**Worksheet – Interactive Lab Project #1**

*Provide concise sentence form answers to the questions below.*

*Include cropped screenshots, diagrams or illustrations to help clearly communicate your answers.*

| **Score** | | **Out of** | | Question |  |
| --- | --- | --- | --- | --- | --- |
|  |  | | 1. a) | | Explain the difference between public and private fields. |
|  | **1** | |  | | Public fields are viewable within the inspector and can accessed from other classes.  Private is only viewable with the script and is only accessed within the class |
|  |  | | b) | | Give an example of where you used a private field in your project. Explain why you chose to make it a private field? |
|  | **1** | |  | | An example of a good use of a private variable is when you don’t want the variable to be seen within the inspector, you do this to avoid confusions with both yourself and fellow coders. |
|  |  | | c) | | Give an example of where you used a public field in your project. Explain why you chose to make it a public field? |
|  | **1** | |  | | Exaples for using public varubles:   * Varble testing; you may need to adjust a variable during testing, an example would be finding the best speed for the player ie Break Force:      * Referencing an object; you may need to reference of an object, so the code knows what object your referring as reference looks at broken object in the above image |
|  |  | | 1. a) | | Identify a ***Trigger*** event in your Unity Project. |
|  | **1** | |  | | a  If the “player” moves within the outlined green box, the rest of the screen turns green. Ontrigger is just something the player “triggers” |
|  |  | | b) | | For each of the 2 overlapping game objects, list the requirements to produce the trigger message. Refer to the ***Collision Action Matrix*** in your answer. |
|  | **2** | |  | | To avoid any issue with the collision box of the trigger event by a “Collision Action Matrix” the trigger event of happen when it collides with the “player” tag |
|  |  | | c) | | There are multiple component configurations for producing a trigger message. Provide a rationale for why you chose the specific configuration in 2b. |
|  | **2** | |  | | I used “OnTriggerEnter2D(Collider2D other)” for the mere reason as it was the simplest and most effective way (that I know of) to create the trigger event. Same with comparing the tag and the camera background change |
|  |  | | 1. a) | | Identify a ***Collision*** event in your Project. |
|  | **1** | |  | | Colliding with the blue wall with enough force causes it to delete, this uses OnCollisionEnter2D(Collision2D collision) |
|  |  | | b) | | For each of the 2 colliding game objects, list the requirements to produce the collision message. Refer to the Collision Action Matrix in your answer. |
|  | **2** | |  | | Player must collide with the wall with a minimum of 5f force to break it (done by jumping from a certain distance)  else the player will only bump into the wall  It other terms using the Collision Action Matrix, if player moves fast enough into the wall, they activate the brackable script. Else only the movement script will be activated |
|  |  | | c) | | There are multiple component configurations for producing a collision message. Provide a rationale for why you chose the specific configuration in 3b. |
|  | **2** | |  | | I used OnCollisionEnter2D(Collision2D collision), for the same reason previously stated with trigger, I felt this was the most effective way to produce a collision event within both the project scope and overall goal of the project |
|  |  | |  | | Why and when is it important to use Time.deltaTime() in algorithms within Update() and FixedUpdate()?  Provide an example of where Time.deltaTime() was used in your project and explain why. |
|  | **2** | |  | | Update() is called every frame and FixedUpdate() is called opon physics occurring within the project, it important to use Time.deltaTime because without it framerates tend to fluctuate, thus, to have constant framerates you must use Time.deltaTime.  I used Time.deltaTime in the players horizontal movement as to make the movement feel smooth. |
|  |  | |  | | Provide an example of where you created a reference to another game object in the Scene.  How was the reference created?  Why was this reference necessary? |
|  | **2** | |  | | Reference was created by calling a public GameObject and was necessary so the script would know which object it was affecting |
|  |  | | 1. a) | | Provide an example of where a Prefab was instantiated in the project. |
|  | **1** | |  | | I used prefabs for the walls, floor and blue wall as I was unsure how much I’d need of each and didn’t want to continuously create box colliders |
|  |  | | b) | | List the steps you took to create the Prefab. |
|  | **1** | |  | | Steps I took   * Make square * Move square to project menu * Put necessary components on prefab * Put prefab in scene |
|  |  | | c) | | List the steps required to create an instance of the Prefab at runtime. |
|  | **2** | |  | | * Create GameObject * Drag from the Hierarchy to the Assets. This creates a Prefab Asset * Rename your Prefab * delete the original object in the Hierarchy. |
|  |  | |  | | Provide an example of adjoining game objects that were placed using Vertex snapping. |
|  | **1** | |  | | Vertex snapping is simply unity detecting nearby vertices and snaping them together.  Adjoining is making that perminate thus the objects are now attached to each other |
|  | | | | | |
|  | **22** | |  | |  |