**Computer Animation Production 1**

**Maya Python Scripting Assignment Report**

**i7208422**

**Sim Chun You**

**Contents**

**- User Manual**

**- Introduction**

**- Algorithms**

- Object Placement

- Traffic Simulation

**- Flow of control**

**- Results**

**User Manual**

Detailed examples of how the program can be used are in the artefacts\Videos section.

Create a project and import the textures into the source images folder.

To run the code, copy the code into the script editor and run it.

To enable the AI system, run in the expression editor: **python(“update(“+frame+”)”);** and make sure the renderer is set to: **Legacy Default Viewport**

**Introduction**

Inspired by city building games such as Sim City, I tried to create a python program that allows the user to customise his own building or use existing ones and create his own city by placing objects along grids.

**Algorithm**

**Object Placement**

For the placement algorithm, I need to find the coordinates and rotation of a selected face to place an object.

First, I calculate both the normals and the coordinates of the face centres of the selected face by using the face points (Figure 01). Next, I did 2 different versions of the dot product rule to get the rotation values. (Figure02, Figure03). The first method involves looking at the normal of the face in 3D space and rotating accordingly. The second method involves representing the normal of the selected face as 2 2D vector. These methods give different results as shown in (Figure04).

Figure01

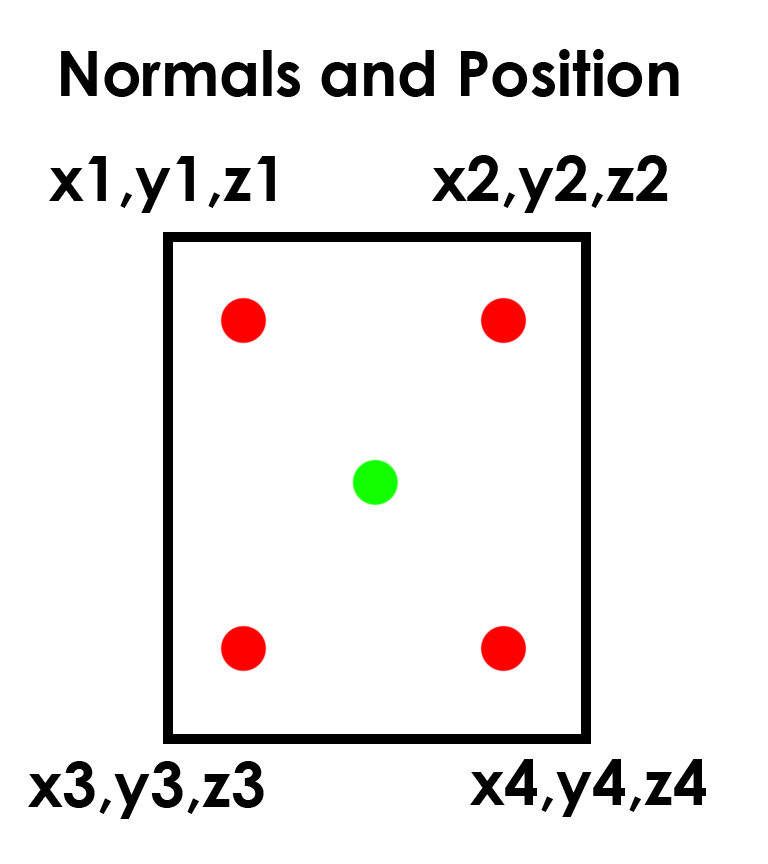


Figure02

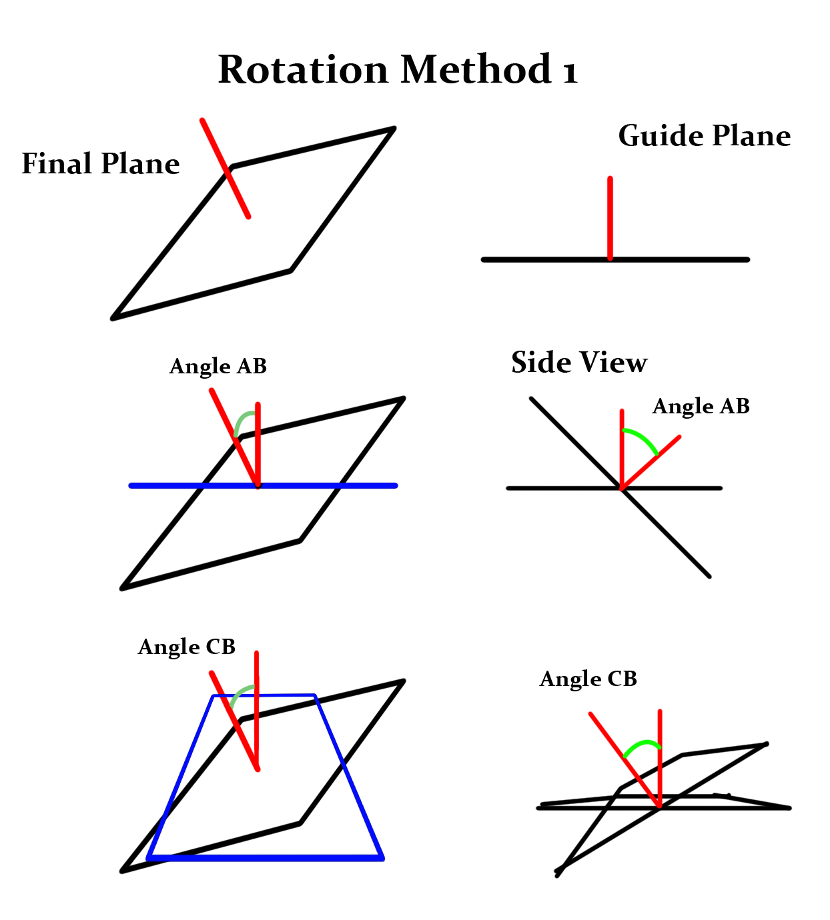


Figure03

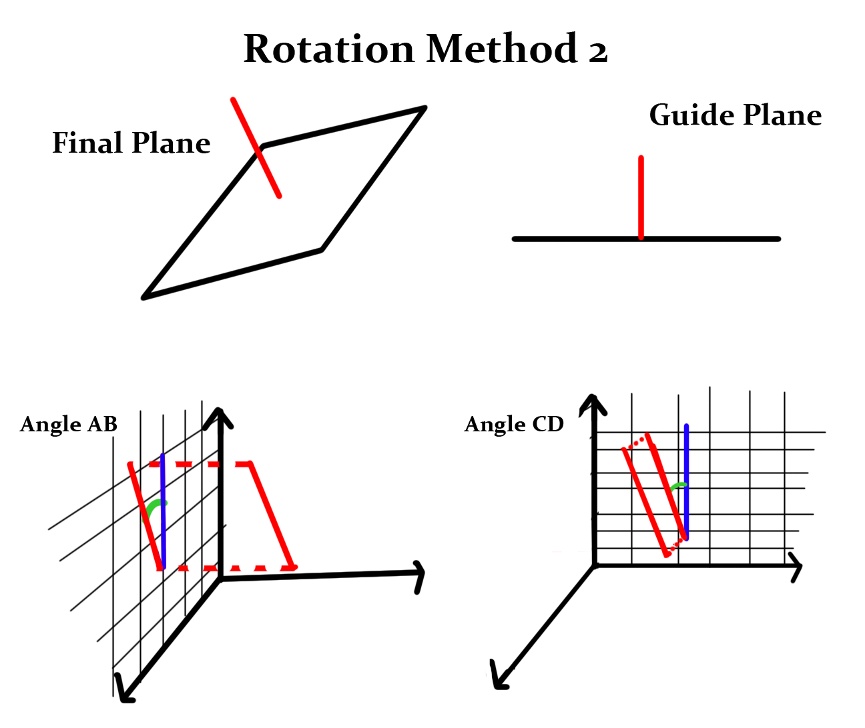
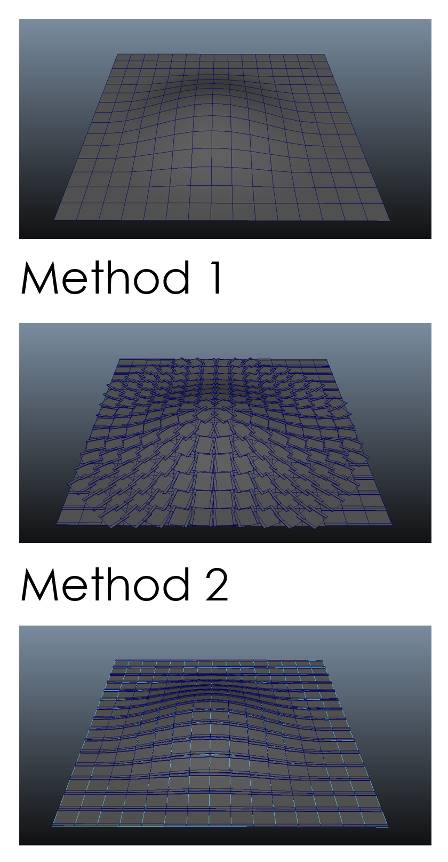
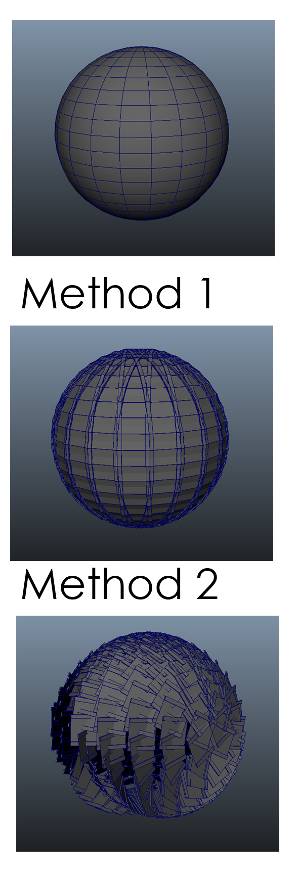


Figure04



**Traffic Simulation**

For the traffic simulation to work, I tried using the concept of finite state machines. This is done by having an object undergo various states and performing various actions depending on the conditions that it will meet.

I used Ramesh Balachandran’s approach to creating traffic simulation. I create several lists with each element of the list representing an object as well as other information needed for calculations (Figure05, Figure06). With the expression **python(“update(“+frame+”)”);** being created in the expression editor, every time I move the time slider, the function ‘update’ will rerun, allowing all calculations to be recalculated and all lists to be updated. For each time the update function is run, the code check whether the car is in any junction, if it is near any traffic lights, cars etc and if true, it will react accordingly while updating its state in the list(Figure07, Figure08). A more detailed representation of the update function is shown at the bottom of the report. For a quick summary of the calculations done in the update function, each time the function is run, it will check and update the state of the traffic lights, check and update the rotation and position of the cars and check if a car is near a traffic light, another car or at a junction.

Figure 05



Figure 06

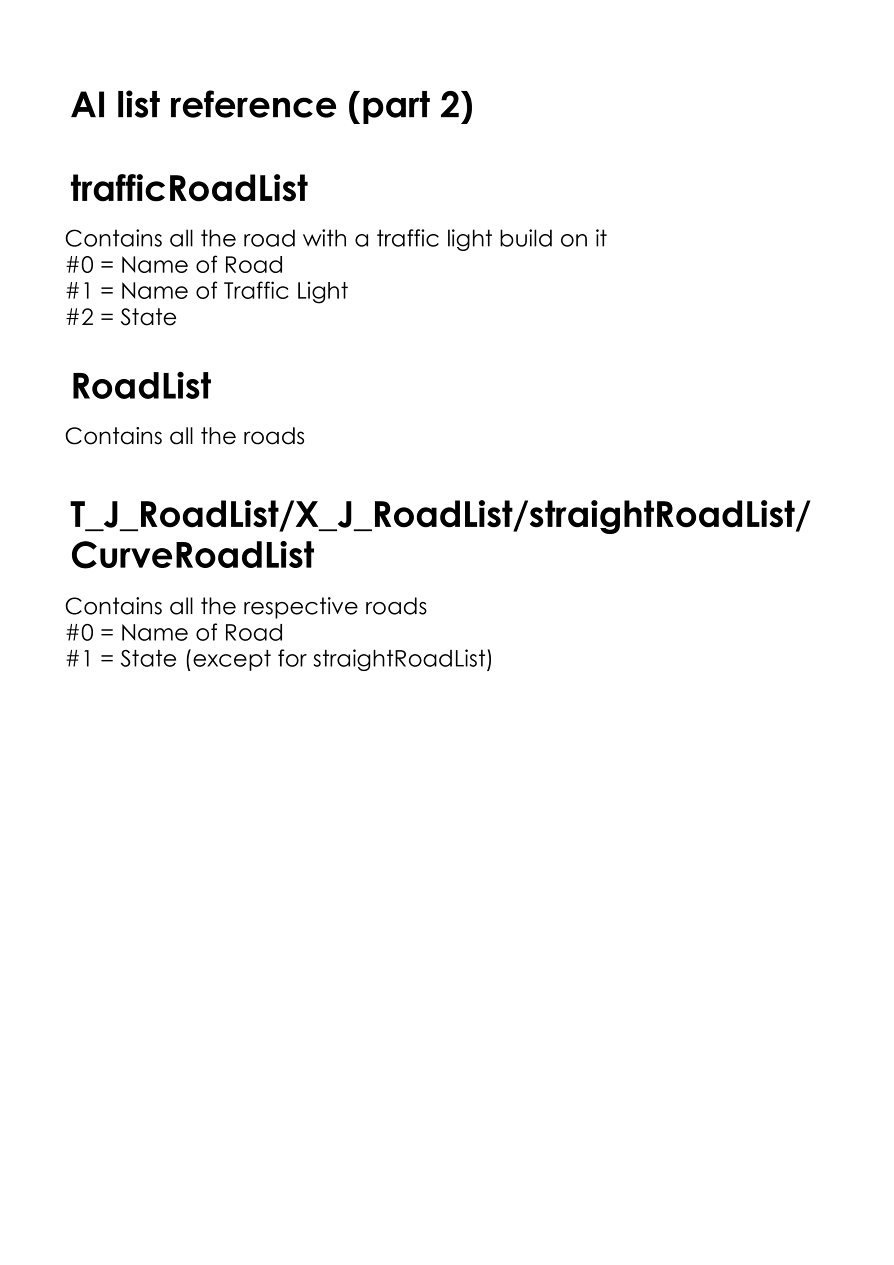


Figure07

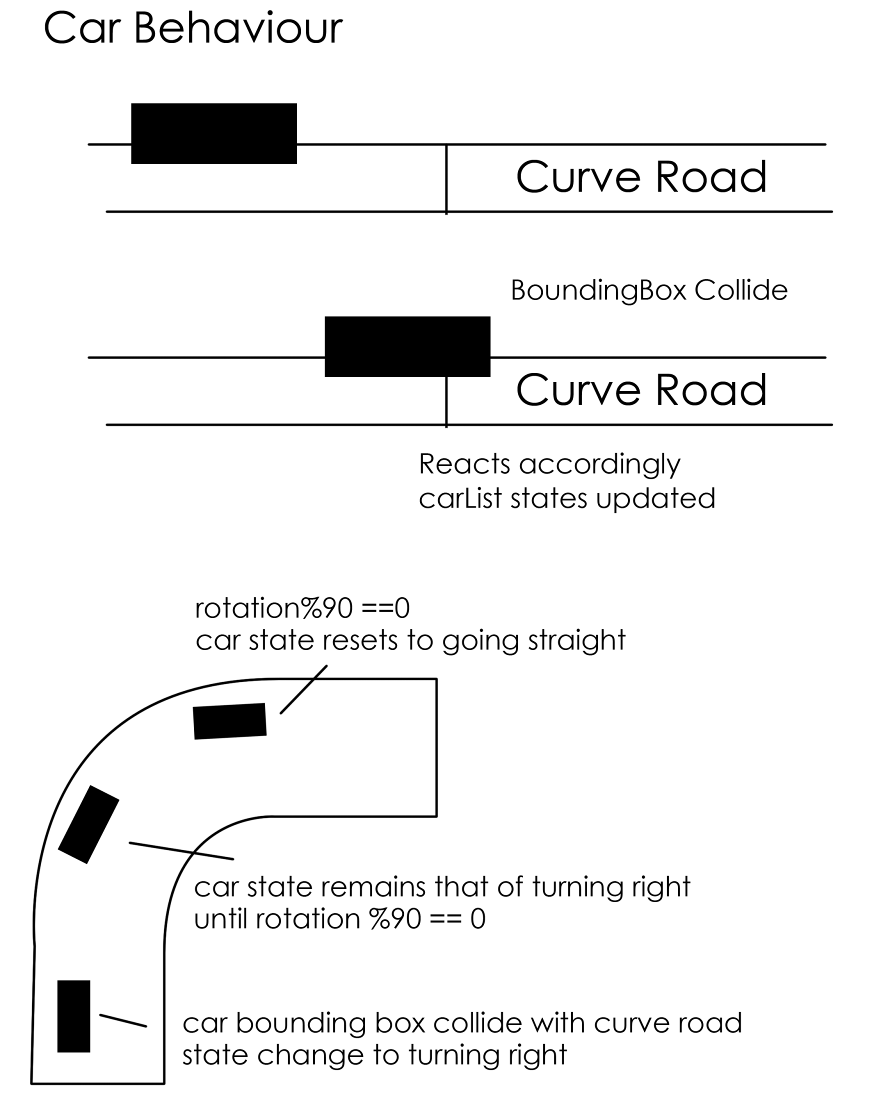
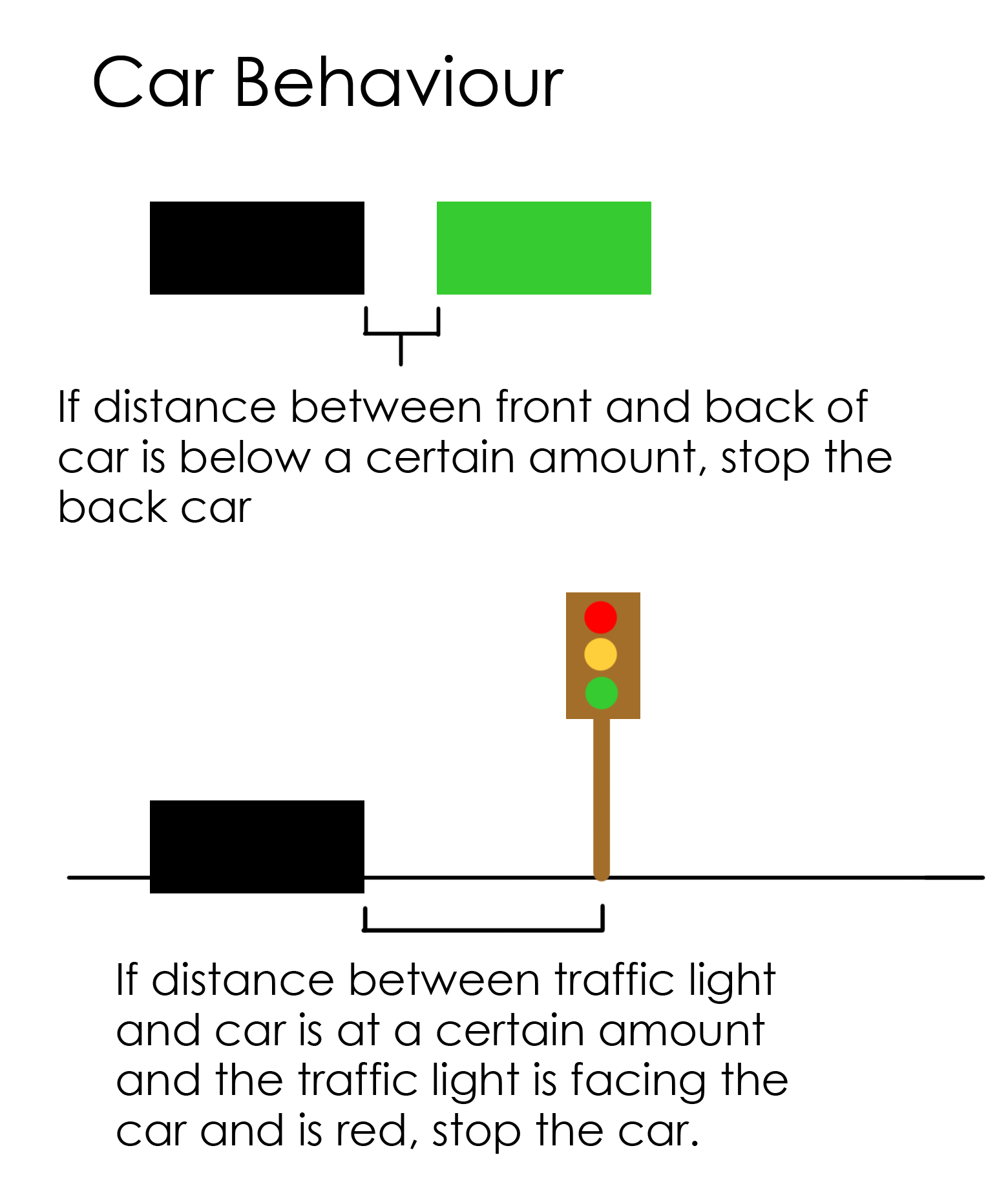
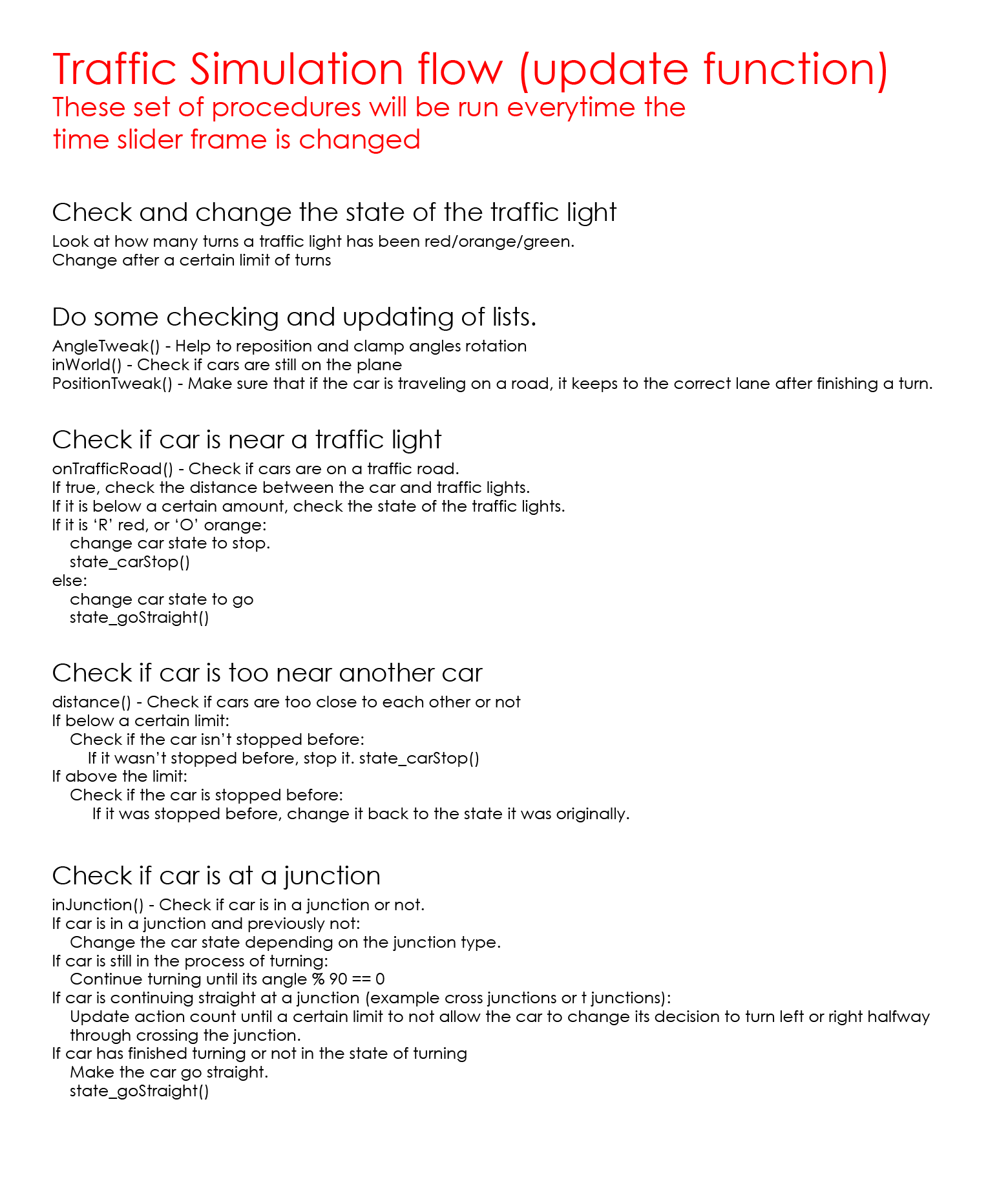
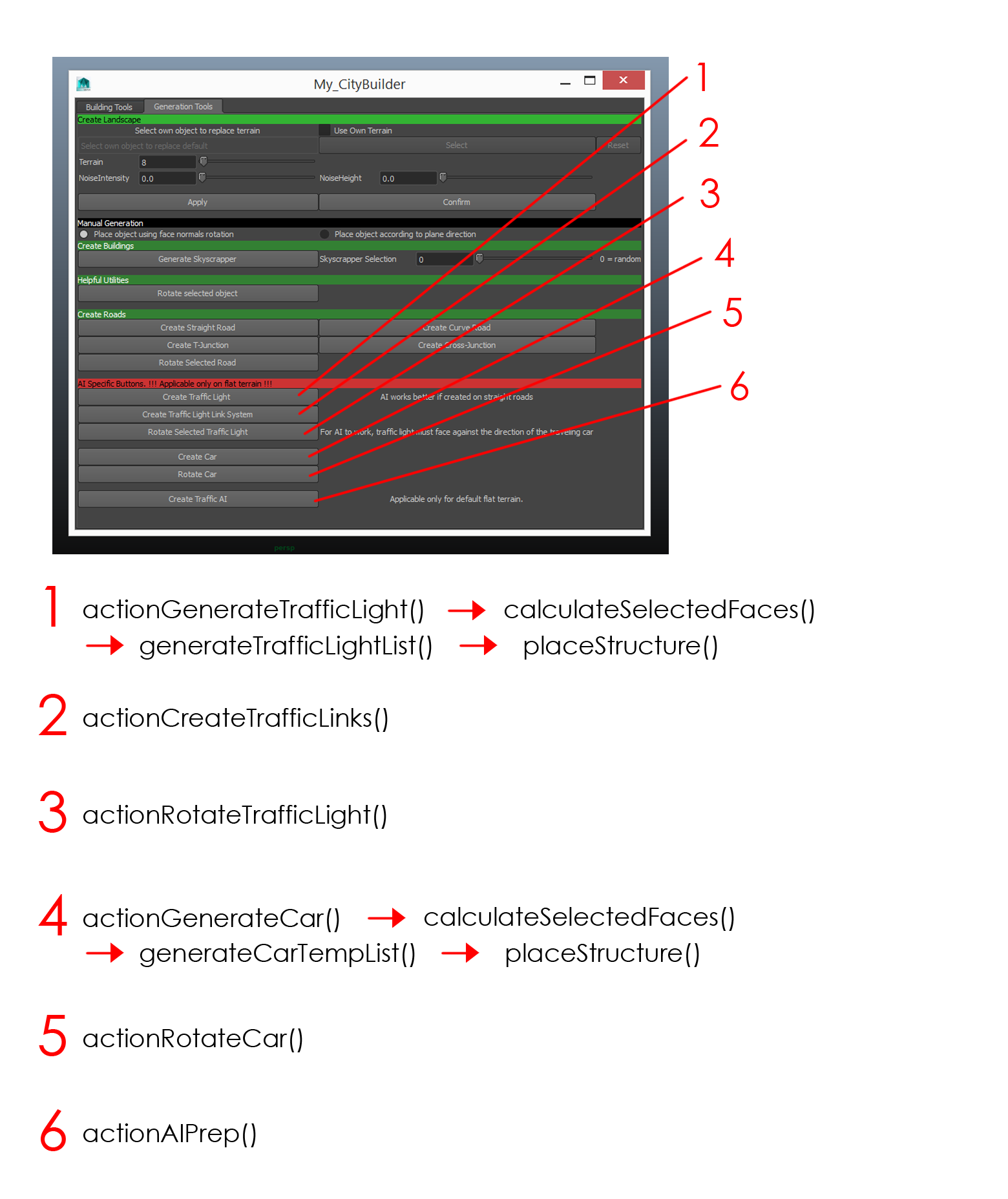
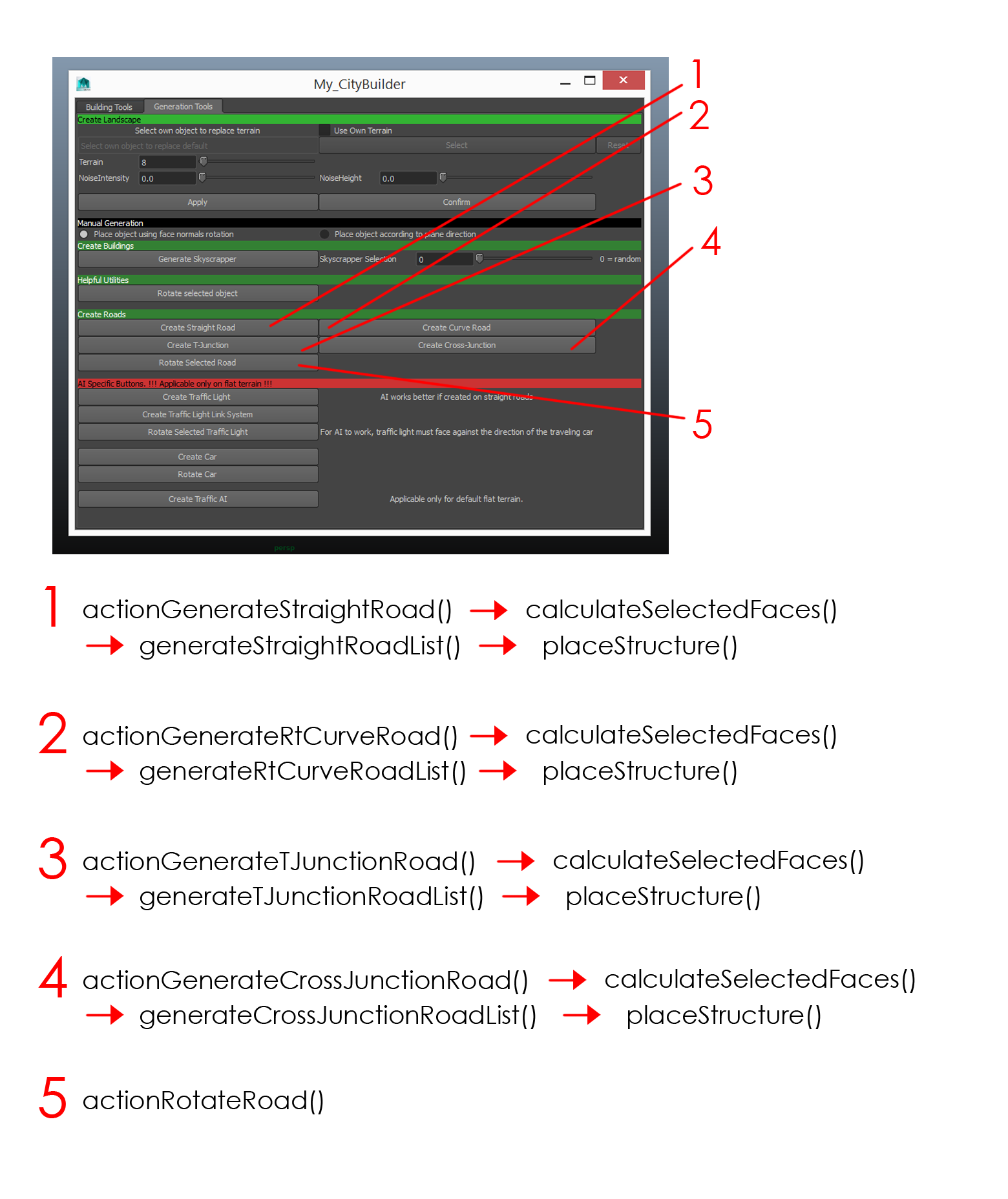
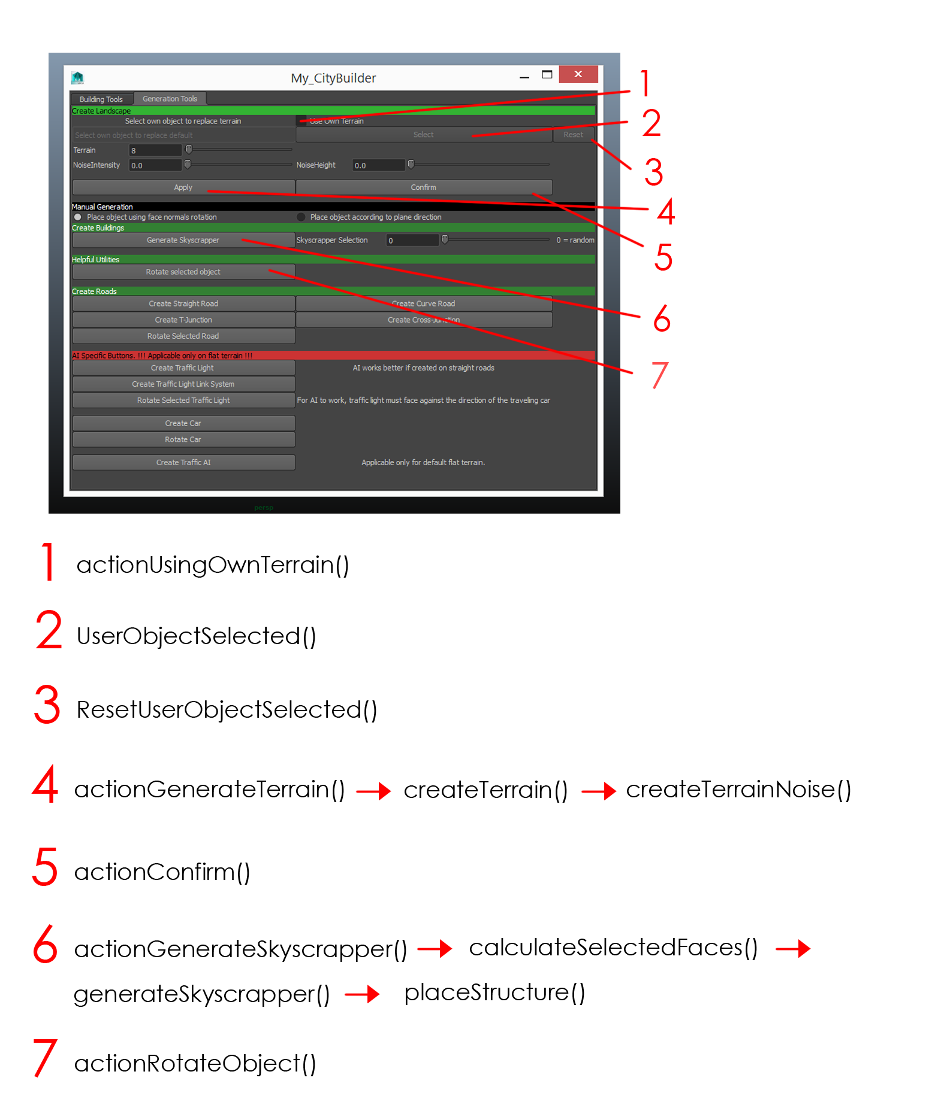
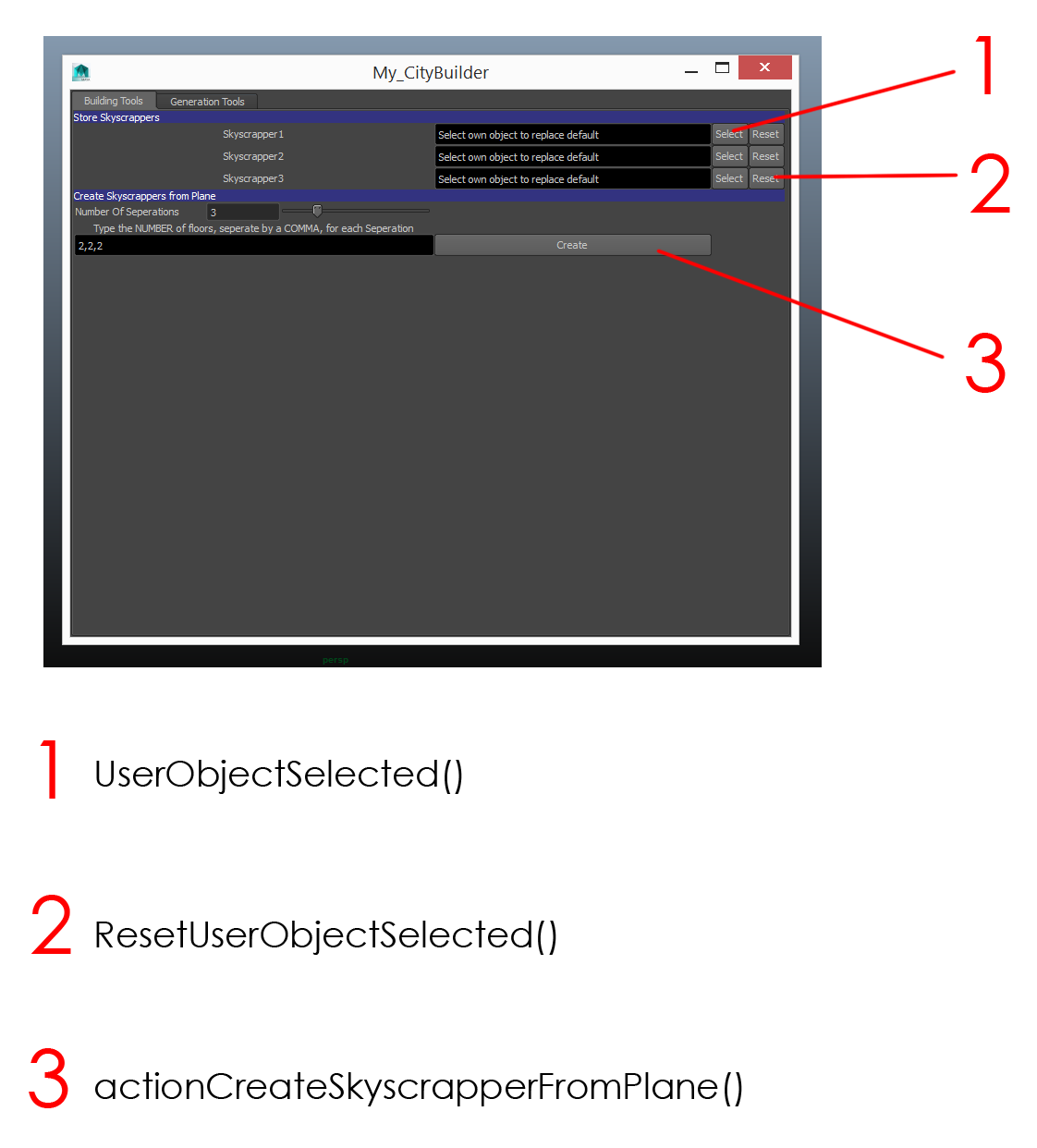


Figure08

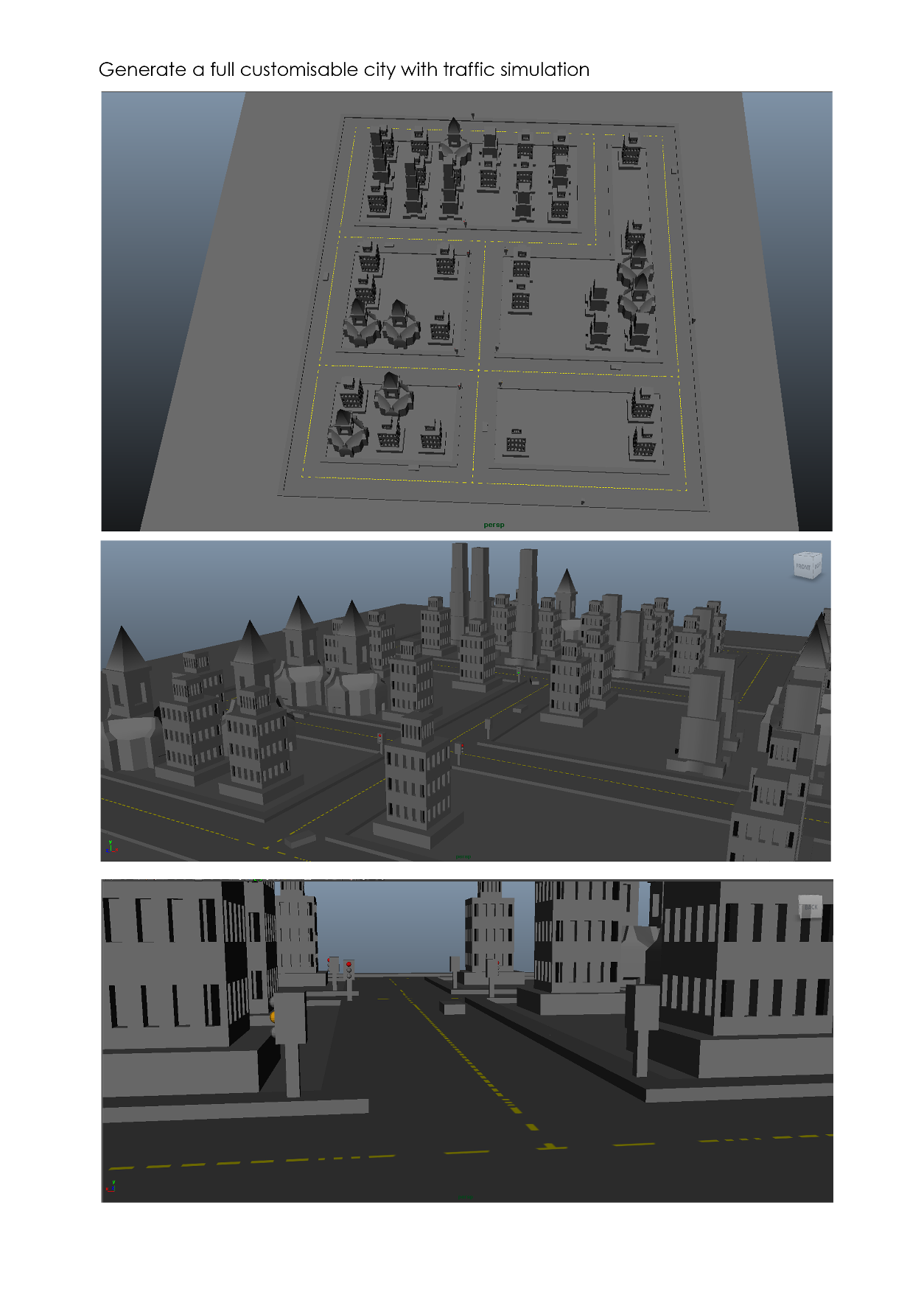
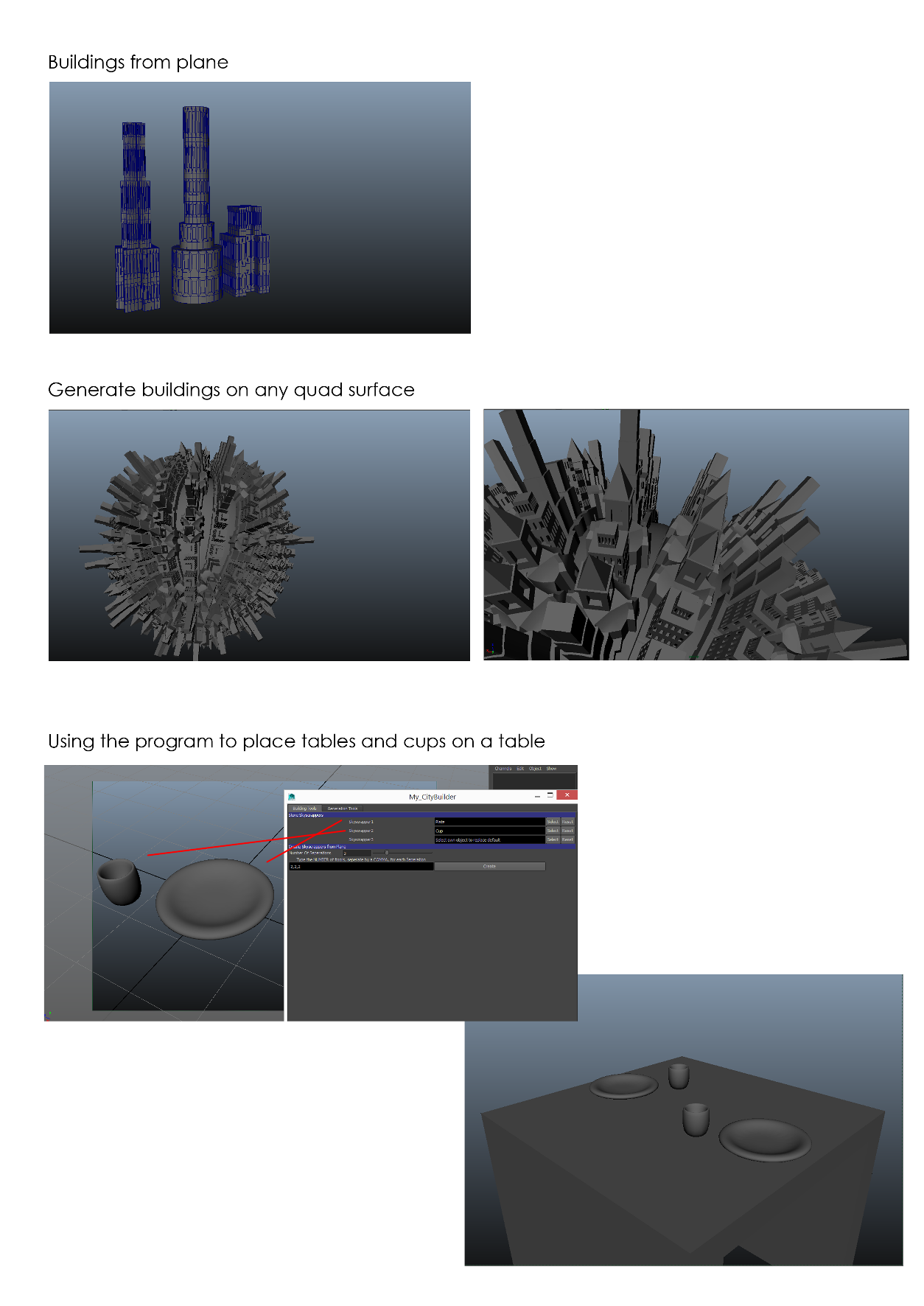


**Flow of control**

The flow of control from the user interface to the various functions are as shown below.



**Results**



**Bibliography**

Balachandran,R.,No Date.*AI\_Script\_Final.py*[online]. Available from: <https://drive.google.com/folderview?id=0By0_WhpQpCZlWmNmaHYtMi1SbTQ&usp=sharing> [Accessed 5 April 2015]