log("Door was unlocked");
            }
            //if they don't match, show a popup
            else if(currentSlot.playerColorTag != doorColor || (currentSlot.playerColorTag != doorColor && isExit))
            {
                wasDenied = true;
                enablePopup
```

```
ts progress
        if(timerStart)
        {
            //start countdown
            targetTime -= Time.deltaTime;

            if(targetTime <= 0.0f)
            {
                Debug.Log("Popup Timer has finished");
                timerStart = false;

                //disable the text popup
                disablePopup();
            }
        }
    }

    public void InstantiateDye(Vector3 spawnPosition)
    {
        //play sound
        spawnSound.Play(0);

        //Spawn dye
        GameObject dyeGameObj = Instantiate(dyePrefabSpawn, spawnPosition, Quaternion.identity);
```

```
();

                Debug.Log
("Door cannot be unlocke
d");
            }
        }
    }
}
```

GameColorPalette.cs

A scriptable object that stores an array of colors used for the game

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;

[CreateAssetMenu(fileName =
"GameColorPalette", menuNam
e = "GameColorPalette", ord
er = 0)]
public class GameColorPalet
te : ScriptableObject
{
    [System.Serializable]
    public class Entry
    {
        public string name;
        public Color color;
    }

    public List<Entry> pale
tte = new List<Entry>();
```

```
        //Add force so the
items slide into a random d
irection when they spawn
        float dropForce = 5
f;
        Vector2 dropDirecti
on = new Vector2(0, Random.
Range(1.5f, 5.5f));

        dyeGameObj.GetCompo
nent<Rigidbody2D>().AddForc
e(dropDirection * dropForc
e, ForceMode2D.Impulse);
    }

    //enable the text popup
    public void enablePopup
()
    {
        //enable bool
        popupEnabled = tru
e;
        //enable text bg
        popupBG.enabled = t
rue;

        //update the text
        popupText.text = "P
ress Space to Dispense Dy
e.";
        //enable texts
        popupText.enabled =
true;

        Debug.Log("Vending
```

```
    public Color GetColor(s
tring name)
    {
        var entry = palett
e.Find(c => c.name == nam
e);
        if (entry != null)
            return entry.co
lor;

        return Color.white;
    }
}

//source:https://gamedev.st
ackexchange.com/questions/1
67014/how-can-i-use-the-col
ours-of-a-swatch-in-a-scrip
t
```

GameManger.cs

Controls all the UI texts to display the player's location, the elapsed time

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneMana
gement;
using TMPro;

public class GameManager :
Machine text is visible");

        //start the popup t
imer
        timerStart = true;
    }

    //disable the text popu
p
    public void disablePopu
p()
    {
        //disable everythin
g
        popupBG.enabled = f
alse;
        popupText.enabled =
false;
        popupEnabled = fals
e;

        Debug.Log("Vending
Machine text is no longer v
isible");
    }

    void OnCollisionStay2D
(Collision2D collision)
    {
        if(collision.gameOb
ject.tag == "Player")
        {
            enablePopup();

            if(Input.GetKey
Down("space"))
```

```
MonoBehaviour
{

    [Tooltip("The location
text canvas")]
    public GameObject locat
ionContainer;

    [Tooltip("The location
text")]
    public TMP_Text locatio
nText;

    [Tooltip("This object's
audio source")]
    public AudioSource bgm;

    //notation for adding m
ore scenes into the array:
new string[4] {"B1", "F1",
"F2", "A1"};
    //adjust the number ins
ide new string[_] to the co
rrect array length if you a
dd more levels
    string[] floors = new s
tring[1] {"Demo"};
    //should be offset by o
ne since it doesn't count s
tart/game over scenes
    /*
        In Build settings:
        SceneIndex 1 = Dem
o, [0]
        SceneIndex 2 = [1]
        SceneIndex 3 = [2]
            {
                Instantiate
Dye(transform.position);

                //ignore co
lliders after spawning dye
                Physics2D.I
gnoreCollision(gameObject.G
etComponent<Collider2D>(),
GetComponent<Collider2D>
());
                gameObject.
GetComponent<Collider2D>().
isTrigger = true;

                //greyscale
the machine so it looks dea
ctivated
                thisMachin
e.GetComponent<SpriteRender
er>().color = Color.grey;
            }
        }
    }
}
```

InventoryManager.cs

Holds all the inventory slots in an array and also adds the dye to the slot. It also controls the clicking and selection

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;
```

```
        SceneIndex 4 = [3]
    */
    [SerializeField] public
int currentScene = 0;

    void Start()
    {
        bgm = GetComponent<
AudioSource>();

        locationContainer =
GameObject.Find("UI Canva
s");

        //makes sure the te
xt is set
        if(locationText ==
null)
        {
            //find the cont
ainer for the text and set
the text by getting from th
e container
            locationText =
locationContainer.GetCompon
ent<TMP_Text>();
        }

        currentScene = 0;
        checkScene();
    }

    void Update()
    {
        checkScene();
        //uncomment for add
```

```
public class InventoryManag
er : MonoBehaviour
{
    //Ref. inventory slot s
cript from an array of inve
ntory slots
    [Tooltip("An array of a
ll inventory slots, Drag al
l/any new slots from the sc
ene under inventory inside
this dropdown")]
    public InventorySlot[]
inventorySlot;

    public int AddDye(strin
g color, Sprite sprite, int
amount)
    {
        Debug.Log("Dye_Colo
r: " + color + "\n" + "Amou
nt: " + amount + "\n"  +
"Dye_Sprite: " + sprite);

        for(int i = 0; i <
inventorySlot.Length; i++)
        {
            //not full and
the names match or there is
no quantity
            if(inventorySlo
t[i].isFull == false && inv
entorySlot[i].slot_dyeColor
== color || inventorySlot
[i].slot_dyeAmount == 0)
            {
```

```
ing background music
        //playBGM();

        //quit application
any time
        if (Input.GetKey(Ke
yCode.Escape))
        {
            Application.Qui
t();
        }
    }

    //forces unity to set t
he text objects because it
resets every scene change
    public void setTextObje
cts()
    {
        locationContainer =
GameObject.Find("Location T
ext Canvas");

        if(locationText ==
null)
        {
            locationText =
locationContainer.GetCompon
ent<TMP_Text>();
        }
    }

    //for playing backgroun
d music, make sure this obj
ect has an audio source
    public void playBGM()
                int leftOve
rItems = inventorySlot[i].A
ddItem(color, amount, sprit
e);
                //check the
number of leftovers
                if(leftOver
Items > 0)
                    leftOve
rItems = AddDye(color, spri
te, leftOverItems);

                return left
OverItems;
            }
        }

        return amount;
    }

    public void DeselectAll
Slots()
    {
        for(int i = 0; i <
inventorySlot.Length; i++)
        {
            inventorySlot
[i].selectedPanel.SetActive
(false);
            inventorySlot
[i].isSelected = false;
        }
    }
}
```

```
    {
        //while the scenes
stay on the dungeon levels
play the background music
        while(currentScene
<= 0 && currentScene >= 3)
        {
            //Play sound an
d animation
            bgm.Play(0);
        }
    }

    //check what level the
player is on
    public void checkScene
()
    {
        //check the current
scene name and change to th
e next one
        //names should matc
h the scene names in the sc
enes folder
        if(SceneManager.Get
ActiveScene().name == "Colo
r Puzzle - Demo")
        {
            currentScene =
0;
            updateLocation
(currentScene);
            setTextObjects
();
        }
        //uncomment for add
```

InventorySlot.cs

Controls the individual slots within the inventory. Clears the slot when there's nothing there anymore. On click the dye is added to the mini player inventory slot.

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;
using UnityEngine.EventSyst
ems;
using TMPro;
using UnityEngine.UI;

public class InventorySlot
: MonoBehaviour, IPointerCl
ickHandler
{
    //dye data
    [Tooltip("The dye color
inside this inventory slot,
will change once the player
picks up an item, DO NOT ED
IT")]
    [SerializeField] public
string slot_dyeColor;

    [Tooltip("The dye amoun
t inside this inventory slo
t, will change once the pla
yer picks up an item, DO NO
T EDIT")]
    [SerializeField] public
```

```
ing new scenes to the arra
y, make sure add these scen
es in the build settings
        /*else if(SceneMana
ger.GetActiveScene().name =
= "Level  One")
        {
            currentScene =
1;
            updateLocation
(currentScene);
            setTextObjects
();
        }*/
    }

    //function to update th
e location text when the sc
ene changes to the next lev
el
    public void updateLocat
ion(int floorIndex)
    {
        locationText.text =
"You are Here: " + floors[f
loorIndex];
    }
}
```

LevelTimer.cs

This displays a timer for the elapsed time in a level

```
using System.Collections;
using System.Collections.Ge
```

```
int slot_dyeAmount;

    [Tooltip("The dye sprit
e inside this inventory slo
t, will change once the pla
yer picks up an item, DO NO
T EDIT")]
    [SerializeField] public
Sprite slot_dyeSprite;

    [Tooltip("Checks whethe
r or not this slot is ful
l")]
    [SerializeField] public
bool isFull;

    [Tooltip("The dye color
tag that is passed to the
'Player Paint' script, DO N
OT EDIT")]
    [SerializeField] public
string playerColorTag;

    [Tooltip("Sets the maxi
mum amount of items allowed
in this slot")]
    [SerializeField] privat
e int maxNumOfItems;

    //dye slot
    [Tooltip("The 'Quantity
Text' text object that will
display the number of items
in this slot")]
    [SerializeField] privat
e TMP_Text slotAmount;
```

```
neric;
using UnityEngine;
using TMPro;

public class LevelTimer : M
onoBehaviour
{
    [Tooltip("Checks whethe
r or not the timer started,
DO NOT EDIT")]
    public bool timerStart
= false;
    [SerializeField] privat
e float targetTime = 0.0f;
//time in seconds

    [Tooltip("The timer tex
t object")]
    public TMP_Text timer;

    // Start is called befo
re the first frame update
    void Start()
    {
        timerStart = true;
    }

    // Update is called onc
e per frame
    void FixedUpdate()
    {
        //prints timer prog
rams
        if(timerStart)
        {
            //convert the t
```

```
    [Tooltip("The 'Dye' ima
ge object that will display
the image of the items in t
his slot")]
    [SerializeField] privat
e Image slotImage;

    [Tooltip("The inventory
panel that the inventory is
displayed on")]
    public GameObject selec
tedPanel;

    [Tooltip("Checks whethe
r or not a current item is
selected, DO NOT EDIT")]
    public bool isSelected;

    //Ref. to inventory man
ager
    private InventoryManage
r inventoryManager;

    //Ref. to player's mini
inventory
    private CurrentSlot cur
rentSlot;

    private void Start()
    {
        inventoryManager =
GameObject.Find("Inventory
Canvas").GetComponent<Inven
toryManager>();
```

```
ime to an int
            int seconds =
(int)targetTime;

            displayTimer(ta
rgetTime);

            if(seconds >= 3
00)
            {
                Debug.Log
("Timer's up! ");
                timerStart
= false;
            }
        }
    }

    void Update()
    {
        //checks if the tim
er has started and tracks i
ts progress
        if(timerStart)
        {
            //start countdo
wn
            targetTime += T
ime.deltaTime;

            if(targetTime >
= 300.0f)
            {
                Debug.Log
("Timer has finished");
                timerStart
```

```
        currentSlot = GameO
bject.Find("Current Dye").G
etComponent<CurrentSlot>();
    }

    public int AddItem(stri
ng name, int amount, Sprite
sprite)
    {
        slotImage.enabled =
true;

        //check to see if t
he slot is full
        if(isFull)
            return amount;

        //update name and s
prite
        this.slot_dyeColor
= name;
        this.slot_dyeSprite
= sprite;
        slotImage.sprite =
sprite;

        //update amount
        this.slot_dyeAmount
+= amount;
        //when the amount i
s over the max
        if(this.slot_dyeAmo
unt >= maxNumOfItems)
        {
            slotAmount.text
= maxNumOfItems.ToString();
```

```
= false;
            }
        }
    }

    void displayTimer(float
timeToDisplay)
    {
        timer.enabled = tru
e;

        float minutes = Mat
hf.FloorToInt(timeToDisplay
/ 60);
        float seconds = Mat
hf.FloorToInt(timeToDisplay
% 60);

        string timeOut = st
ring.Format("{0:00}:{1:0
0}", minutes, seconds);

        timer.text = "Elaps
e Time: " + timeOut;
        Debug.Log("Level Ti
mer: " + timeOut);
    }
}
```

MenuScript.cs

Controls all the menu buttons on start/gameover screens. Quit, Demo,Start, and Retry buttons.

```
            slotAmount.enab
led = true;
            isFull = true;

            //return leftov
ers
            int extraItems
= this.slot_dyeAmount - max
NumOfItems;
            this.slot_dyeAm
ount = maxNumOfItems;
            return extraIte
ms;
        }

        //update text witho
ut the amount calculations
        slotAmount.text = t
his.slot_dyeAmount.ToString
();
        slotAmount.enabled
= true;

        //when there are no
leftovers
        return 0;
    }

    //Key press event
    public void OnPointerCl
ick(PointerEventData eventD
ata)
    {
        if(eventData.button
== PointerEventData.InputBu
tton.Left)
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MenuScript : MonoBehaviour
{
    // Play button is clicked
    public void DemoGameButton()
    {
        SceneManager.LoadScene("Color Puzzle - Demo");
    }

    public void StartGameButton()
    {
        SceneManager.LoadScene("Level One");
    }

    //Retry Button is clicked after Game Over
    public void RetryGameButton()
    {
        //could be set to load the last completed level
        SceneManager.LoadSc
```

```
        {
            OnLeftClick();
        }
    }

    public void OnLeftClick()
    {
        if(isSelected)
        {
            //push info to the mini slot
            currentSlot.AddToCurrentSlot(slot_dyeColor, slot_dyeSprite);

            this.slot_dyeAmount -= 1;
            slotAmount.text = this.slot_dyeAmount.ToString();

            //when the slot becomes 0
            if(this.slot_dyeAmount <= 0)
            {
                this.slot_dyeAmount = 0;
                EmptySlot();
            }
        }
        else
        {
            //deselects pre
```

```
ene("Color Puzzle - Demo");
    }

    // Exit button is click
ed
    public void ExitGameBut
ton()
    {
        Application.Quit();
    }
}
```

PlayerGridMovement

Allows the player to move in a chess-like movement

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;

public class PlayerGridMove
ment : MonoBehaviour
{

    [Tooltip("The player's
movement speed")]
    [SerializeField] privat
e float moveSpeed = 5f;

    [Tooltip("Object the pl
ayer follows.")]
    public Transform movePo
inter;
```

```
vious slot and switches
            inventoryManage
r.DeselectAllSlots();
            selectedPanel.S
etActive(true);
            isSelected = tr
ue;
        }
    }

    public void EmptySlot()
    {
        slotAmount.enabled
= false;

        slotImage.enabled =
false;
        slotImage.sprite =
null;
    }
}
```

PaintPlayer.cs

Controls when the player needs to be painted once a dye is added to the mini slot

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;
using TMPro;

public class PaintPlayer :
MonoBehaviour
```

```
    [Tooltip("Handles all c
ollision on the collider ma
sk")]
    public LayerMask collid
erMask;

    [Tooltip("Player animat
or")]
    public Animator anim;

    //Ref. door script from
an array of doors
    [Tooltip("An array of a
ll doors slots, Drag all do
ors from the scene inside t
his dropdown")]
    public Door[] doors;

    void Start()
    {
        //transform is no l
onger a child of the player
        movePointer.parent
= null;
    }

    void Update()
    {
        gridMovement();
    }

    public void gridMovemen
t()
    {
        Vector3 horzMovemen
t = new Vector3(Input.GetAx
```

```
{
    [Tooltip("The color tex
t object")]
    public TMP_Text colorTe
xt;

    [Tooltip("The player's
sprite renderer")]
    public SpriteRenderer P
layer;

    [Tooltip("The player ob
ject")]
    public GameObject playe
rObject;

    [Tooltip("The player's
default sprite")]
    public Sprite playerDef
ault;

    //Ref. to the current s
lot script
    [Tooltip("The player's
mini inventory slot")]
    CurrentSlot currentSlo
t;

    //Ref. to the player sp
rite that while change
    [Tooltip("Array for all
the player's sprite in the
same order as the colors in
side the scriptable object
color palette array")]
    [SerializeField] Sprite
```

```
isRaw("Horizontal"), 0f, 0
f);
        Vector3 vertMovemen
t = new Vector3(0f, Input.G
etAxisRaw("Vertical"), 0f);

        //player moves towa
rds the move point
        transform.position
= Vector3.MoveTowards(trans
form.position, movePointer.
position, moveSpeed * Time.
deltaTime);

        //makes sure the pl
ayer is near the move point
in order to move
        if(Vector3.Distance
(transform.position, movePo
inter.position) <= .05f)
        {
            if(Mathf.Abs(In
put.GetAxisRaw("Horizonta
l")) == 1f)
            {
                //in every
door in the array, check if
the wasDenied variable is a
ctive, if so trigger the pl
ay knockback
                for(int i =
0; i < doors.Length; i++)
                {
                    //if th
e door denied the player do
the knockback here
```

```
[] playerSprites;

    [Tooltip("The player's
color picker color, DO NOT
EDIT")]
    [SerializeField] Color
printColor;

    //Ref. to the Color scr
iptable object
    [Tooltip("The Game Colo
r Palette Scriptable Objec
t, which stores all the col
ors for easy access")]
    public GameColorPalette
colors_db;

    [Tooltip("Checks whethe
r or not the player has cha
nged colors, DO NOT EDIT")]
    public bool iscolorSwap
ped;

    void Start()
    {
        colorText.enabled =
false;
        currentSlot = GameO
bject.Find("Current Dye").G
etComponent<CurrentSlot>();
    }

    void Update()
    {
        //check the selecte
d dye and swap player sprit
```

```
                    if(doors[i].wasDenied)
                    {
                        doors[i].enablePopup();
                        movePointer.position -= horzMovement;
                    }
                }

                //checks if the player is colliding with layer mask before allowing the movement, else get pushed back
                if(!Physics2D.OverlapCircle(movePointer.position + horzMovement, .25f, colliderMask))
                    movePointer.position += horzMovement;
                else
                    movePointer.position -= horzMovement;
            }
            //else if, to prevent diagonal movement
            else if(Mathf.Abs(Input.GetAxisRaw("Vertical")) == 1f)
            {
                for(int i = 0; i < doors.Length; i++)
```

```
e
        CheckAndSwap();
    }

    //changing the player's colors
    public void CheckAndSwap()
    {
        //count should correspond with the index of the player sprite array:
        //red color matches red player sprite, which both are index 0
        int count = 0;

        if(currentSlot.playerColorTag == "Red")
        {
            var color = colors_db.GetColor("Red");
            //swapSprite(count);
            swapSprite(count, color);
        }
        else if(currentSlot.playerColorTag == "Orange")
        {
            count = 1;
            var color = colors_db.GetColor("Orange");
            //swapSprite(count);
```

```
                {
                    //if th
e door denied the player do
the knockback here
                    if(door
s[i].wasDenied)
                    {
                        doo
rs[i].enablePopup();
                        mov
ePointer.position -= vertMo
vement;
                    }
                }

                if(!Physics
2D.OverlapCircle(movePointe
r.position + vertMovement,
.25f, colliderMask))
                    movePoi
nter.position += vertMoveme
nt;
                else
                    movePoi
nter.position -= vertMoveme
nt;
            }

            //uncomment bel
ow to add proper movement a
nimation
            //anim.SetBool
("moving", false);
        }

        //uncomment below t
```

```
            swapSprite(coun
t, color);
        }
        else if(currentSlo
t.playerColorTag == "Yello
w")
        {
            count = 2;
            var color = col
ors_db.GetColor("Yellow");
            //swapSprite(co
unt);
            swapSprite(coun
t, color);
        }
        else if(currentSlo
t.playerColorTag == "Gree
n")
        {
            count = 3;
            var color = col
ors_db.GetColor("Green");
            //swapSprite(co
unt);
            swapSprite(coun
t, color);
        }
        else if(currentSlo
t.playerColorTag == "Aqua")
        {
            count = 4;
            var color = col
ors_db.GetColor("Aqua");
            //swapSprite(co
unt);
            swapSprite(coun
```

```
o add proper movement anima
tion
        /*else
            anim.SetBool("m
oving", true);*/
    }
}
```

Reset.cs

For the reset button

```
using System.Collections;
using System.Collections.Ge
neric;
using UnityEngine;
using UnityEngine.SceneMana
gement;

public class Reset : MonoBe
haviour
{
    //function to reset the
square's color
    public void resetButton
()
    {
        //Get the current s
cene
        Scene scene = Scene
Manager.GetActiveScene();

        //Reload the curren
t scene
        SceneManager.LoadSc
ene(scene.name);
t, color);
        }
        else if(currentSlo
t.playerColorTag == "Blue")
        {
            count = 5;
            var color = col
ors_db.GetColor("Blue");
            //swapSprite(co
unt);
            swapSprite(coun
t, color);
        }
        else if(currentSlo
t.playerColorTag == "Purpl
e")
        {
            count = 6;
            var color = col
ors_db.GetColor("Purple");
            //swapSprite(co
unt);
            swapSprite(coun
t, color);
        }
        else if(currentSlo
t.playerColorTag == "Pink")
        {
            count = 7;
            var color = col
ors_db.GetColor("Pink");
            //swapSprite(co
unt);
            swapSprite(coun
t, color);
        }
```

```
    }
}
```

ToggleInventory.cs

On a button press activate the inventory panel so that it can be displayed to the player. This also pauses the game.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ToggleInventory : MonoBehaviour
{

    [Tooltip("The Inventory Menu panel object")]
    public GameObject InventoryMenu;

    [Tooltip("Checks whether or not the menu is open, DO NOT EDIT")]
    public bool toggleMenu;

    // Start is called before the first frame update
    void Start()
    {
        InventoryMenu.SetActive(false);
        toggleMenu = false;
        //make sure that sprite has a default state
        else
        {
            Player.sprite = playerDefault;
        }
    }

    //Function to change the sprite
    public void swapSprite(int index, Color color)
    {
        printColor = color;
        colorText.color = color;

        iscolorSwapped = true;

        //index should correspond with the index of the palette database: red color, matches red player sprite
        Player.sprite = playerSprites[index];

        if(iscolorSwapped)
        {
            Debug.Log("The player is now a different color!" + "\n" + "Color: " + ColorUtility.ToHtmlStringRGB(printColor));
```

```
        Time.timeScale = 1;
    }

    // Update is called once per frame
    void Update()
    {
        if(Input.GetKeyDown("i") && !toggleMenu)
        {
            InventoryMenu.SetActive(true);
            toggleMenu = true;

            Debug.Log("Menu Opened");

            //pause
            Time.timeScale = 0;
        }
        else if(Input.GetKeyDown("i") && toggleMenu)
        {
            //unpause
            Time.timeScale = 1;

            InventoryMenu.SetActive(false);
            toggleMenu = false;

            Debug.Log("Menu Closed");
```

```
            colorText.text = "Now playing: " + "\n" + ColorUtility.ToHtmlStringRGB(printColor);
        }

        colorText.enabled = true;
    }
}
```

```
            }
        }
    }
```