Rule of thumb: When you're stuck, google with: Unity script <your problem keywords>, e.g., "Unity script mouse input"

Based on the discussion with each student, it seems that most students can follow this three-step learning path with optional content in each step.

Step 1: Setting up your environment

All students

Create a Unity account and check that you can download assets from AssetStore, which is a great resource for creating prototypes efficiently.

To work efficiently with syntax coloring and code autocompletion, it's good to use Visual Studio with Unity.

https://unity3d.com/learn/tutorials/topics/scripting/installing-tools-unity-development?playlist=17117

https://unity3d.com/learn/tutorials/topics/scripting/building-your-first-unity-game-visual-studio?playlist=17117

Here are instructions if you are a Mac user:

https://code.visualstudio.com/docs/runtimes/unity

https://www.assetstore.unity3d.com/en/#!/content/45320

If there are problems, you can also use Unity's default MonoDevelop editor, which is still better than just a regular text editor.

Students with prior version control experience:

Students with prior programming background and experience on version control systems should also check how to use version control with Unity, e.g., Git and SourceTree. Info on that:

http://adamsingle.com/unitygitsourcetreebitbucketmactute/

https://unity3d.com/learn/tutorials/topics/cloud-build/creating-your-first-source-control-repository

There is some contradictory information on whether one should use binary or text asset serialization. Version control systems work better with text files – changes made to a file by multiple persons can often be automatically merged. However, this often fails with Unity's scene files, which is why some recommend using binary for them. In my opinion, using text still helps, as merging still works for small and common changes such as tweaking property values. Multiple persons inserting and deleting objects can break the merging, however.

Step 2: Learning the basics of C# scripting

Beginners with no programming background

Go through C# beginner scripting tutorials 1-10 at https://unity3d.com/learn/tutorials/topics/scripting and the extra exercises given in MyCourses. After that, depending on your interests, either continue through the rest of the beginner scripting tutorials or move to Step 3.

Others

Check the topics of all the C# beginner scripting tutorials, go through the ones that you don't already feel familiar with. Do the same with the extra exercises given in MyCourses.

Step 3: Applying the scripting skills to creative work

Artists with no significant coding background

Rework one or more of Unity's tutorial projects with your own visuals or sounds. The projects can be found at: https://unity3d.com/learn/tutorials (e.g., Space Shooter).

For more info on how to get your art into Unity, check the following:

https://docs.unity3d.com/Manual/HOWTO-importObject.html

https://docs.unity3d.com/Manual/ImportingAssets.html

https://unity3d.com/learn/tutorials/topics/audio/sound-effects-scripting

Artists with significant coding background or who want more challenge

Create your own game prototype that uses your art.

Students with primarily a technical and coding background

Create your own game prototype using AssetStore assets, or try to reverse-engineer and copy the gameplay of an existing game, e.g., Angry Birds.

Try to select a game that makes you learn new things, e.g., network multiplayer, 3D character animation (one of Unity's strengths, see https://www.youtube.com/watch?v=Xx21y9eJq1U and the various character and animation assets in AssetStore), or game analytics.