The Tales of Welda Tutorial

Table of Contents

[Setting up Unity 2](#_Toc478024404)

[Download the assets 2](#_Toc478024405)

[Use the Assets 2](#_Toc478024406)

[Setting up colliders 2](#_Toc478024407)

[Scripts 3](#_Toc478024408)

[Player Movement and Camera Movement 3](#_Toc478024409)

[Making the Inventory Interface 6](#_Toc478024410)

[Implementing the Inventory System 14](#_Toc478024411)

[Event Trigger Script(s) 16](#_Toc478024412)

[Dead Screen 17](#_Toc478024413)

[Main Menu Script 17](#_Toc478024414)

[Show scroll, and open menu script 18](#_Toc478024415)

[Hey where is the rest of the scripts? 19](#_Toc478024416)

[Audio 19](#_Toc478024417)

[Main screen 20](#_Toc478024418)

[Conclusion 21](#_Toc478024419)

# Setting up Unity

First we need to set up the Unity IDE

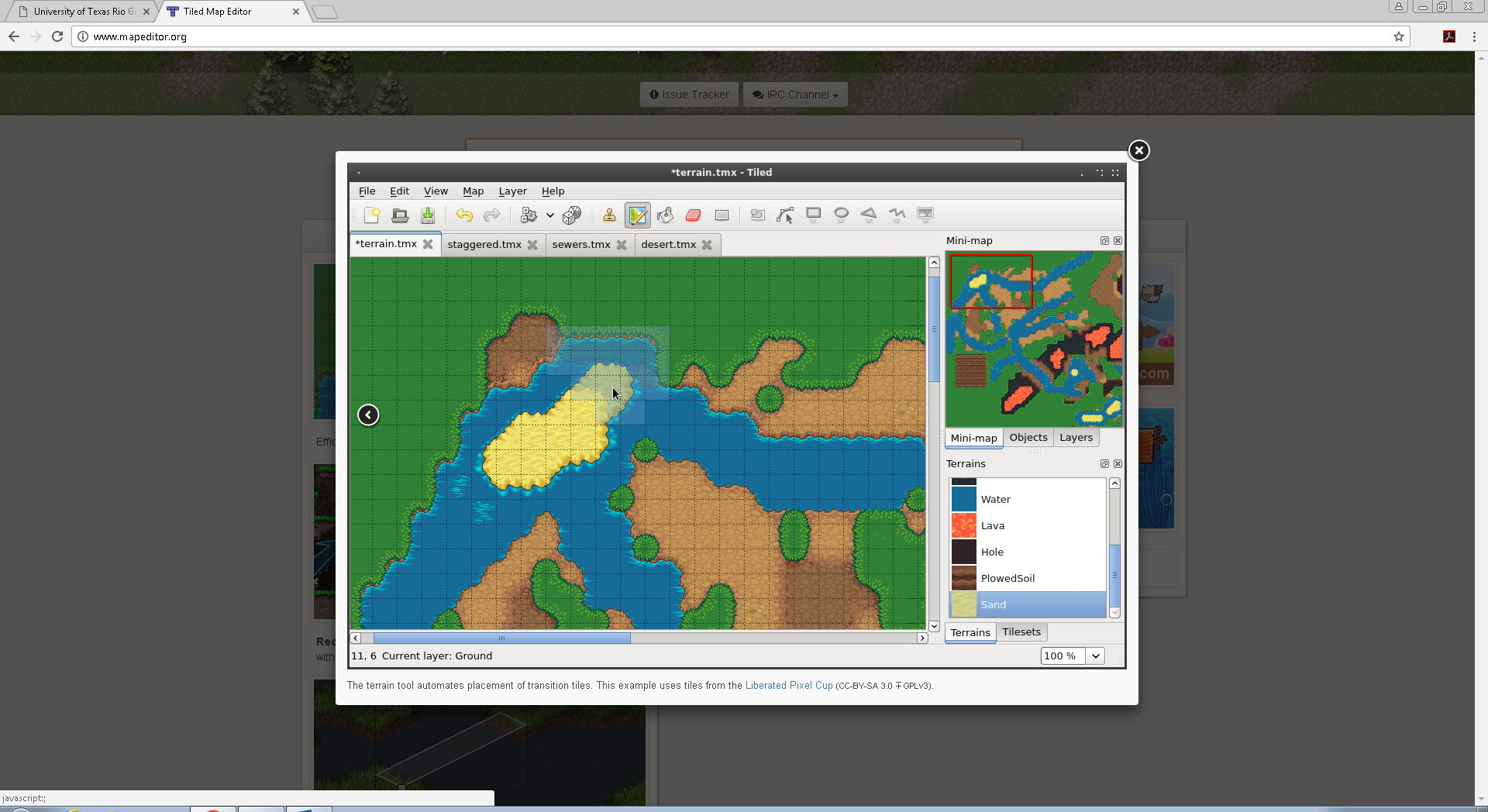
## Download the assets

Launch Unity and create a 2D project. Then download the sprite asset pack from [here](https://drive.google.com/drive/folders/0B-6tbB7_VGCkSUx2SG1udUF3T2s?usp=sharing) and drag it into your asset folder.

## Creating the Assets

Tile assets were created in Adobe Photoshop CS 6. Maps were created using the freeware software Tiled.





## Use the Assets

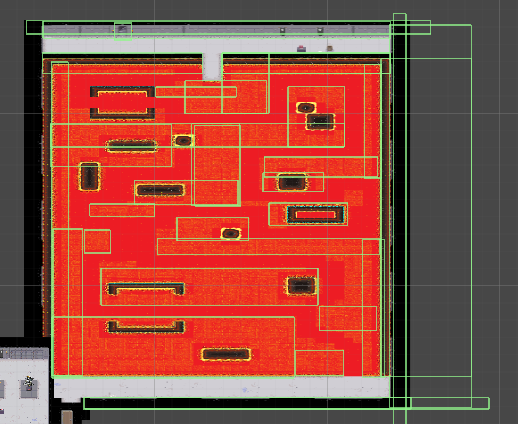
Once you have the assets on your IDE, make a new scene and create an empty game object called ENVIRONMENT.

On that environment drag all the map sprites, and arrange them as you please

Tip: change to the 2 by 3 layout view to see the game view and the scene view more easily.

# Setting up colliders

This game has a lot of colliders. A lot of them, I mean it.

Setting colliders is easy and tedious at the same time because our maps were made finished, instead of individual pieces. Because of that we need to draw the colliders instead of grabbing a specific game object and automatically setting it as a collider. The best way to set up the colliders is to draw a collide on anything that the player is not supposed to walk through. That includes walls, chests, statues, etc.

Once you have made the coliders, make sure you create them as a child of the room theyre in. that way if you need to resize the rooms you wont need to redo all the colliders

# Scripts

The level of writing in this game is to the bare minimum. We did not use any complex formula or algorithm to solve the problems, we used pretty basic scripting skills.

Here are the scripts:

## Player Movement and Camera Movement

In order to move the player with the mouse please copy this script and add it to your character.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.EventSystems;

public class PlayerMovementWithClick : MonoBehaviour {

private Vector3 targetPos;

private Vector3 oldposition;

// private int keepTrack = 0;

public new AudioClip audio;

public GameObject player;

void Start()

{

oldposition = transform.position;

targetPos = transform.position;

GetComponent<AudioSource>().clip = audio;

GetComponent<AudioSource>().playOnAwake = false;

}

void Update()

{

if (Input.GetMouseButton(0))

{

targetPos = Camera.main.ScreenToWorldPoint(Input.mousePosition);

targetPos.z = transform.position.z;

if (transform.position != targetPos)

{

GetComponent<AudioSource>().Play();

}

GetComponent<AudioSource>().Play();

}

transform.position = Vector3.MoveTowards(transform.position, targetPos, 3 \* Time.deltaTime);

if(transform.position == targetPos)

{

GetComponent<AudioSource>().Stop();

}

}

}

This will allow the character to move where ever you click on the mouse.

For the camera movement, you just need to copy this script and add it to the main camera.

using UnityEngine;

public class CameraFollow : MonoBehaviour {

public Transform target;

public float cameraSpeed = 5.0f;

Vector3 offset;

// Use this for initialization

void Start () {

offset = transform.position - target.position;

}

// Update is called once per frame

void FixedUpdate () {

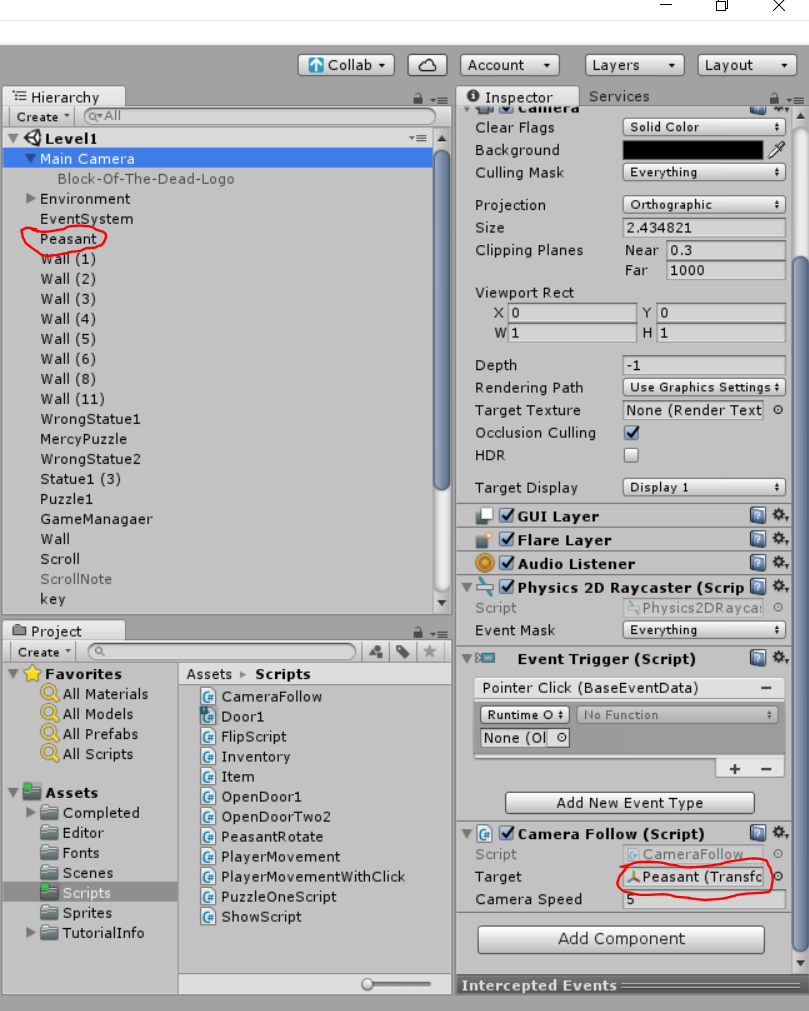
Vector3 camPos = target.position + offset;

transform.position = Vector3.Lerp(transform.position, camPos, cameraSpeed \* Time.deltaTime);

}

}

Once added you will then, through the inspector, add the peasant (name of our character) to the target field.



To make sure your character rotates and faces the position you clicked; you will need to add the following script your character. using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class FlipScript : MonoBehaviour

{

float \_posX;

private Vector2 moveLocation;

private Vector3 mouseRotation;

private Vector3 mousePosition;

private Vector2 currentLocation;

// Use this for initialization

void Start()

{

}

void Update()

{

if (Input.GetMouseButtonDown(0) || Input.GetMouseButton(0))

{

mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);

moveLocation = new Vector2(mousePosition.x, mousePosition.y);

currentLocation = new Vector2(transform.position.x, transform.position.y);

if (moveLocation.x <= currentLocation.x)

{

transform.eulerAngles = new Vector3(0, 180, 0); // Normal

}

else

{

transform.eulerAngles = new Vector3(0, 0, 0); // Flipped

}

}

}

}

## Making the Inventory Interface

Before we begin making the interface you must add this script to your scripts folder.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

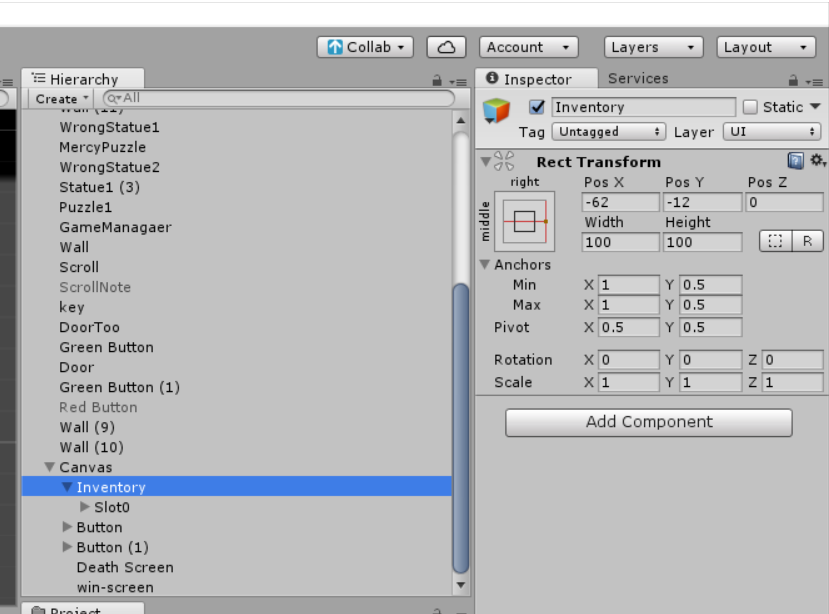
[CreateAssetMenu]

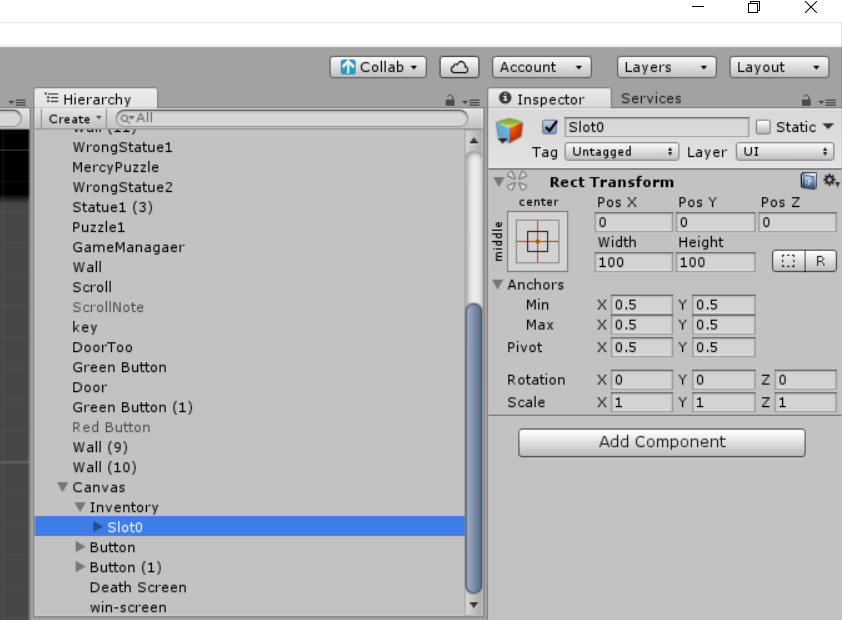
public class Item : ScriptableObject {

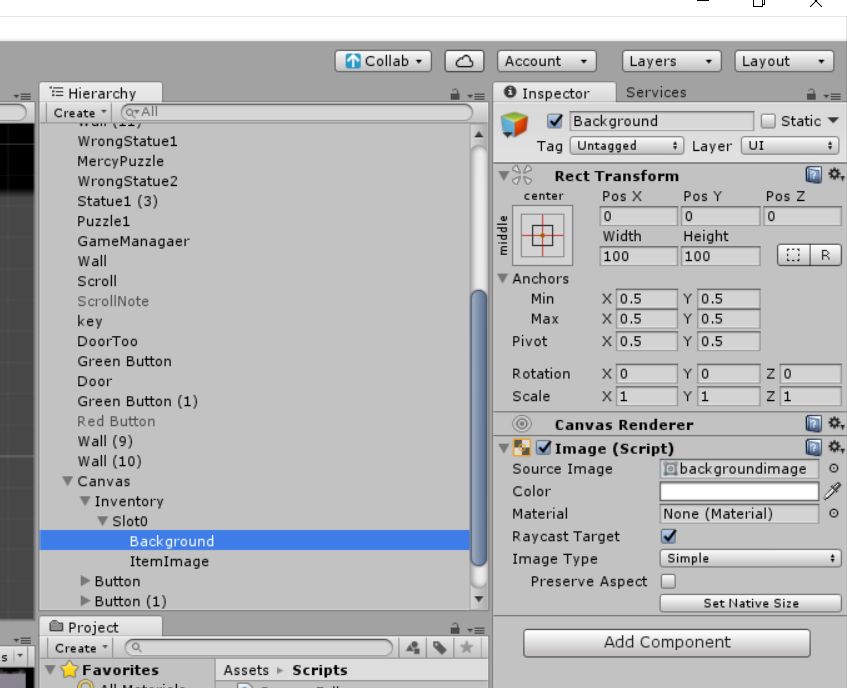
public Sprite sprite;

}

To make the inventory interface of the project you will need to make a canvas and within that canvas create and empty game object. Rename the GameObject as Inventory or any name you wish. You will right click the Inventory game object and create an empty game object. Rename that game object to Slot0, or any name you want. Below are the screen shot of how it should look and settings for each game object you made.



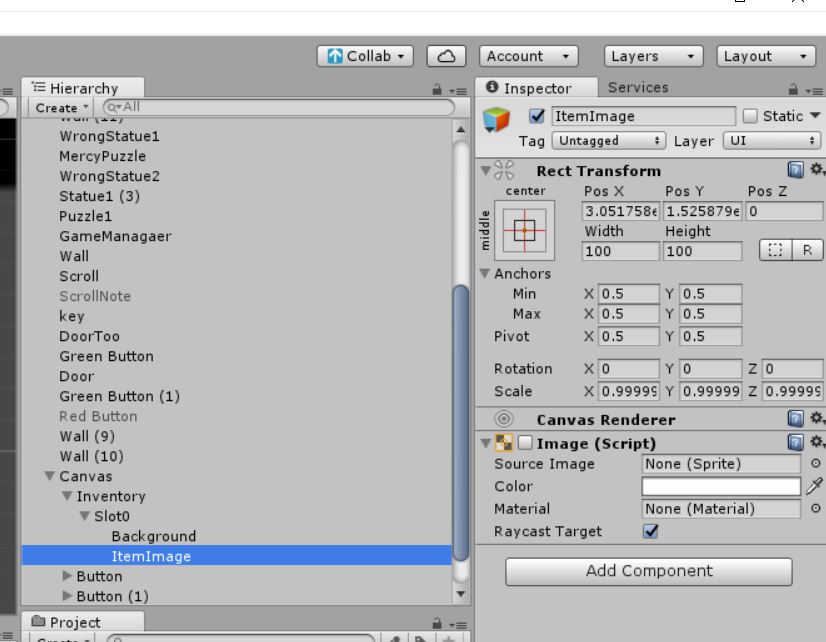
You will right click slot0 look for UI and select image. You will create two of these and name on background image and itemImage. Below are the setting for both image game objects.



For source image, I just looked for a gray square on google and added it there. You can add whichever you like. Once added you should see this little square in your game view.



Below is the settings for itemImage



You will copy and paste the script below and add it to the canvas.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Inventory : MonoBehaviour

{

public Image[] itemImages = new Image[numItemSlots];

public Item[] items = new Item[numItemSlots];

public const int numItemSlots = 1;

public void AddItem(Item itemToAdd)

{

for (int i = 0; i < items.Length; i++)

{

if (items[i] == null)

{

items[i] = itemToAdd;

itemImages[i].sprite = itemToAdd.sprite;

itemImages[i].enabled = true;

return;

}

}

}

public void RemoveItem(Item itemToRemove)

{

for (int i = 0; i < items.Length; i++)

{

if (items[i] == itemToRemove)

{

items[i] = null;

itemImages[i].sprite = null;

itemImages[i].enabled = false;

return;

}

}

}

public bool hasKey(Item key)

{

for(int i = 0; i < items.Length; i++)

{

if (key == items[i])

return true;

}

return false;

}

}

Once that is done you will create a new folder under assets named editor and paste this script to it.

using UnityEngine;

using UnityEditor;

[CustomEditor(typeof(Inventory))]

public class InventoryEditor : Editor

{

private bool[] showItemSlots = new bool[Inventory.numItemSlots];

private SerializedProperty itemImagesProperty;

private SerializedProperty itemsProperty;

private const string inventoryPropItemImagesName = "itemImages";

private const string inventoryPropItemsName = "items";

private void OnEnable()

{

itemImagesProperty = serializedObject.FindProperty(inventoryPropItemImagesName);

itemsProperty = serializedObject.FindProperty(inventoryPropItemsName);

}

public override void OnInspectorGUI()

{

serializedObject.Update();

for (int i = 0; i < Inventory.numItemSlots; i++)

{

ItemSlotGUI(i);

}

serializedObject.ApplyModifiedProperties();

}

private void ItemSlotGUI(int index)

{

EditorGUILayout.BeginVertical(GUI.skin.box);

EditorGUI.indentLevel++;

showItemSlots[index] = EditorGUILayout.Foldout(showItemSlots[index], "Item slot " + index);

if (showItemSlots[index])

{

EditorGUILayout.PropertyField(itemImagesProperty.GetArrayElementAtIndex(index));

EditorGUILayout.PropertyField(itemsProperty.GetArrayElementAtIndex(index));

}

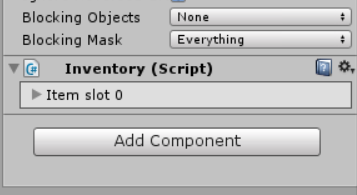
EditorGUI.indentLevel--;

EditorGUILayout.EndVertical();

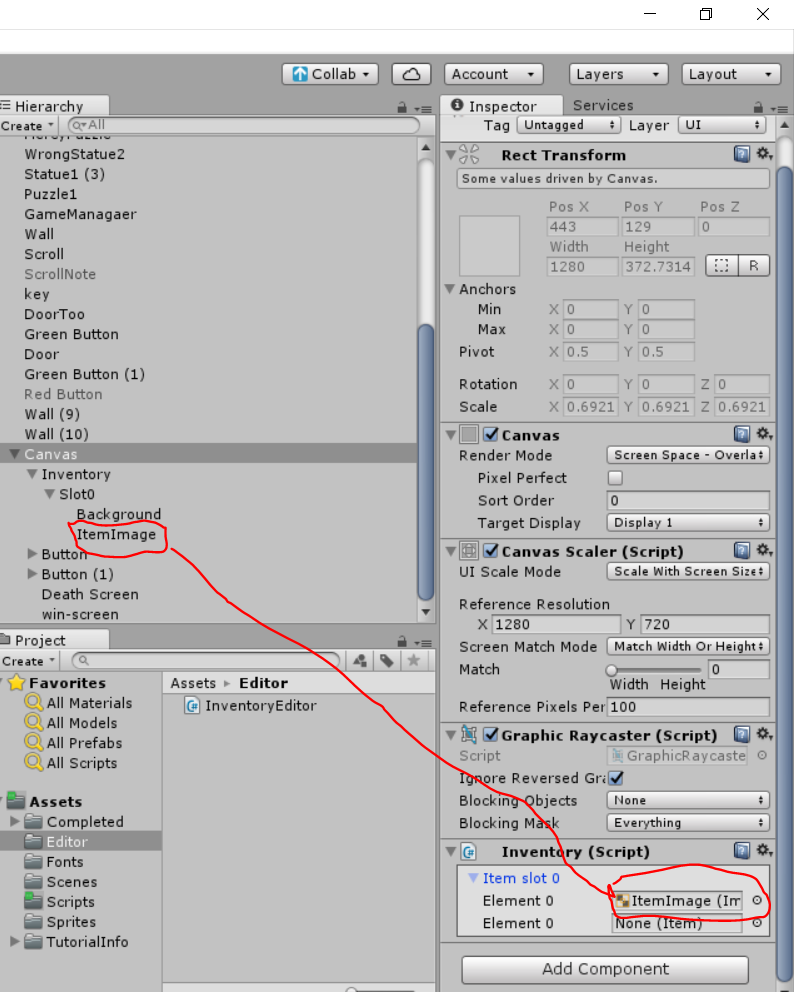
}

}

Now go to the canvas on the inspector and your inventory script section should look like this.



Expand that slot and add the itemImage game object to the first element0. As shown below.



Once that is done your inventory is now ready.

## Implementing the Inventory System

Modify the opendoor1 script attached to the key by overwriting the code with the one below.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class OpenDoor1 : MonoBehaviour {

public GameObject Key;

public GameObject door1;

private Inventory inv;

public new AudioClip audio;

private void Update()

{

GetComponent<AudioSource>().clip = audio;

GetComponent<AudioSource>().playOnAwake = false;

}

private void OnCollisionEnter2D(Collision2D collision)

{

if(Key.tag == "Key1")

{

inv = FindObjectOfType<Inventory>();

Door1 door = FindObjectOfType<Door1>();

GetComponent<AudioSource>().Play();

Item item = ScriptableObject.CreateInstance<Item>();

item.sprite = Key.GetComponent<SpriteRenderer>().sprite;

//SceneController.addItemToInventory(inv);

inv.AddItem(item);

door.addKey(item);

//Destroy(door1);

Destroy(Key);

}

}

}

On the door that will use the key to open add this script to it, and make sure it has a tag called Door2

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Door1 : MonoBehaviour {

private Item keyItem;

private GameObject door;

void Start()

{

door = GameObject.FindWithTag("Door2");

}

public void addKey(Item itemToAdd)

{

keyItem = itemToAdd;

}

private void OnCollisionEnter2D(Collision2D collision)

{

if(collision.transform.name == "Peasant")

{

Inventory data = FindObjectOfType<Inventory>();

if(data.hasKey(keyItem) && keyItem != null)

{

data.RemoveItem(keyItem);

Destroy(door);

}

}

}

}

In our case the door that uses the key is the one shown below.



Now your door should open when you have the key in your inventory.

## Event Trigger Script(s)

I say scripts because it is the same script but used two times with multiple game objects

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PuzzleOneScript : MonoBehaviour

{

public GameObject PuzzleStatue;

public GameObject Key;

public GameObject greenButton;

public GameObject redbutton;

public new AudioClip audio;

// Use this for initialization

void Start()

{

GetComponent<AudioSource>().clip = audio; //grab the audio component from the start

GetComponent<AudioSource>().playOnAwake = false; //disables auto play

Key.SetActive(false); //sets the key game object to false

greenButton.SetActive(false); //sets the active button to false

redbutton.SetActive(true); //sets the disable button to true

}

private void OnCollisionEnter2D(Collision2D collision)

{

if (PuzzleStatue.tag == "Puzzle1") //if you collide with the game object puzzle1

{

Key.SetActive(true); //creates a gameobject key

GetComponent<AudioSource>().Play(); ///plays the sound

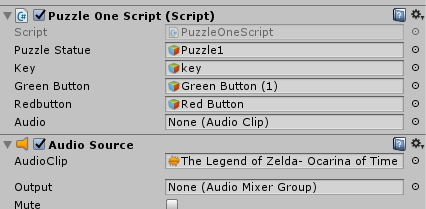
greenButton.SetActive(true); //enables active button

redbutton.SetActive(false); //disables disable button

}

}

}



The trick on this script is that the object are already created and ready to go, but theyre turned off by default as necessary and only turn on when an event occurs

## Dead Screen

The cheap trick for dead screen is that it just reloads the scene as a way to restart since there is no checkpoints on the game

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class ReloadScene : MonoBehaviour {

public GameObject WrongStatue1;

public GameObject DeathScene;

// Use this for initialization

void Start () {

DeathScene.SetActive(false); //dead splash screen off by default

}

private void OnCollisionEnter2D(Collision2D collision)

{

if(WrongStatue1.tag == "Death") //uf you touch the wrong thing

{

DeathScene.SetActive(true); //sets dead screen on

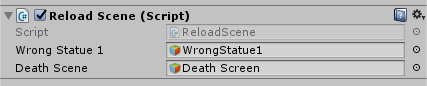
System.Threading.Thread.Sleep(2000); //waits 2 seconds

SceneManager.LoadScene(SceneManager.GetActiveScene().name); //reloads the scene and starts the game again

}

}

}



## Main Menu Script

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class MenuScript : MonoBehaviour {

public GameObject menuScreen;

public GameObject menuScreenResume;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

// this is used to make the menu follow the player camera and always be on the center.

//it is not needed because we already made the canvas follow the player on the inventory script

// Vector3 pos = Camera.main.WorldToViewportPoint(transform.position);

// pos.x = Mathf.Clamp01(pos.x);

// pos.y = Mathf.Clamp01(pos.y);

// transform.position = Camera.main.ViewportToWorldPoint(pos);

}

public void resumegame()

{

menuScreen.SetActive(false); //sets the first button active

menuScreenResume.SetActive(false); //sets the second menu button active

}

public void goToExit()

{

SceneManager.LoadScene("MainScene"); //if quit button is selected, load main scene

}

}

## Show scroll, and open menu script

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class ShowScript : MonoBehaviour {

public GameObject scroll;

public GameObject trigger;

public GameObject menuScreenExit, menuScreenResume;

// Use this for initialization

void Start () {

//sets game objects to disable

scroll.SetActive(false);

menuScreenExit.SetActive(false);

menuScreenResume.SetActive(false);

}

// Update is called once per frame

void Update() {

if (Input.GetKeyDown(KeyCode.Escape)) //if esc is pressed open memnu

{

menuScreenExit.SetActive(true);

menuScreenResume.SetActive(true);

}

}

//colider scroll setup

private void OnTriggerEnter2D(Collider2D collision)

{

if(trigger.tag == "Scroll")

{

scroll.SetActive(true);

}

}

//when player leaves the colider, disable the scroll

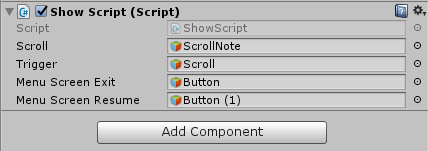
private void OnTriggerExit2D(Collider2D collision)

{

scroll.SetActive(false);

}

}



## Hey where is the rest of the scripts?

Like I said, most scripts are exactly the same but with different game objects selected, some use door1 to open the first door and some use the same script but referencing to door2

You can find those scripts [here](https://drive.google.com/drive/folders/0B-6tbB7_VGCkQTZHUUxQb1ZlLXc?usp=sharing)

# Audio

Most audio in this game is copyrighted. I do not recommend doing this if you want to sell this game for a profit

# Main screen

The main screen is really just a bunch of google images put together and edited using GIMP 2.0. You can find those images on the sprites folder, and arrange them as you please.

Then create two buttons and use this script

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class TitleScreenUI : MonoBehaviour {

     // Update is called once per frame

     void Update () {

         //used to quit the game

          if (Input.GetKey("escape")) //if player presses esc

               Application.Quit(); //application quits

     }

     public void StartGame()

     {

          SceneManager.LoadScene("Level1"); //loads scene for level 1(the only level)

     }

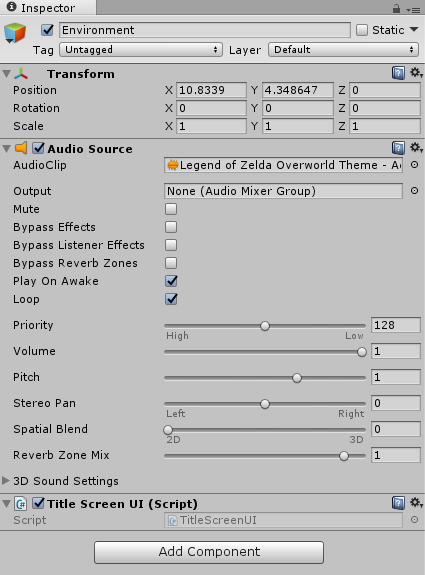
     public void QuitGame()

     {

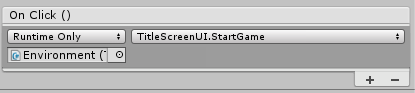
          Application.Quit(); //player quits if press the quit button

     }

}

Then drop the script into a game object called environment on your main title scene. 

On the two buttons you create, select the on click action and drag the environment game object, then from the list select the quit game or start game function



# Conclusion

Once you set up the player movement script and the puzzle scripts to the rest of the map, the game will be pretty much done.

This is by no mean a game breaking game. It is a pretty simple but working game