Assignment 4

Nicolò Pinciroli

In this assignment, I have addressed some of the requirements, since some other requirements had already been addressed during assignment 2. I will now describe how I have addressed the requirements.

Sun movement animation

During the previous assignment, I had already implemented the functionalities to move the Sun by using two keys to decide the direction. Since in this case it was required to animate the movement of the Sun, I have interpreted this request as the necessity of an automatic animation. For this reason, I have created a latch variable, which is activated when the key P is pressed, and deactivated when P is pressed again. When the value is active, the sun moves at a constant speed, otherwise it stops. This feature has been included in the Sun movement script.

Insertion, update and deletion of buildings

In order to implement those features, it has been necessary to detect the mouse position. In the case of insertion, it is not possible to insert a building on top of another building, and the mouse position detection is performed using raycast. The user positions the pointer at the desired position and presses L to insert a landmark or B to insert a building.

As for the deletion, this happens when the user right-clicks on an existing landmark or buiding, by destroying the object under the cursor.

As for the updating, the user can change the position of a building in the scene by clicking on it with the left button and dragging it to the desired position.

Those features can be applied virtually to any object, but in this case they are allowed only for generic buildings and landmarks, in order to address also part of the last requirement.

Save and load scenes

Saving and loading the scenes can be done in several ways. Currently, I have tried several plugins to do so at runtime, but none of them has shown good results in this case.

For this reason, I have implemented scene saving by assuming that what I had to save was the position of the buildings, of the pavement and of the park, which are the parts that can possibly change from one case to another. The initial scene, indeed, is generated randomly and the user can move the buildings, add new buildings and so on, as previously described.

Moreover, I have assumed that the scene had to be imported again in the same simulator, so it has not been necessary to save, for instance, the Sun.

Differently from the most common approach, I propose to save the scene by generating the C# code which generates it. The reason why I have preferred not to save all the possible information about the buildings, for example, is that in fact I change only some of the properties, and saving those properties allows to recreate the same scene as the initial one.

In order to export the scene, the user can press O, and a cs file is generated. That file is already in the Assets folder in Unity, and it is already associated to the main camera, so when the user presses J (s)he can load the previously saved scene, which completely replaces the current one. However, it may be necessary, depending on the Unity version, to reload the files before seeing the new generated code.

In this way, the saved scene does not take much space and can be possibly used for other Unity projects as well, if the textures and the scripts are imported as well.

Zoom and pan

In this case, I have implemented this feature using other keys on the keyboard. Actually, this feature was already present in the previous version, where I have described it in more detail.

The reason why I have chosen to use keys is that the mouse already has other functions in this case, and the user may be confused by the multiple actions associated to the same buttons.

Textures

Textures were present also in the previous version, but here I have placed them also on the buildings. The rationale behind texture mapping is the same, and has already been described in the previous assignment.

Sky exposure

As for sky exposure, this is calculated when the user clicks on the pavement with the right button of the mouse. In this case, some rays are generated, and the proportion of rays that don’t intersect anything is the sky exposure value. The rays are generated only

for positive y, since it is not meaningful to consider the intersections with the pavement, and spherical coordinates are used. The angle between different rays is of 5 degrees, which allows a reasonable precision and a reasonable computational time, while 1 degree of separation would not have guaranteed the same computational time, for example.

Landmark visibility

Landmark visibility is computed as in the previous assignment, but in this case I loop through all the possible landmarks. Moreover, when the computation start I remove the previous color from the buildings, so that there is no overlap with previous computations. The visibility is calculated by using the V key and the visible buildings are colored in red.