Introduction to Unity's 2D Environment

The Importance of Concept and Planning Ahead

The most important aspect of a video game is to be *fun* to play. Although graphics and a deep story go a long way, many games that are not particularly strong in those 2 aspects are still successful because they're entertaining.

1)Before you start on your project, sit down and think about what you want your game to be:

- A 'minutes to kill' or puzzle-solver? (e.g. Angry Birds, Tetris, Sudoku)
  - Entertainment > Story+Art
- A classic platformer? (e.g. Super Mario Bros, DuckTales Remastered, Rayman)
  - Entertainment+Art > Story
- An adventure/visual novel? (e.g. Detroit: BH)
  - Story+Art>Entertainment
- A combination of 2 or more of the above? (e.g. Legend of Zelda, Professor Layton)
  - Story+Art+Entertainment

The Importance of Story/Premise and Planning (Cont.)

2)After you've decided on a genre, use your favorite games as inspiration

- What are the things your favorite games did GREAT? Re-use the successful concepts and add your own innovations!
- What are the things you didn't like about those games? Bad controls? Ugly backgrounds? Uninteresting story? Recognize their mistakes
- Every work of art is inspired by a work of art that came before it. Do not be afraid to take ideas and lessons from your favorite games. Recycle ideas, but don't plagiarize.

3)READ about game design. Learn the rules that make a game worth playing. Only when you know the rules, you can experiment by breaking them.

4)Plan your game before you start implementing it. You can make changes while you work, but make sure you at least have a basic map to follow.

What is Unity?

Unity is a cross-platform game engine with a built-in IDE developed by unity technologies. It is used to develop video games for web plugins, desktop platforms, consoles, and mobile devices.

– *Wikipedia*

What is IDE, and what is a Game Engine?

- **IDE** stands for “integrated development environment”.
  - An IDE example is Netbeans, where IDE is a software application that provide comprehensive features to computer programs for software development. An IDE brings many of those development-related tools together as a single framework, rather than have the developer select and manage all of these tools separately.

- A **game engine** allows a game created to run in different environments
  - A game engine is framework for game development. This framework helps with several core areas that all games have:
  - Graphics
  - Audio
  - Logic

What is framework?

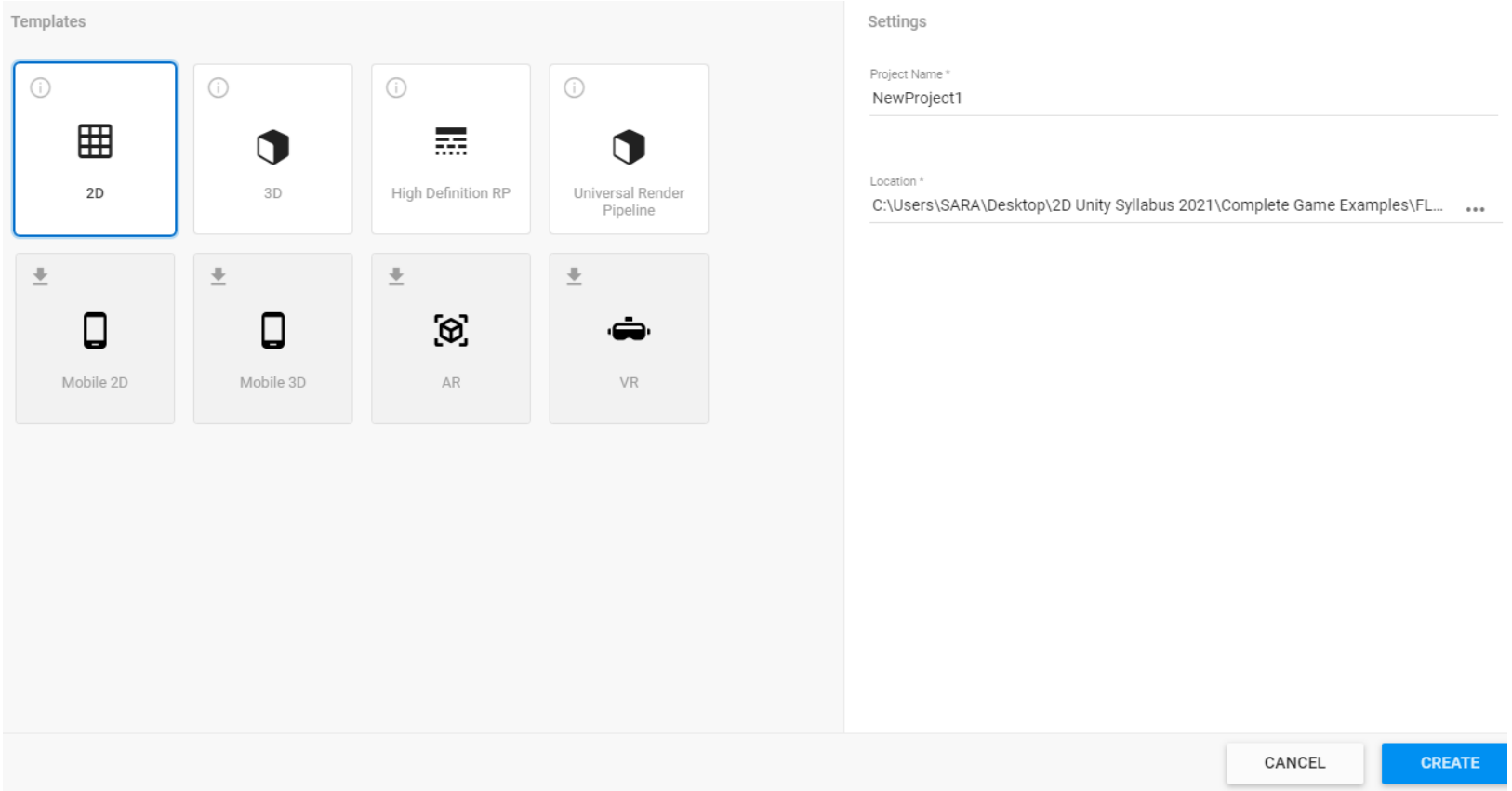
- It is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code providing special functionality

- That’s why the game engine is a framework – it provides tools and structures that every game requires to function

- Using an engine means you don’t have to reinvent the wheel every time you start a new project

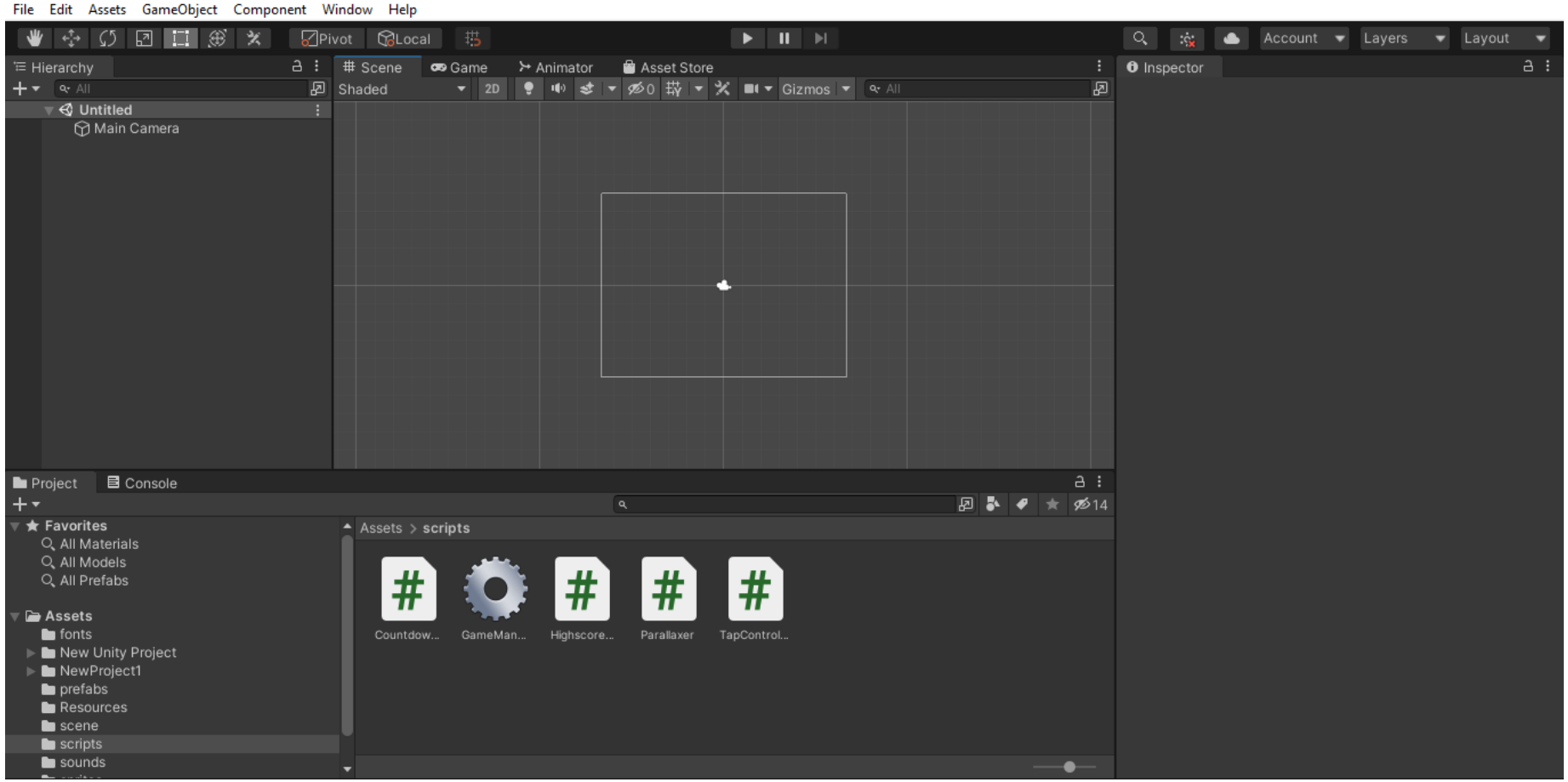
First Look at Unity

- Open Unity to create a new project. Name the project and set the environment to **2D**.



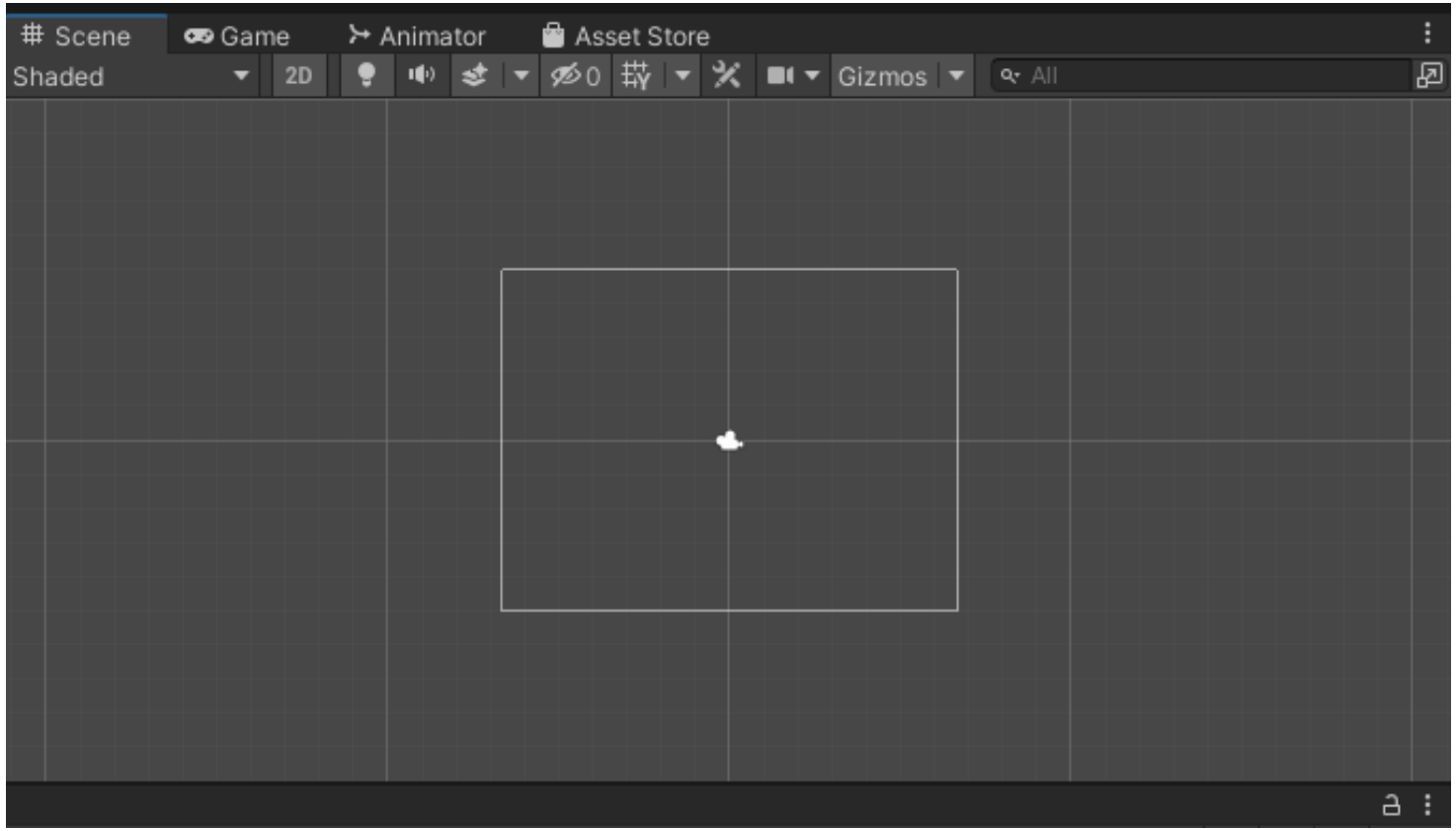
First Look (Cont.)

- This is your default workspace.



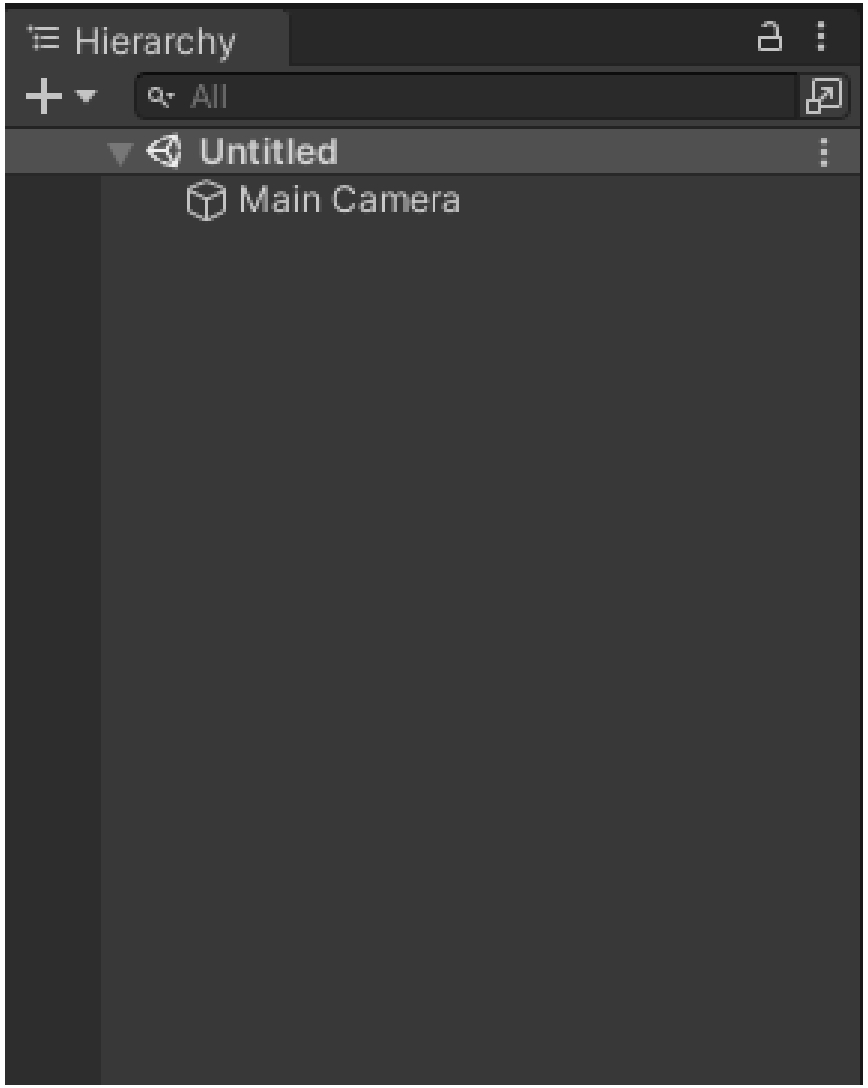
First Look (Cont.)

- Scene/Game panes
  - This is where you assemble all the elements of your scenes in the game



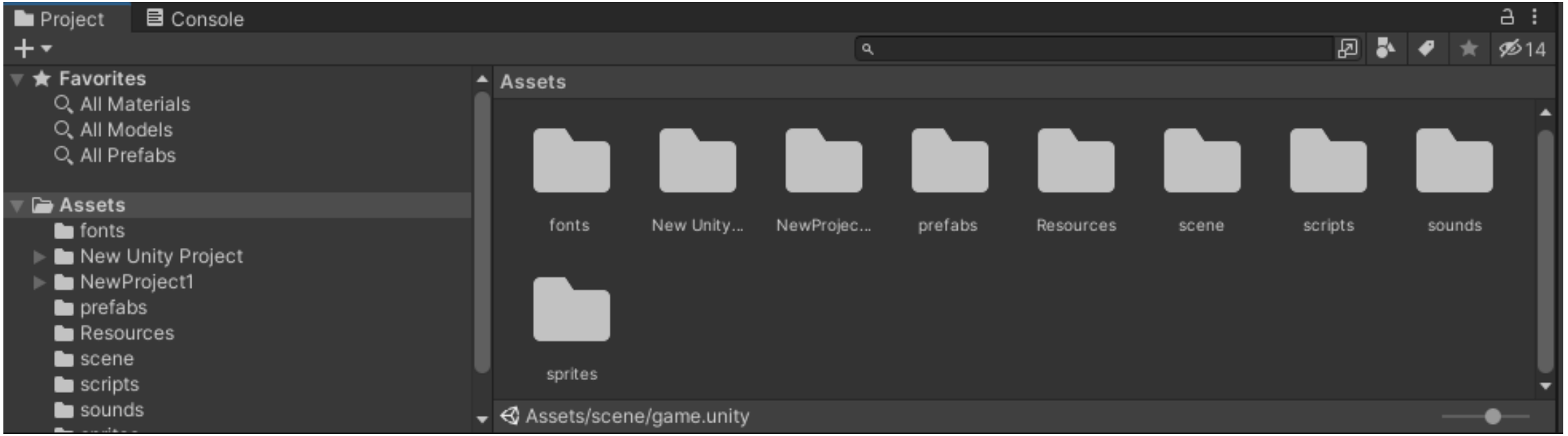
First Look (Cont.)

- Hierarchy Pane
  - Lists all the elements that exist in your scene, and establishes parent-child relationships between them



First Look (Cont.)

- Project Pane
  - This is where you organize all the GameObjects that you'll be using in your scene, from character sprite sheets to animation to background imagery to code. Note: keep your assets organized by putting them in folders! (e.g. Codes, CharacterSprites, etc.)



First Look (Cont.)

- Inspector pane
  - This is where you can see and edit the properties of any element you've selected to work on

First Look (Cont.)

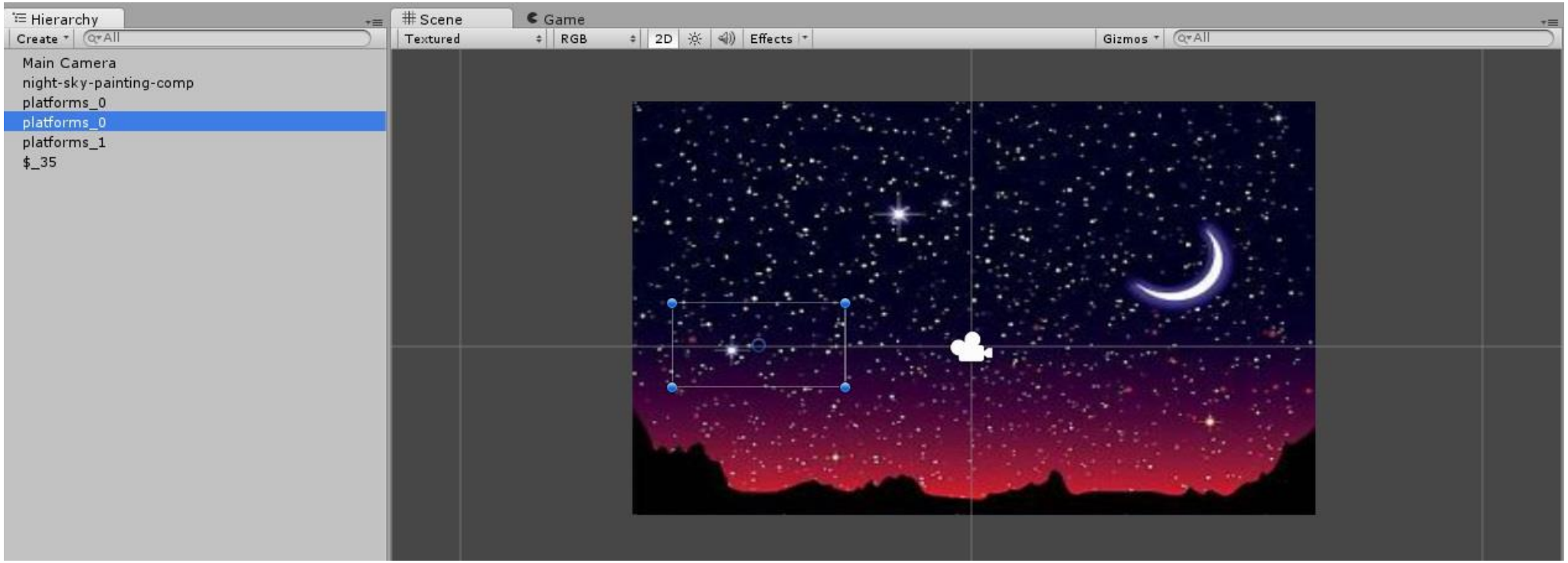
- Transform tools
  - Easy-access tools to change the position, size and orientation of any element in your scene
  - The hand tool moves the entire scene pane canvas
  - The move tool moves a selected element around the scene pane
  - The rotate tool rotates a selected element around the x or y axis
  - The transform/scale tool allows for increasing the size of an element along the x axis, y axis or both
  - **Note: if these tools are not accurate enough, use the x and y axes values in the inspector pane instead!**

Exercise #01 – Compose a Simple BG

- Go online and choose several elements to compose a pleasant foreground and background. E.g. trees, grass, mountains, birds, blue sky, etc. Be creative!
- Create a folder in your project pane. Name it Background_Sprites
- Drag and drop sprites into the folder
- Drag and drop the sprites from the folder to the project pane. They'll automatically also appear in the scene pane. Assemble everything to create an interesting background
- Use the transform tools in the toolbar to help you rotate, resize, etc.

Exercise #01 – Create a Simple BG (Cont.)

- You may find that the images are not being assembled the way you want (e.g. the suspended platform is hiding behind the sky)

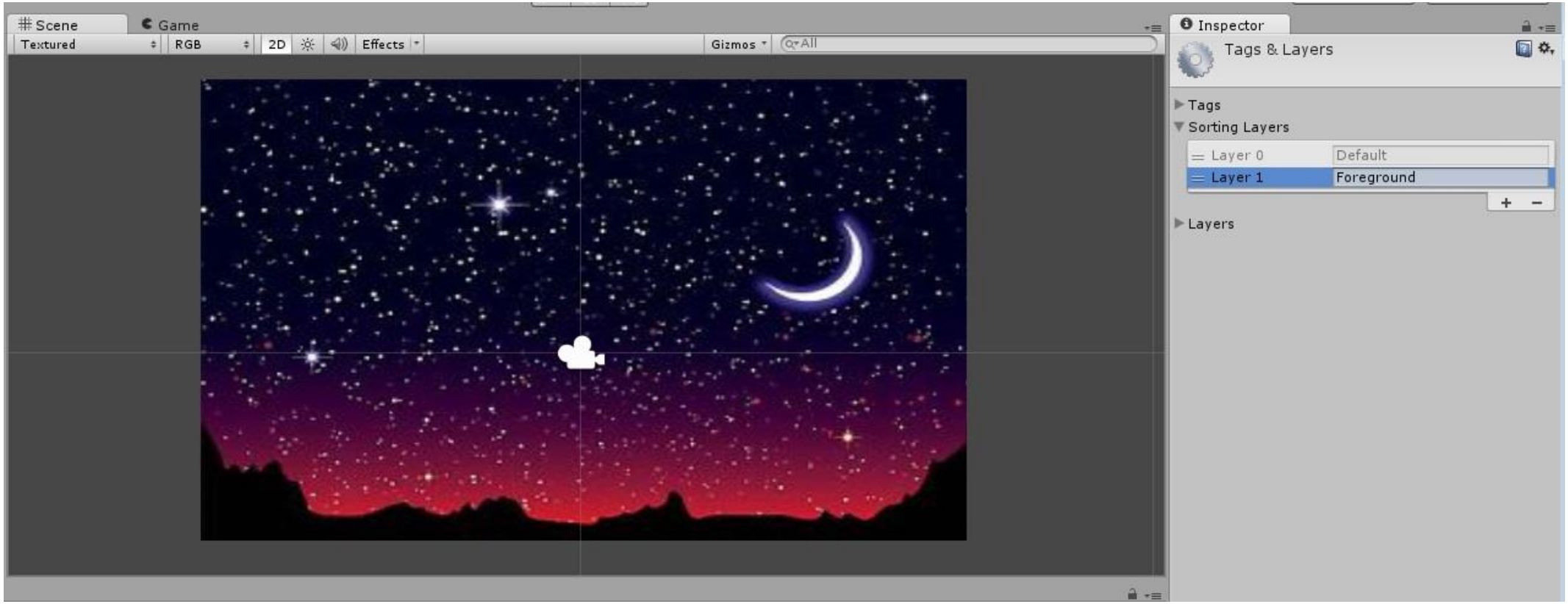


Exercise #01 – Create a Simple BG (Cont.)

- To re-assemble them correctly, select the platform and go to the inspector pane
- Choose Sorting Layer > Add Sorting Layer

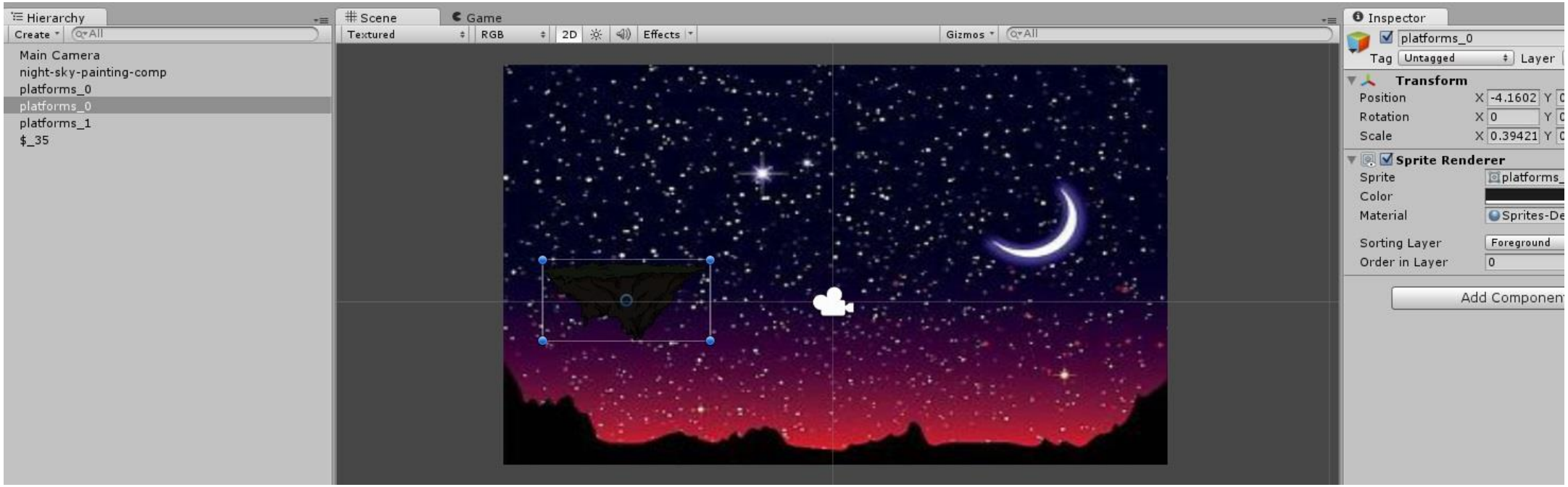
Exercise #01 – Create a Simple BG (Cont.)

- Click the + sign, and add a new layer named Foreground. The horizontal lines on the left of each layer allow you to re-assemble if needed



Exercise #01 – Create a Simple BG (Cont.)

- Re-click on the platform's sprite name in the hierarchy pane, and change its sorting layer from Default to Foreground
- Now create another sorting layer and name it Background. Place all your background elements like the sky in it



Exercise #01 – Create a Simple BG (Cont.)

- To see what your scene looks like so far, click the Game pane

Exercise #01 – Create a Simple BG (Cont.)

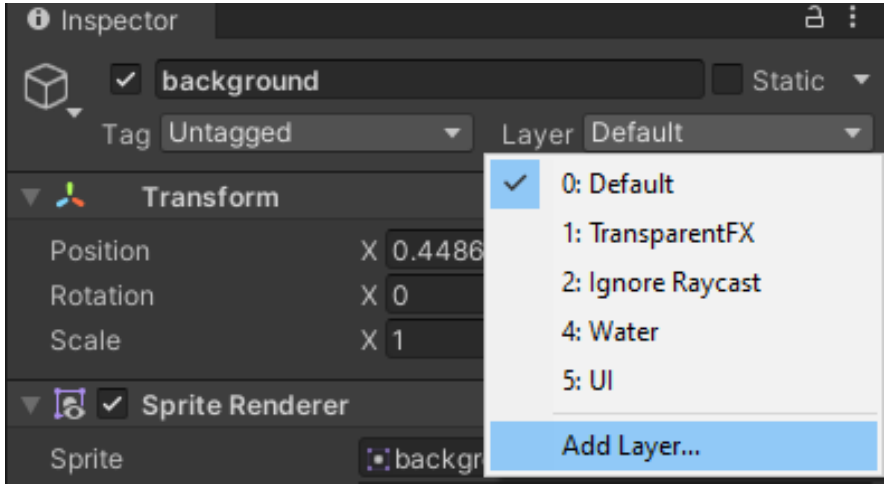
- **Quick note:**
- *Sorting Layer* helps organize how sprites are seen by the camera. Since it is a property of the Sprite Renderer component, it determines how sprites are rendered in the scene.

- As for *Layer*, which is above the Inspector pane next to Tags, is for grouping objects and can be later manipulated in code. For example, you may want some layers to allow collision (e.g. Ground), but not others.

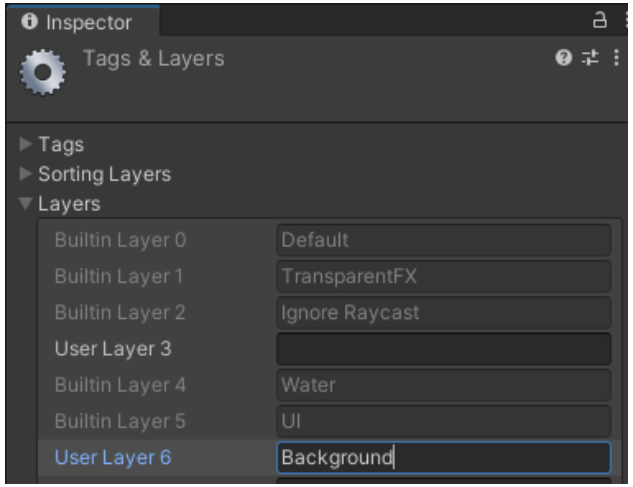
- So Sorting layers serve the visual side, whilst Layers more serve the functional side.

Exercise #01 – Create a Simple BG (Cont.)

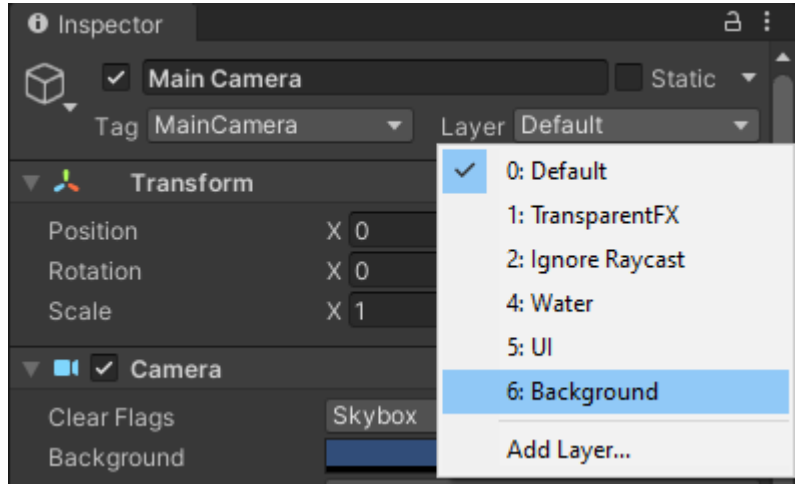
- To lock a layer and avoid accidentally moving it around, go to Layer in the Inspector pane. Choose Add Layer, and create a new one (e.g. Background)
- After it's created, select it from the list instead of Default



- Step 1



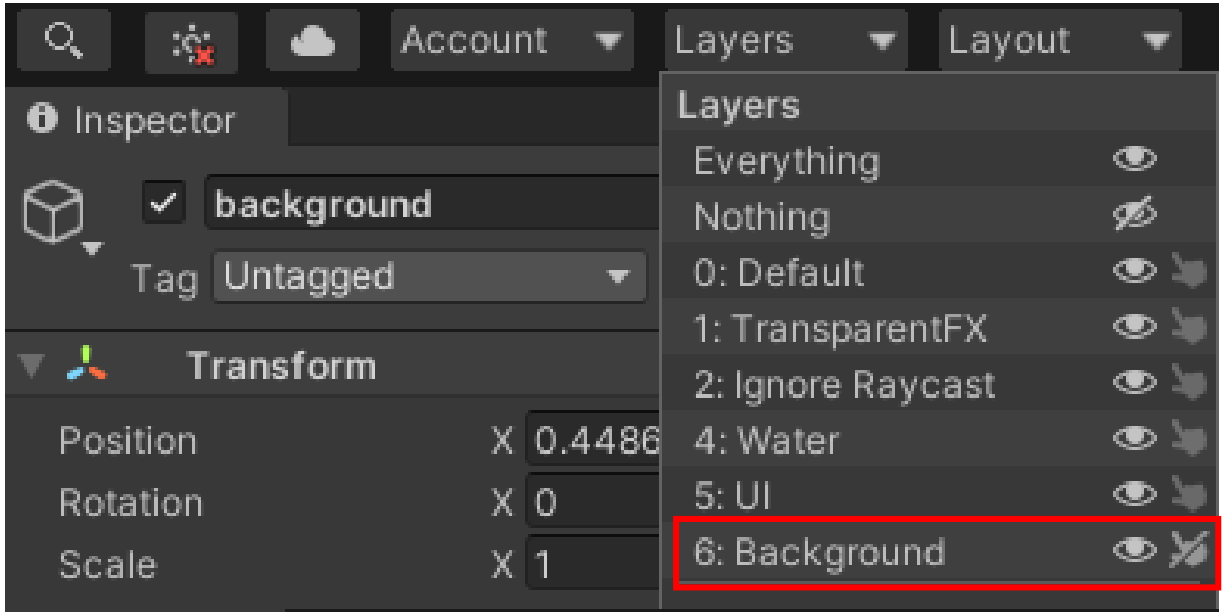
Step 2



Step 3

Exercise #01 – Create a Simple BG (Cont.)

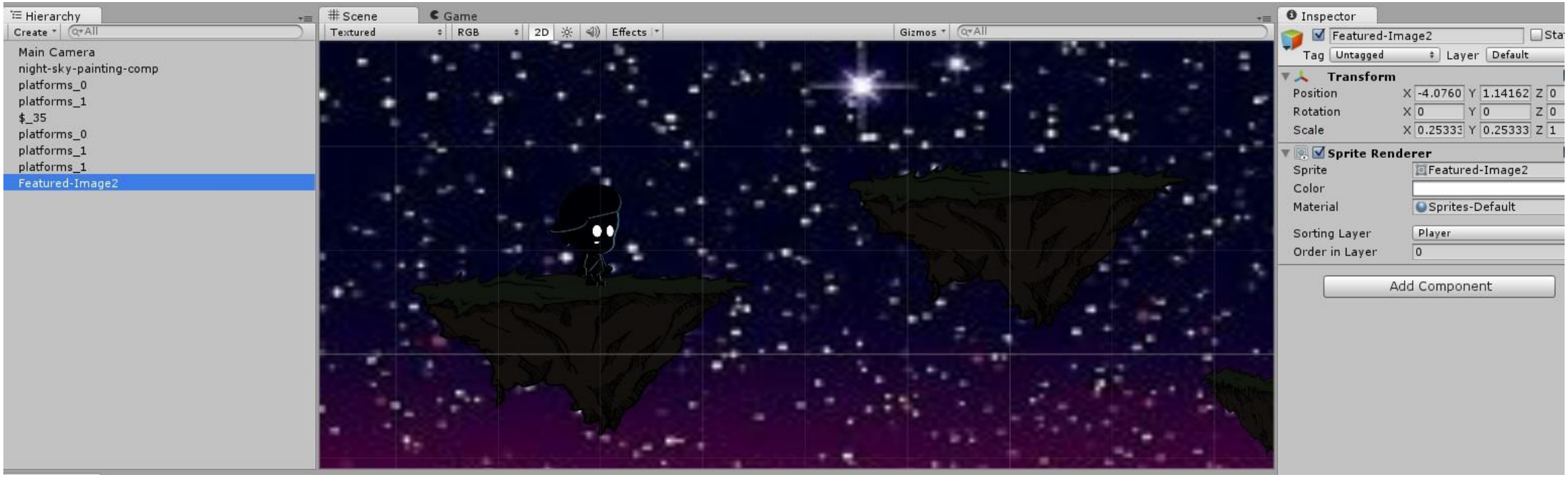
- Go to the Layers drop-down list above the Inspector pane and choose Background (or whatever layer you had created)



- To lock the layer, click the “Pickable” icon
- Now click outside the sprite you wanted to lock. When you try selecting it again, it will not be highlighted or moved

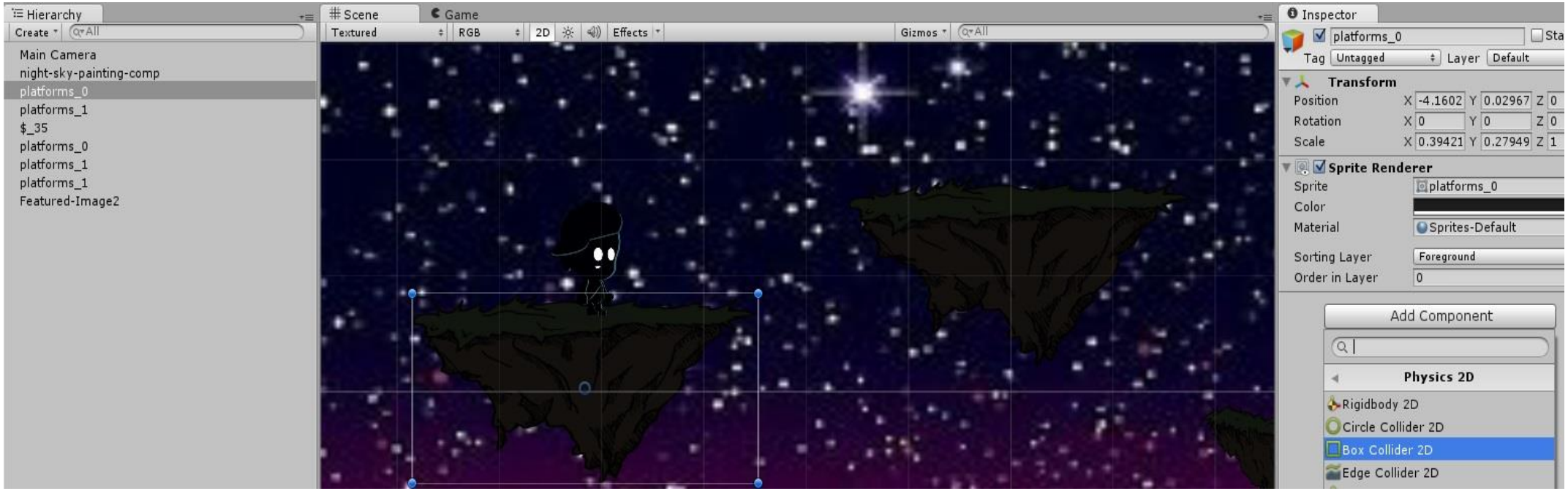
Exercise #02 – Add a Character

- Choose a character sprite from the internet and position them in your scene
- You may again have a problem with layers. Create a new sorting layer, name it Character/Player and position it above both foreground and background



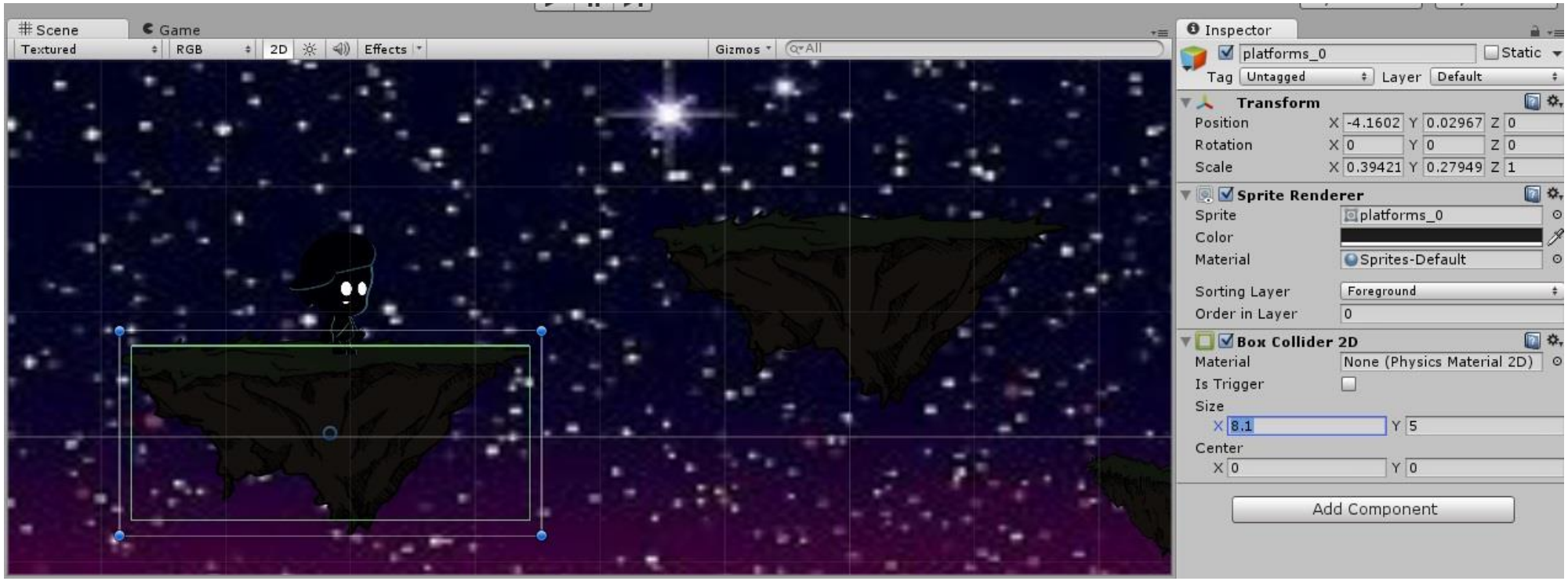
Exercise #03 – Make the Ground Solid

- You'll need to make the ground solid for your character to stand on
- Select ground>go to Inspector pane>Add Component>Physics2D>BoxCollider2D



Exercise #03 – Make the Ground Solid (Cont.)

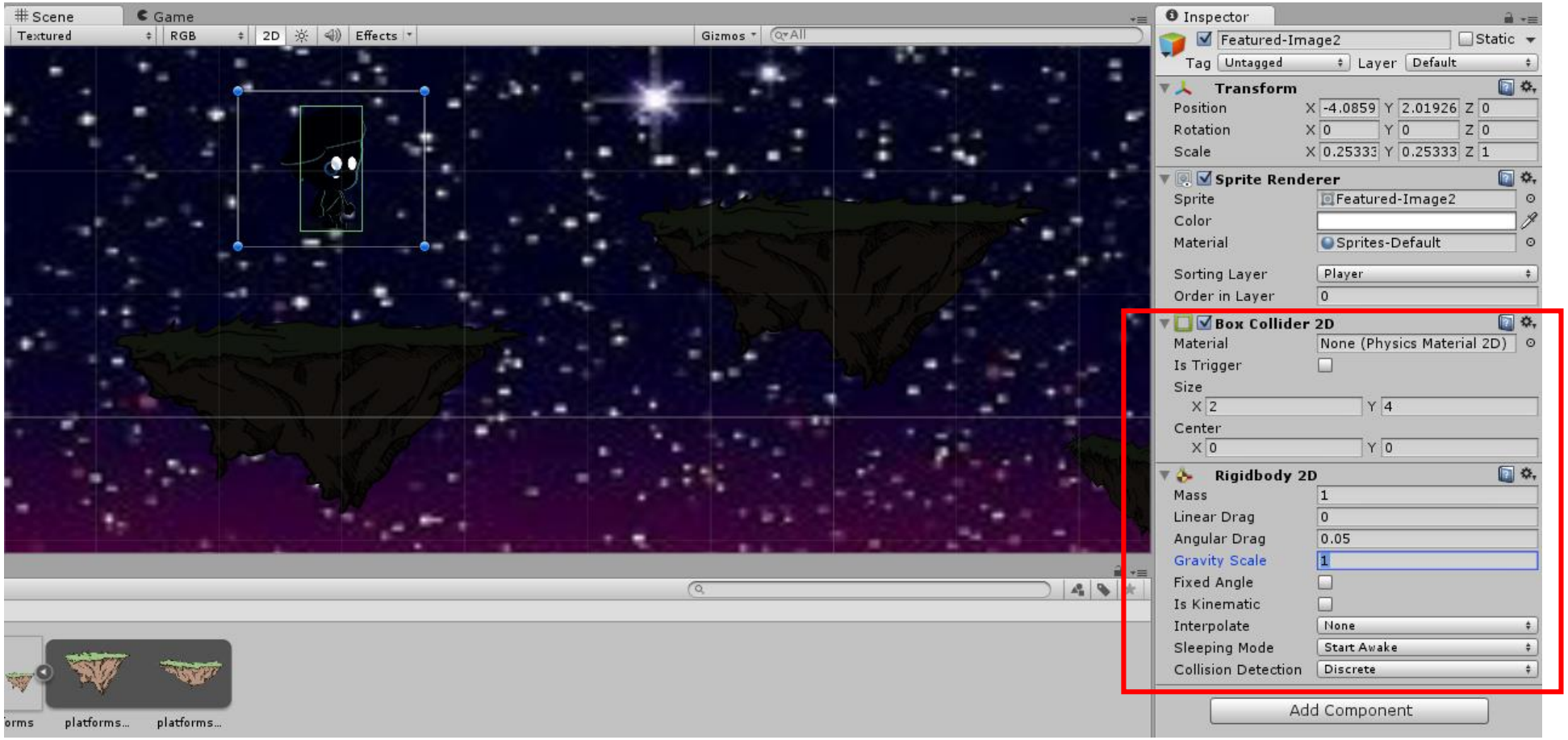
- To adjust the size of the (green) collider around the ground exactly the way you want, play with the values of the x and y axes in the Inspector pane
- Now the ground is solid



Exercise #04 – Make the Character Solid

- Using exercise #03 as an example, add a suitable collider to your player character
- Click the Play button, but still nothing happens!
- To make the player character respond to physics, a RigidBody must be added
- Select the player sprite> click Add Component button>Physics2D>RigidBody2D
- As you'll see, the Rigidbody component has many useful settings. Experiment with Gravity. Suspend the character up in the sky, then click the Play button to see them fall and hit the ground
- Click the play button again to stop Game Mode and change the gravity's value. Click play again. What happens?

Exercise #04 – Make the Character Solid



Camera

- In 2D games, the camera type is **Orthographic**, because we are working in 2 dimensions only
- If there is a problem with how your scene appears in front of the camera, click on the camera in the project pane to open its settings in the inspector pane
- Change the Size setting to whatever’s suitable

Camera (Cont.)

- Size set to 5. Too big.

Camera (Cont.)

- Size set to 3. Just right.

Quick Note

- **Never make changes to your scene while the Play button is clicked (Game Mode on), as they won’t be saved!**

Your Project

- Your project in this module is to create an intuitive 2D platformer game. It does not have to be complex or long. Creating a simplistic **retro game** like this with **smooth gameplay** and a **clear story** is completely acceptable.

My first beginner 2D platformer game on Unity 5.6

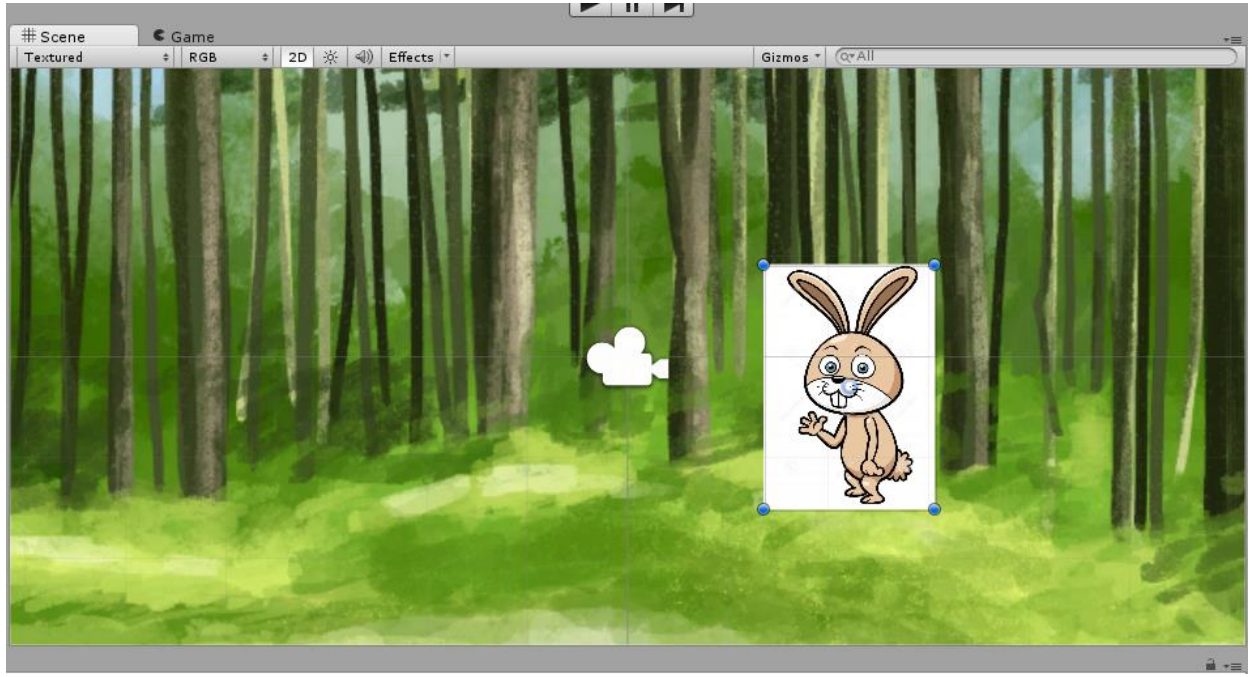
Source:
https://www.youtube.com/watch?v=Qx-GXD7L0vg

Reminder

- Before creating your level, determine the type/genre of game you want to make
- Plan all the important stuff on paper. Draw sketches.
- Determine the aesthetic and atmosphere, because they will dictate your art style and music
- Make use of the references at the end of this powerpoint

Types of Sprites

- In a 2D game, the art you use is called a Sprite.
- Your sprite must be **PNG** image filetype, to allow transparency
- If for example the sprite of your character does not have transparent a BG, you will have something like this:



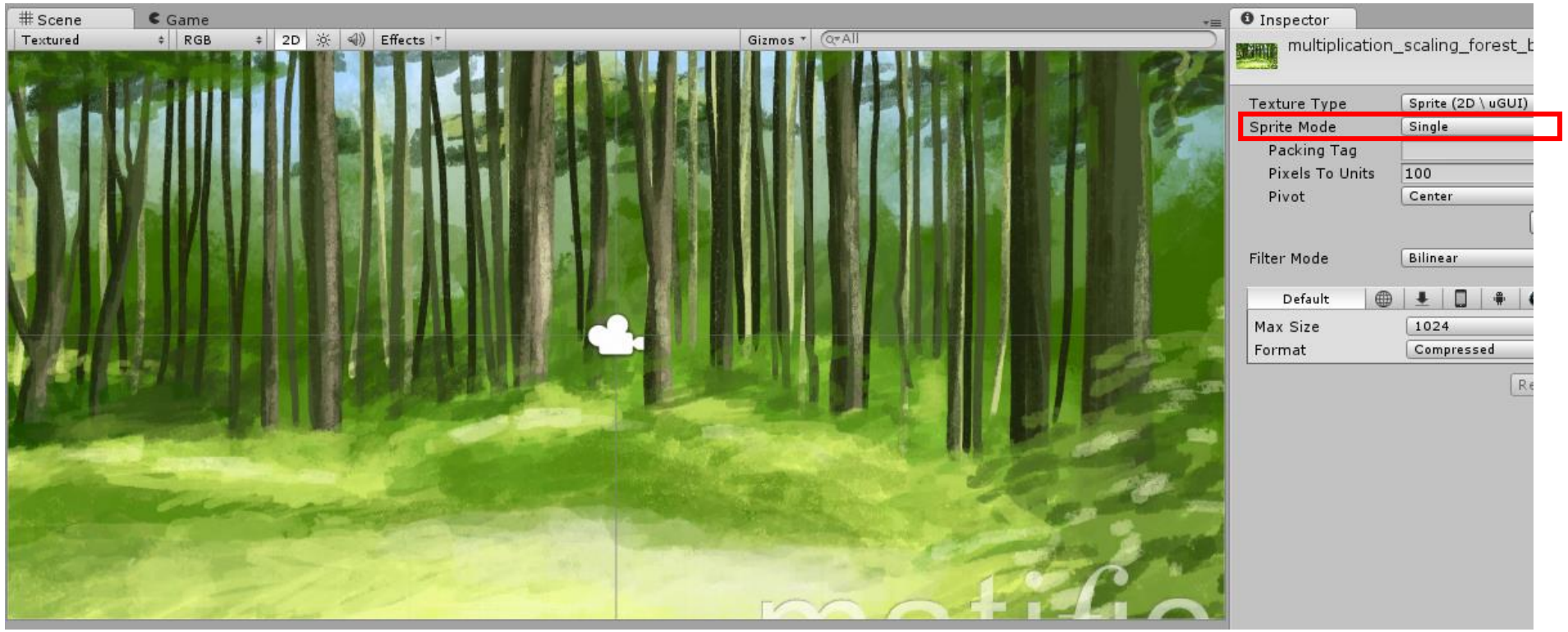
Types of Sprites (Cont.)

- But if your sprite has a transparent background, it will have gray and white checker marks as seen below:

- Now *this* can be used in designing your level without problem

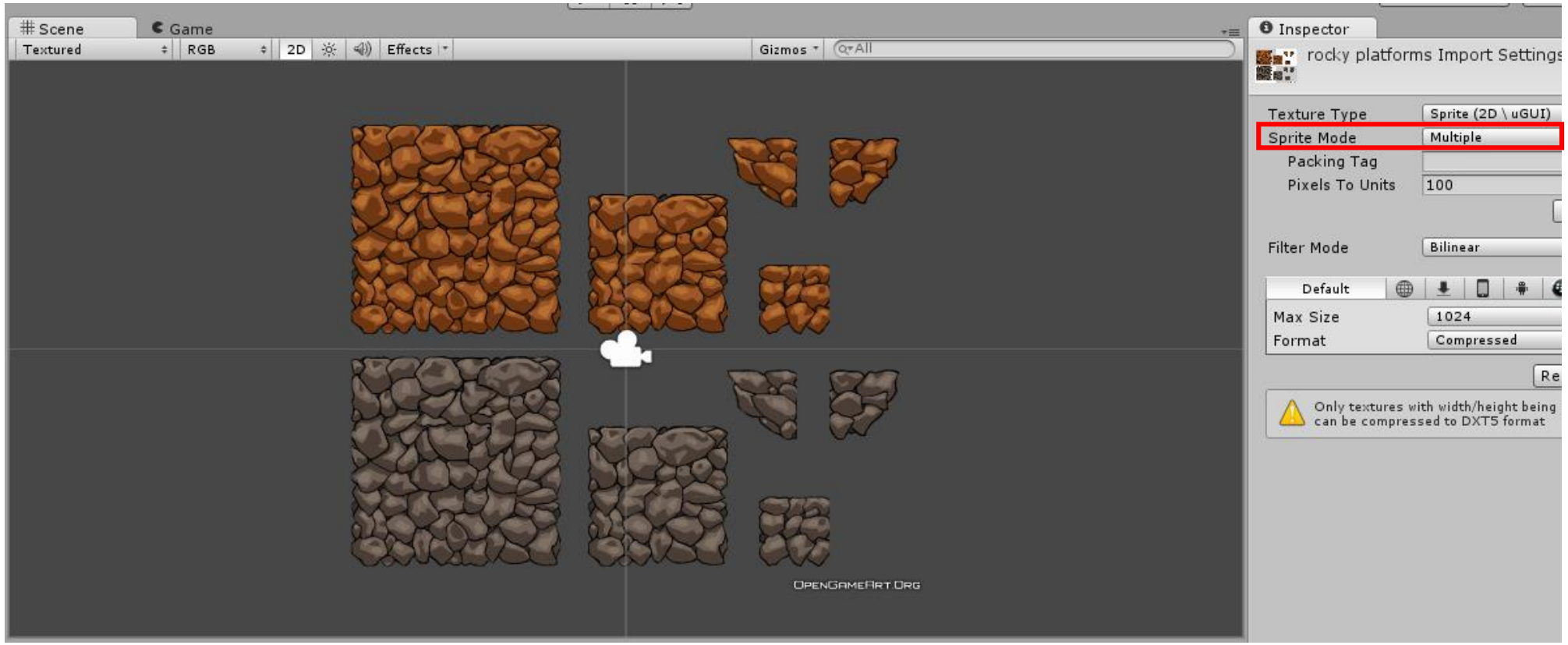
Types of Sprites (Cont.)

- A sprite in Unity can be 2 modes: single or multiple
- If a sprite is only 1 image, then make sure its mode is ***single***



Types of Sprites (Cont.)

- If a sprite is several images, make sure its mode is ***multiple***



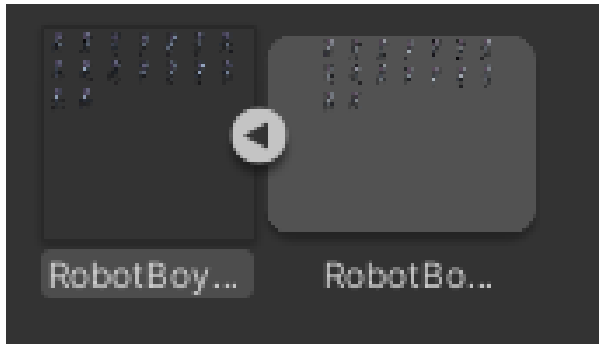
Types of Sprites (Cont.)

- Have you noticed that whenever you add a sprite to your assets pane, a copy of that sprite appears next to a small arrow?

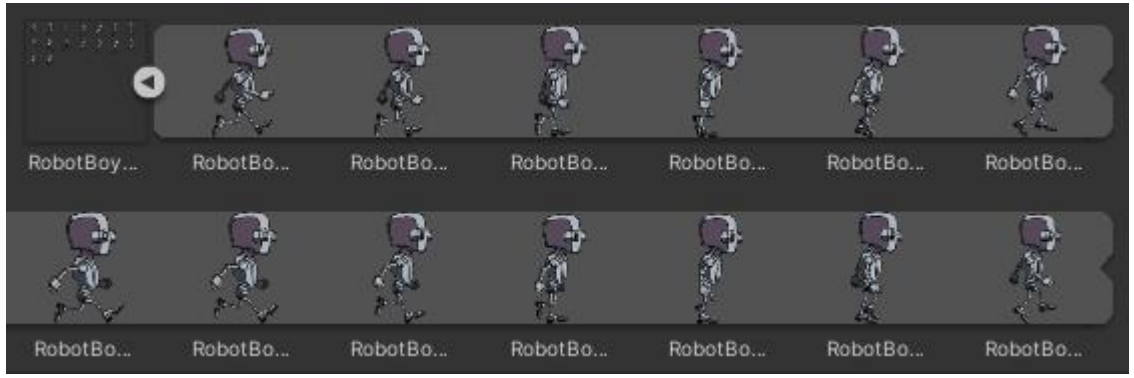
  - Unity does that automatically. It does not actually use the original copy of your sprite. It creates a *GameObject* out of the sprite and uses that instead.

Sprite Editor

- We want to cut up the spritesheet of RobotBoy into separate sprites so we can create the animations later
- How do we go from this

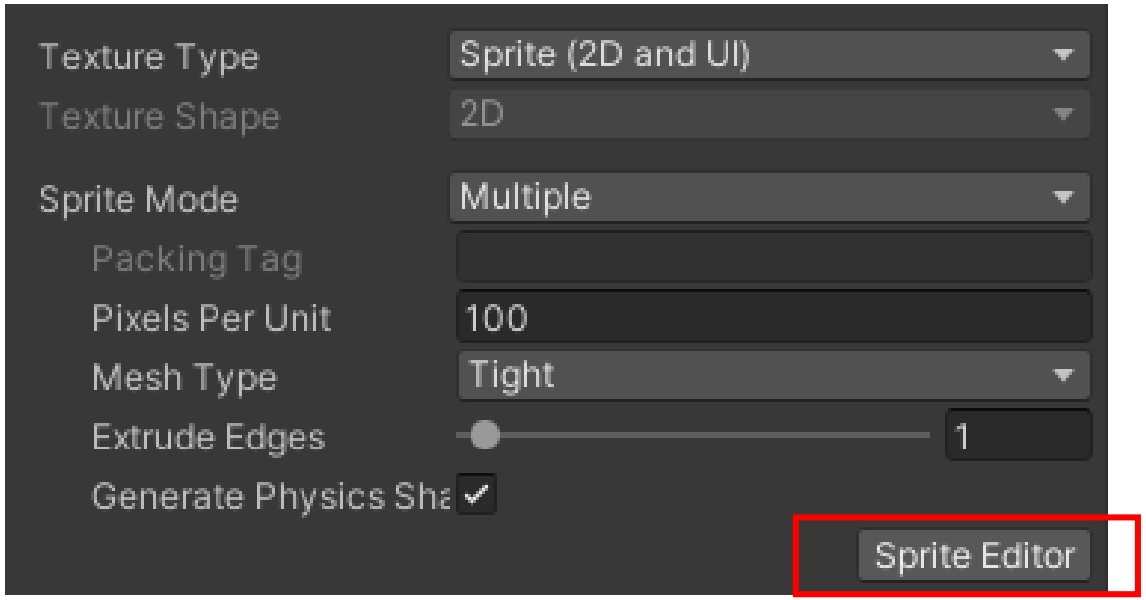


- To this?



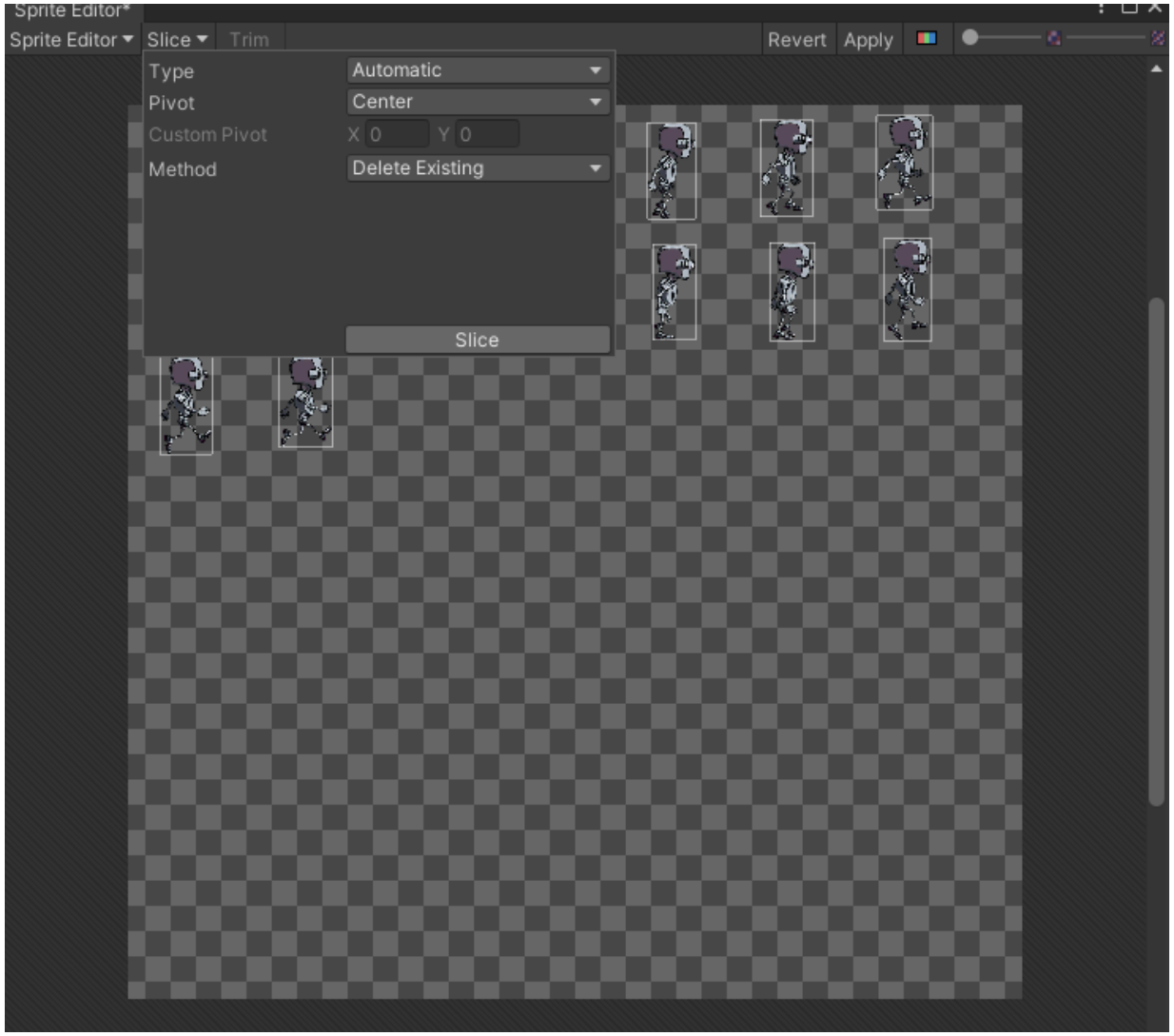
Exercise #05 – Slicing Sprites

- First, click the **Sprite Editor button** in the Inspector pane



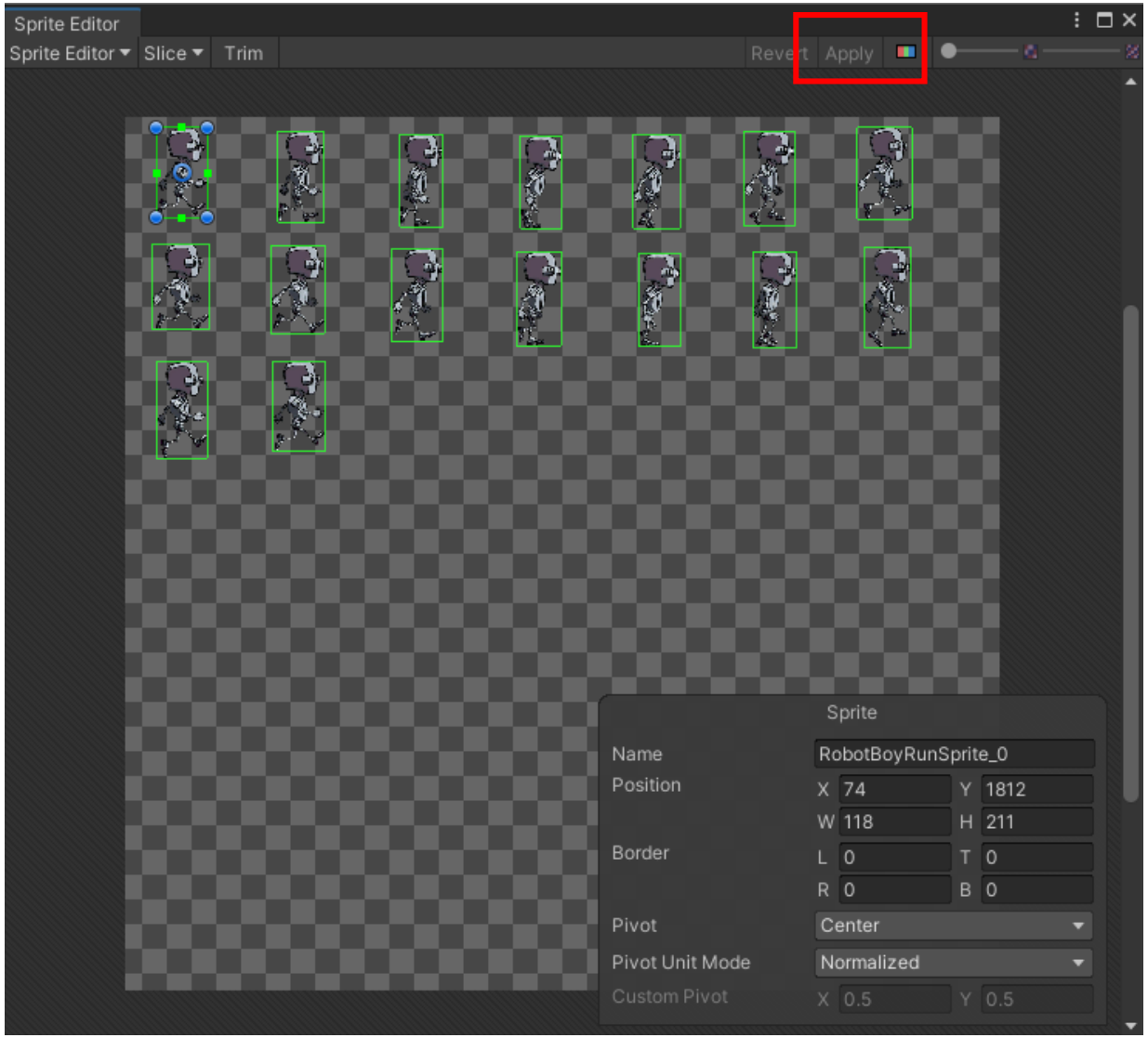
Exercise #05 – Slicing Sprites (Cont.)

- In the Editor, click the Slice button and Unity will automatically create outlines around each image in the spritesheet.
- No need to change any other settings for now.



Exercise #05 – Slicing Sprites (Cont.)

- When you're done, click **Apply** on the right



Prefabs

- A prefab is a master entity of a GameObject that you can use in your game endlessly. In other words, creating a prefab is like creating a template that contains all the characteristics of something (ex: an enemy or a bullet), and that template can be used to create many copies. That saves a lot of time.

Exercise #06 – Create a Prefab

- To create a prefab, first create a folder in the Project pane under Assets and name it PrefabsFolder
- Right-click the folder and choose Create > Prefab
- Rename that new prefab (e.g. Enemy)
- Select the GameObject you want to turn into a prefab from the Hierarchy pane and drag and drop it to the prefab you created
- If it turns blue, then it has been created successfully

Exercise #07 – Design a Character

- Use a pencil and paper or a graphics tablet to sketch down the outline of the characters that will appear in your game. Examples of characters:
  - The protagonist
  - The main antagonist
  - The antagonist's minions
  - Small enemies (not directly related to main antagonist)
  - Non-playable characters (e.g. a wise old man who gives you the map, etc.)
- In 2D games, simple designs generally work best

Exercise #08 – Design Level Layout

- What does a level in your game look like? Draw a general outline of level 1 from start to finish. It doesn't have to be detailed, only broad lines to describe how it looks and where the player is supposed to move.

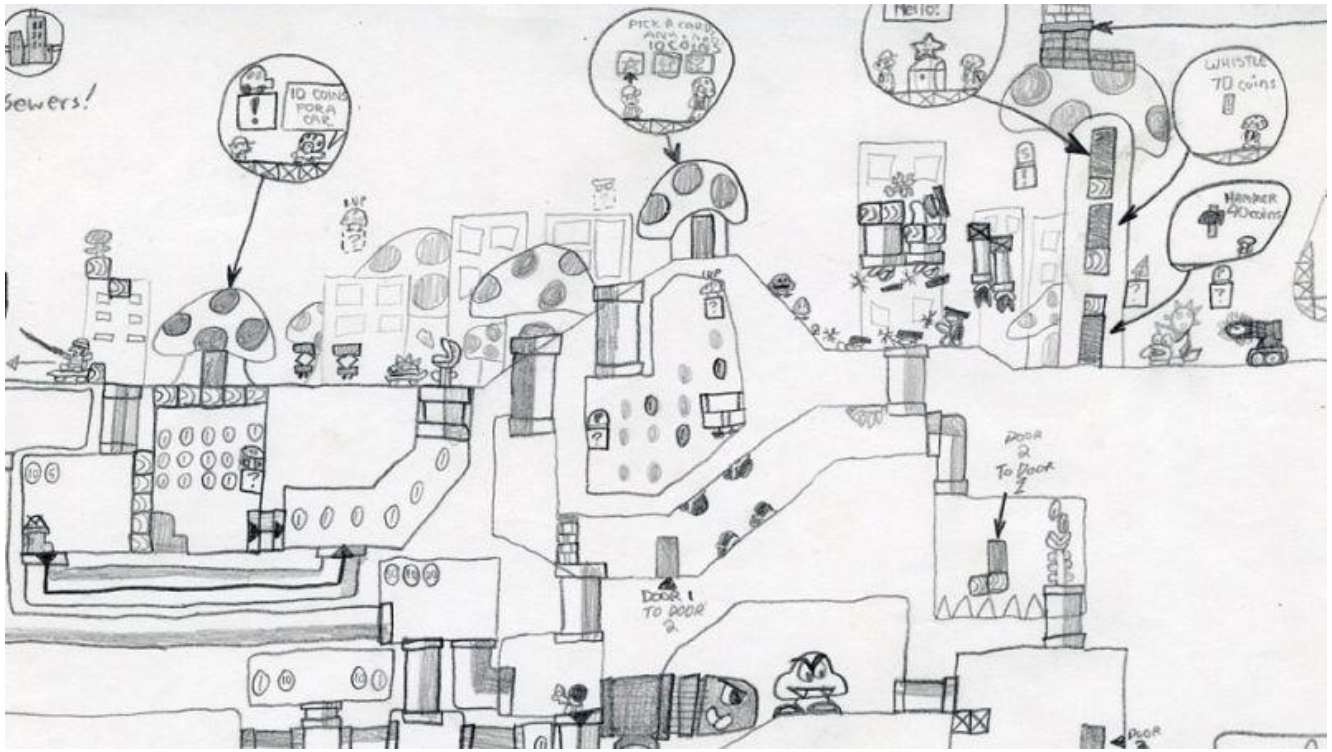


Image Source: http://kotaku.com/5902193/scott-pilgrims-creator-designed-this-adorable-mario-level-when-he-was-a-kid

For Next Week

- Next lab we will be creating the Player Character and giving him/her basic movement
- Prepare a sprite sheet of your character containing the poses and movements he/she will need in the game (e.g. idle, walk, run, die) and bring it with you
  - If unable, go online and search for a suitable, HIGH-RESOLUTION sprite sheet to use in next week's lab
  - OpenGameArt.org is a good resource for free 2D art

Useful References

- **Game Conceptualization and Design:**
  - Johnson, M., Hasankolli, R., & Henley, J. A. (2014). *Learning 2D Game Development with Unity: A Hands-on Guide to Game Creation*. Pearson Education
  - **The Big List of Game Design. URL retrieved from:** http://www.pixelprospector.com/the-big-list-of-game-design/
  - **Jonkers, D. (2011). How to design Levels for a Platformer. URL retrieved from:** http://devmag.org.za/2011/07/04/how-to-design-levels-for-a-platformer/
  - **Game Background Art on Pinterest. URL retrieved from:** https://www.pinterest.com/explore/game-background-art/
  - **How to design levels for a platformer. URL retrieved from:** http://devmag.org.za/2011/07/04/how-to-design-levels-for-a-platformer/
  - **11 Tips for making a fun platformer. URL retrieved from:** http://devmag.org.za/2011/01/18/11-tips-for-making-a-fun-platformer/

Useful References

- **Level Analysis and Design:**
  - Johnson, M., Hasankolli, R., & Henley, J. A. (2014). *Learning 2D Game Development with Unity: A Hands-on Guide to Game Creation*. Pearson Education.
  - **How to make a 2D Platformer – Unity Tutorial by Brackeys. URL retrieved from:** https://www.youtube.com/playlist?list=PLPV2KyIb3jR42oVBU6K2DIL6Y22Ry9J1c
  - **Beginning Level Design by Tim Ryan. URL retrieved from:** http://www.gamasutra.com/view/feature/131736/beginning_level_design_part_1.php?page=1
  - **Becoming a Level Designer and Env. Artist Part 1.URL retrieved from:** http://www.worldofleveldesign.com/categories/level_design_tutorials/becoming-level-designer-environment-artist-part1.php
  - **The Level Design Analysis Playlist by Sunder. URL retrieved from :** https://www.youtube.com/playlist?list=PLHKJTKyEfjmGegraWzCJYjbJYnIbOHi-y
  - **100 Level design Ideas and Locations. URL retrieved from :** http://www.worldofleveldesign.com/categories/level_design_tutorials/100_level_design_ideas/100_level_design_ideas_and_locations.php

Useful References

- **Game Development Tutorials:**
  - **How to make a 2D Platformer – Unity Tutorial by Brackeys. URL retrieved from:**
  - https://www.youtube.com/playlist?list=PLPV2KyIb3jR42oVBU6K2DIL6Y22Ry9J1c
  - **Unity Tutorial - 2D Side Scroller (Super Platformer Bros) Episode 1. URL Retrieved from:** https://www.youtube.com/watch?v=BdlL5bwbCiI
  - **How to make a 2D RPG Game in Unity. URL retrieved from:** https://www.youtube.com/playlist?list=PL_4rJ_acBNMH3SExL3yIOzaqj5IP5CJLC
  - **Unity Tutorial - 2D RPG Tutorial Playlist. URL Retrieved from:** https://www.youtube.com/playlist?list=PL_S_hCof18KzyQ2sl4AfjSvmo1ILCuywH
  - **Unity 2D Tile Mapper. URL retrieved from:** https://www.youtube.com/watch?v=_x0bMTxP7Yw