- Start with the Unity HUB and create a new 2D project. Name it HelloKitty.
- Use resolution 1024x768 from the settings in the game window*
- Create new project folders so you can organize all game elements like Sprites, Scripts and so on.

* If this resolution is not there then click the plus(+) and add this yourself by name a setting and adding its resolution.

Creating the project folders is easy. Go to the Project tab. One folder is already there called Scenes. Use the Right mouse button and click it in the grey part. Select Create and then Folder from the menu. Name the folder and repeat these steps until you have all the folders you need. For this project you will need these folders:

- ***Fonts***
- ***Scenes***
- ***Scripts***
- ***Sounds and Music***
- ***Sprites***

Right under the tab Assets you will find a little slider. When you hold your left mouse, button clicked on it you can drag it left or right so the folders will show bigger or smaller. You can stretch out this whole tab if you need by holding the left mouse button on one of the edges of the tab and drag it to make it bigger or smaller. Creating project folders is a basic thing that returns doing with every new game project you will create.

To import all the elements, you need to simply move into the folder to use. Right mouse click in the grey area and choose Import New Asset. Go to the folder where the files are located, select them and click the Import button. All the files will appear in your project folder.

To change the background color of the camera all you need to do is select the Main Camera in the Hierarchy list. In the Inspector window you will see the Camera options Click on the Background color and select the color you want it to be.

We can add game objects to our Scene view by holding the left mouse button on it and dragging it to a place in the Scene View. When this object is selected in the Hierarchy you can see in the Inspectors Window the options of this game Object. We set the Draw Mode to Tiled. If you select the Rect Tool we can stretch out the game object as big as we want. Because of the draw mode it will repeat itself continuously. You can make a full line using multiple numbers of the same image this way. When you don’t need this don’t use the Draw Mode Tiled but set it too Simple.

You can change the name of any game Objects. There are two ways of doing this. Right Click the Game Object name in the Hierarchy and select rename to change the name of it. You can change the name also by typing it directly in the Inspector Window with the game Object selected.

Place the second game Object in the way you learned. When this object is selected you can give it an Order In Layer number in the Inspector Window. The higher this number is the more in front of other game Objects with a lower number it will be placed. This way game Objects can hide behind others or make it look like the slide behind each other.

When the Scale tool is selected you can scale game Objects bigger or smaller by holding the left mouse button and drag it into the direction scaling is needed. If you want to scale more than one game Object at the same time just select them all by using hold with left mouse button and draw a rectangle on all objects you want to scale. You can also select the game Objects by holding the control key and click the object names in the Hierarchy.

Place the heart game Object and scale it as you learned how to do that.

Save your project by selecting File on the top bar. Select Save as and go to the Scenes folder. Store it under the name you want to use. Once saved go to the build settings and make sure your new level name is selected by selecting Adding Open Scenes. The one not used can be taken out by unchecking the selection box.

Although we are creating a 2D game it's made in a 3D Environment. You can see this when you click the 2D button. It will go to 3D mode and you will see that the camera is placed away from all used game objects. It has to be, otherwise it could not see anything or it was looking straight through the game objects because it was to close. Clicking the 2D button again makes it toggle back to 2D view, the view we need to build this game in.

We can give each Object their own unique tag name. This will be useful as we can call those names in scripts. This way Game Objects can be found and recognized when the game is running. The second game Object will be the player so we give it the Player tag that is standard there in Unity.

Select the game Object then go to the Inspectors view to select Tag and choose Player. For the heart we will create the tag name Heart. Select the heart game Object and select in the Inspectors window Tag and select Add Tag. Click the + (Plus) and type the name you need and save it as a tag. Now with the game Object selected you can Add a tag as before you will see that Heart is now in the list to select.

It is time to create our first script. Move into the Scripts project folder as we need our scripts to be stored in there. Right click in the grey area and select Create – C#-Script. A new Script Icon will show that you can give any name you want. This will be our first script and we will create it for the player game Object. Double click the script and it will open up in any Script editor that you set to use.

Each new script has a basic set up. The first 3 lines that start with Using are references to libraries of Unity. These libraries contain scripts and elements that unity uses so we don't have to create them ourselves.

The class line contains the name of the Script. Make sure it is exactly the same otherwise the script will not be found and gives you errors.

Void means function so there is a function called Start. A function is always noted: void (name of function)() between the brackets you can insert your own script coding instructions. It's an empty function when the new script is created. The same happens next where an Update function is placed. (This one is also empty when the script is freshly new created).

The same way

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

public class Kittymove : MonoBehaviour
{

// Start is called before the first frame update
void Start()
{

}

// Update is called once per frame
void Update()
{
}
}
```

A game will play on the monitor of the user by using frames like a movie or animation. In the start function you can place instructions that should happen before this movie runs. In the update function the instructions in here will continuously play every frame until we stop it with another script. You heard of framerates in a game? Well, this is what is meant with it. Games or movies they both work with frames and framerates.

We will edit the script so it will make the player move to the right automatically at a given speed.

First, we create a public float variable and we name it MovementSpeed. A float variable is like a box where you can put in or get out numbers that can use decimals. This way you can store these numbers and call them whenever you need. Public is used so the name will show up in the Inspectors View and we can change it there. When public is not used, we don't have this option and need to set the speed number in the script itself. Each script line ends with and Semicolon.

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

public class Kittymove : MonoBehaviour
{
public float MovementSpeed;

void Update()
{
transform.Translate(Vector2.right * MovementSpeed * Time.deltaTime);
}
}
```

In the update function we use one line of code to make the movement happen.

Transform is the place of the game Object and Translate means we move it to another place. Between () we say that it should use a Vector2 (Position in a 2D game) to the right. It should move with the given MovementSpeed multiplied by Time.deltaTime. Time.deltaTime makes sure that the speed is the same on any computer system the game is played one. As the start function is not used, we can remove it. Save the script. Now this script is in our project folder. When hold on left mouse button we can drag it to our Scene window and drop it onto the player Game Object. You have now given a game object its own script.

When the player game Object is selected you can see in the Inspector view that it has now the script in its options and we can insert any number for the moving speed because we made this variable public.

In order to test the game, we use the game window and press it's play button. You will see that kitten is moving to the right. It will keep moving even when it goes off screen cause in the update function, we set its moving instruction. It will keep doing this until we script a reason to stop it. Press the play button again to stop testing.

Some extra information.

In a 2D game there are 2 axis used to move. The X axis (Left-Right) and the Y axis (Up- Down). There is also a Z-axis that is used for 3D games. Remember that this 2D Game is made in a 3D environment. Moving on the Z-axis would mean (movement) Forward to the camera or back from the camera. When you select a game Object and move it around you can see its position on all axis in the Inspectors view.

This concludes Chapter 1. If you think this is all clear and you understand it all, it's time to move on to the next chapter for more scripting and unity working adventure.