**Things I’ve learned from chapter 1-3 from the book**

*Creating a game object:*

1. Select the GameObject on the top far left of the screen.
2. Select 3D Object (3D Objects are primitive objects such as sphere, cube, cone, etc.) or you can select to create an empty object but the first 3 chapters of the book haven’t talk about how that works yet.
3. After picking a primitive object of your choice, simply place it into the 3D space and you can adjust the size, position, or rotation by clicking and dragging the arrows on the 3D object or you can precisely input the numbers on the far right side of the screen (Inspector tab).
4. The hierarchy menu on the far left side of the screen tells you the name of the object you’ve selected and under each selection, there is an arrow, beneath the arrow are the components of the whole object.

*Attaching a script to a game object:*

1. Select Project tab near the lower left side of the screen and then select Assets.
2. Create a script by right clicking on an empty space on the right sided box next to the term “Assets”. And then select “Create” and then select “C# Script”.
3. Notice that the script will inherits another class named MonoBehaviour. I am not sure how does the class MonoBehaviour work but all new C# script needs to inherit this class and unity will automatically inherits the class for you.
4. After the script is created, simply drag and drop the script to the game object name under Hierarchy.

*Writing a script:*

1. Any public variable that is declared inside the class will appear in the Inspector tab that will allow developers to change its value using the graphical user interface.
2. When you create a script, the script will automatically have the methods Start() and Update(). The Start() method is where you initialize anything you need for the object. The Update() method will run once each time the frame refreshes when the game runs.
3. Rotation:
   1. Input.GetAxis(“Mouse Y”) or Input.GetAxis(“Mouse X”) return the displacement of the mouse in terms of the Y coordinate or the X coordinate.
   2. transform.Rotate(float x, float y, float z) rotates the object that the script that is attached to. It automatically adds the displacement (float x, float y, or/and float z) to the original angle.
   3. Transform.localEluerAngles is a Vector3 structure that sets the angle in a given vector.
4. Moving the object using the arrow keys:
   1. Input.GetAxis(“Horizontal”) or Input.GetAxis(“Vertical”) return the displacement of the left and right keys or up and down keys.
   2. Declare a CharacterControl object. Use the generic GetComponent<CharacterControl>() to initialize the object. Call the public method of the CharacterControl object Move(Vector3 vector) to transform the game object location. It automatically take parameter (Vector3) as a displacement and add it to the original location of the game object.
5. *Allow object to shoot:*
   1. Term to know:
      1. Ray: An imaginary line that cast from a specific 2D (X,Y) coordinate of the camera and projects perpendicularly from the 2D (X,Y) coordinate towards the Z coordinate.
   2. First declare the Camera object. Then call the generic GetComponent<Camera> to initialize the object. Do it in the Start() method.
   3. Declare a Vector3 object and initialize the x and y coordinate to the center of the camera. You can call Camera’s object’s public methods pixelWidth() and pixelHeight() to get the height and width of the camera. Dividing them both with 2 will get you the center.
   4. Declare a Ray object and call the Camera’s public method ScreenPointToRay(Vector3 vector) and assign the return value to the Ray object.
   5. Declare RaycastHit object, it’s an object that catch to see if anything that the ray hits. To detect whether the ray hits anything, call Physics.Raycast(Ray ray, RaycastHit raycasthit) and check if it’s true. If it’s true, then it hits something and RaycastHit object that you pass it into Physics.Raycast() will catch the object that’s hit.
   6. You can create an game object at the location that was hit. Lets create a sphere. Check the condition of Physics.Raycast() to see if the ray hits anything and then you can create a game object be declaring a GameObject object and initialize it from the return value of GameObject.CreatePrimitive(PrimitiveType.Sphere). And you can place the sphere at the location where it his by assigning RaycastHit.point to sphere.transform.position
6. I also learned how to make another game object react when I cast the ray to the target. The book asked us to make it so that when the ray casted at the target, rotate the target negative 75 degrees around the x axis without any animation. So I modified it so that it generates some sort of animation while rotating instead of immediately rotates negative 75 degrees. The screenshot below is the modified piece of the book’s code.



