Unity Game Scripting – Week 2 Lab Assignment

Before the lab:

- Download and open the Week 2 Unity project from Blackboard.
- Open the RoadCrossing scene.

During the lab:

- Work through the points below one by one.
- Keep the lecture notes/slides (from Lecture 2) and relevant Unity documentation at hand!
- Don't stay stuck for too long on one task; ask your lab teacher for help!
- Even if you solved the tasks, ask for feedback from your lab teacher – you do not need to finish all tasks before doing this!
- Don't worry if you can only finish a small part of these tasks during the lab! (These tasks are created also with self-study time in mind, and towards the end get more challenging.)

Lab assignment:

During this lab assignment, you're going to make the core of a simple single player road crossing game (like the well-known Frogger), and practice with using Unity prefabs, and get started with the basics of Unity physics.

1. *(Rigidbody / AddForce)* Currently, there's one car in the scene that moves using physics (it has a rigidbody, and its velocity is set every frame), but the player does not yet move using physics. You can see that because the player can still move through the ``buildings''. *Add a rigidbody to the player, and change the player controls to use physics: use AddForce as shown in the lecture*. Adjust the force and drag until you like the movement (there will be a bit of inertia).
2. *(OnCollisionEnter)* We want the player to respawn when he collides with a car. Add an OnCollisionEnter method to the player controls, and from there call ResetPosition.
3. *(Tags)* Currently, the player's position resets when he collides with anything, including buildings! We don't want that. Add a *Car* tag, and give the car game object in the scene that tag. In the player's OnCollisionEnter, only reset the position if the player collides with a Car (check the tag of the other game object).
4. *(Changing prefabs)* There's already one prefab in this project (in the Prefabs folder), for a building. It's instantiated four times in the scene, but it has the wrong color. Apply the Building material (in the Materials folder) to this *prefab*, and watch how all buildings in the scene change simultaneously!
5. *(Creating & instantiating prefabs)* There's a single car in this scene. Make it into a prefab by dragging it into the Prefabs folder. Next, instantiate this prefab into the scene a couple of times. Cars on the front line should go to the right, and cars on the back line should go to the left. Cars always move forward (their blue axis), so you can change the direction of the cars by simply changing their initial rotation.
6. *(Changing prefabs)* Select the prefab and fine tune some parameters, such as the car speed (it seems a bit slow).

7. *(Overrides)* In the scene, make some cars go faster than others, and give the different cars different colors, using *scene overrides.* (For the car's speed; see the SimpleCarMovement component. For the car's color, change the first material in the MeshRenderer component).
8. *(Revert & apply)* Check your overrides in the top of the inspector. Try out the *revert* and *apply* options, and see what happens.
9. *(Instantiate by code)* Create a new car spawner script. Place a new *car spawner game object* at the start of both lanes, and let them spawn cars at regular intervals (look at the *Instantiate* method for spawning prefabs, and look at *Time.time* for keeping track of spawn moments). (*Tip*: an easy and designer-friendly way of setting the (start)position and rotation of spawned cars is by simply copying the transform.position and transform.rotation of the spawner game object!)
10. *(Triggers)* Spawning infinitely many game objects is bad: create a new script called *DestroyOnTrigger* which destroys the cars when they hit *any* trigger. Add it to the car prefabs. Put *trigger* objects at the end of the lanes.
11. *(Prefab variants)* Read about *prefab variants* here: https://docs.unity3d.com/Manual/PrefabVariants.html
12. Create a *variant* of the car prefab, with a different color, and a different speed. Make sure that the back lane spawner creates faster cars, and the front lane spawner creates slower cars.
13. What happens if you change the speed of the original prefab? (Not the variant) Can you explain this?
14. *(Inserting models & nested prefabs)* While you worked on the game logic, the artist in your team (called Kenney) finally finished the car models. Create prefab variants where you replace the car's greybox cubes with your favorite car models from the KenneyCars folder. You can first select the car prefab by double clicking it, and then adding one of the car models as child to the root game object of the prefab. You can disable the renderer for the original cubes. Note that the models in the folder don't have a collider, so make sure you keep a properly sized BoxCollider somewhere! Since the car models are also prefabs, you have just created a *nested prefab.* See https://docs.unity3d.com/Manual/NestedPrefabs.html for more information.
15. *(Arrays + random)* In your spawn script, create a public array of GameObjects (instead of a single public GameObject), and fill this array with different car prefab variants in the inspector. When spawning cars, choose a random car from this array.