Unity Game Scripting – Week 3 Lab Assignment

Before the lab:

- Follow the lecture, and/or look at the lecture notes/slides.
- Download and open the Week 3 Unity project from Blackboard.
- Open the Assignment3 scene.

During the lab:

- Work through the points below one by one.
- Keep the lecture notes/slides 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.)

Lab assignment:

During this lab assignment, you're going to make the core of a simple single player "hovercraft" race game, where the player should try to finish a (greybox) track within a given time.

1. Add an "Accelerate" script to the Hovercraft game object, that accelerates / decelerates the vehicle when up / down (or W / S) is pressed, using the *AddForce* method of the attached rigid body component.
2. Add a "Turn" script to the Hovercraft game object, that directly rotates the vehicle around its y-axis, when left / right (or A / D) is pressed. (*Remark:* rotating the transform directly for a game object with a non-kinematic rigidbody attached is not ideal, similar to changing the position directly (=translating). Ideally, you should use the AddTorque method of the rigid body. However, getting responsive controls that way is much more complex, and direct rotation isn't as bad as direct translation, so you're allowed to rotate the transform directly.)
3. Add a check whether the vehicle is *grounded* to your Acceleration script. You can use *OnCollisionStay* or *Physics.Raycast* for this. Make sure that you can only accelerate and decelerate when the vehicle is grounded.
4. Experiment with the amount of acceleration force / turn speed in your previous two scripts (which you both exposed in the inspector!), and the *drag* of the rigid body component, until you get "hovercraft" controls that you like. (Note that it's more like a hovercraft than a car, since it will slide sideways when turning at high velocity.)
5. You might have noticed that getting the drag correctly is hard: either the "ground friction" is too low (leading to low acceleration or high top speed), or the "air viscosity" is too high (leading to the feeling that you're floating through jelly, instead of flying through the air, when doing jumps). Fix this by adding some extra "drag" (ground friction) by code whenever the vehicle is grounded. You can do this by scaling the rigid body velocity by a factor around 0.98 every physics update.

6. Create some loose objects that can be pushed around / fall over when crashing into them, by adding a rigid body to them. A good candidate for this is the “HighBackwall” game object. Experiment with the *mass* that you should give to (the rigid bodies of) these objects and to the vehicle, to make the collision response feel correct.
7. Create a finish trigger object, and a Finish script, that prints the current time to the Console (use Time.time) whenever the player hits this object. Use *OnTriggerEnter*.
8. Create objects that are moved by script. For instance let the MovingPlatform game object move up and down, and let the RotatingWall game object rotate around its y-axis. Add a rigid body to these objects. How should the rigid body be configured? Does the vehicle interact with these objects correctly?
9. Create a new *tag* called “Road”, and add this tag to various road piece game objects. In your Accelerate script, switch between low friction and high friction depending on whether the vehicle is on the road or off road. (Test for this tag at the same place where you test whether the vehicle is grounded.)
10. *(Advanced/optional - designers?):* Make a race track that’s fun to play by copying, moving, rotating and scaling the different road / ramp / moving object pieces. Feel free to use a terrain as well. Use prefabs for recurring elements.
11. *(Advanced/optional - engineers?*): Change your hover craft controls into *car controls.* Two essential steps are:
    a. The turn speed should be related to the forward velocity: when the car isn’t moving (forward), the car doesn’t turn.
    b. When the car is grounded, it doesn’t move sideways (unless maybe it’s in sliding / drifting mode).

    For both of these you can use the given *VelocityTools* script (check the comments in the script). For (b), experiment with adding a counter force, and the different *ForceMode* options that the rigid body’s *AddForce* method has.