Assignment Brief 2 Documentation:

**Game Level requirements:**

At least 1 object labelled ‘player’ in object name. Preferably with a ground of some sort. Do not put more than 1 ‘player’ object. ‘player2’ is required for a network level. The network level is a separate script from the single player level lua script. Single player lua script ‘elements.lua’, Multiplayer level is ‘networklevel.lua’. **Network level only supports 1 level with 3 objects total, label levels as level 0.** Single player level supports multiple levels, never ending.

**Mechanics of game:**

Do not fall below screen dimensions (800x600), victory condition is triggered if any player falls below (x, 600) (SFML coordinates count (0,0 from top left). Make levels in mind with this condition, therefore no objects below 600. Players can fly throughout the level infinitely, and move left to right.

**What can be generated:**

Players (1 and 2), NPC’s (dynamic) , environment objects (static). Only 1 shape is able to be produced (4 sided shape such as square or rectangle). Do not overlap shapes.

**Properties of level editor:**

Each element has 10 properties: e\_type, density, friction, posX, posY, sizeX, sizeY, textureName, ObjectName, level.

Int e\_type:

e\_type determines what type the object will be, static or dynamic. 1 represents dynamic, 2 represents static. Dynamic means object can move, static means object cannot move. All objects can stack on each other regardless of their type.

Example: e\_type = 2,

Float Density:

Density is more important for dynamic bodies, however for static bodies it still needs a value, set as 0.0 for static bodies. Density determines how easy it is to push the physics body, density determines how easily manoeuvrable the object is as well, more dense objects will be very slow to move on the ground.

Example: density = 0.0,

Float Friction:

Friction is used to make objects slide along each other realistically. Determines how objects will slide off it. Friction is usually set between 0 and 1.

Example: friction = 0.0,

Float posX & posY:

Determine where the object will spawn on the level, objects take after the fact or top left corner being (0.0,0.0). Do not set Y to be 600.0 or greater, object will be detected as passed goal and travel to next level.

Example: posX = 600.0,

Example: posY = 0.0,

Float sizeX & sizeY:

Determines how big the object is, if you want a horizontally wide rectangle, adjust the y value, x determines vertical size. Textures will scale to this size, meaning if you assign a cube texture to a rectangle, the texture will stretch out.

Example: sizeX = 10.0,

Example: sizeY = 16.0,

String textureName:

Determines the texture the game will look for to assign to the object, texture name must be based off a real texture in the game files, must add filetype to end of it. Use PNG files to ensure texture will work. Make sure texture name has .png at end of it, for example “block.png”. Do not give texture a name greater than 15 characters for the network level, will make network packets larger.

Example: textureName = "box.png",

String objectName:

Determines what object will be called inside the game. Level must have an object called ‘player’. Network games need an object called ‘player2’. Do not give object a name greater than 15 characters for the network level, will make network packets larger.

Example: objectName = "player",

Int level:

Determines what level specific object is on. Levels must be in order, if there is a level 1, for there to be another level, next level MUST be 2 and so on. No level 1 objects, then level 3 objects, without level 2 objects, the level 3 objects will not be loaded.

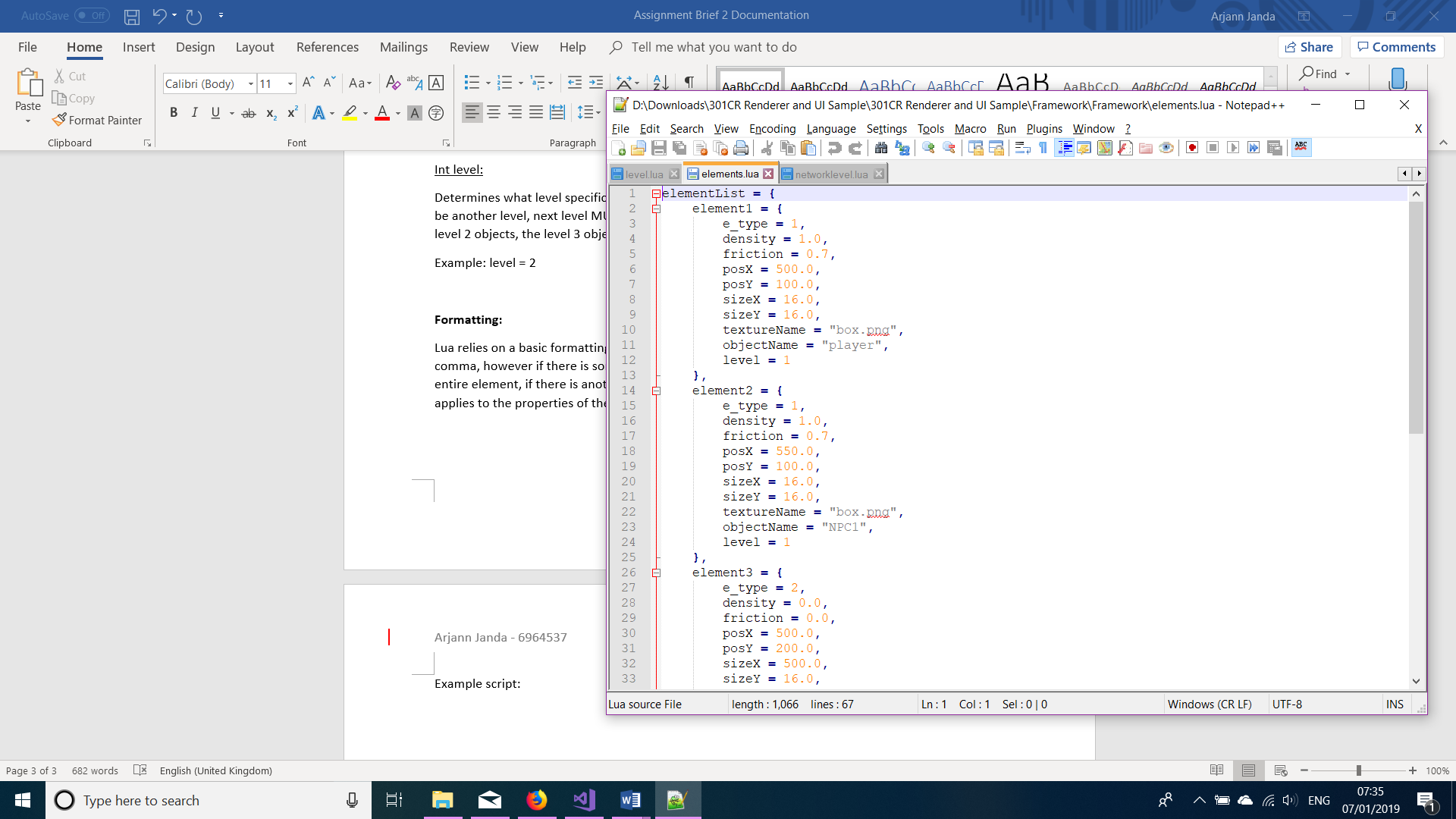
Example: level = 2

**Formatting:**

Lua relies on a basic formatting, if there is no following value, that final value does not need a comma, however if there is something after, it requires a comma. For instance, when declaring an entire element, if there is another element to follow, it will end with a comma, else it won’t. This applies to the properties of the elements inside it as well.

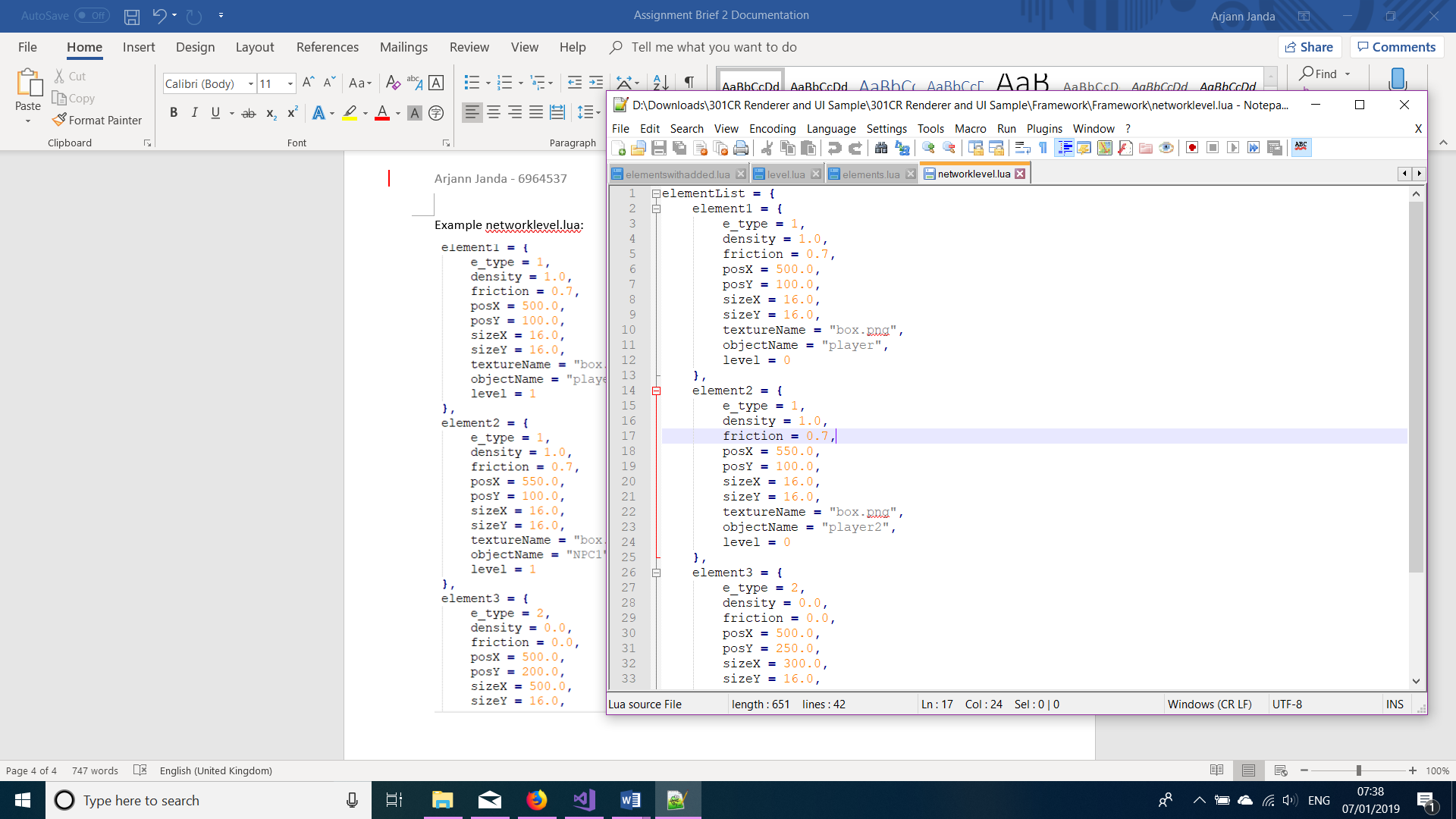
Example script:

elements.lua:



Note how element 1 needs a comma after the bracket, also each property inside element 1 follows the same rule as described above. Elements are defined by an = with {}, in between the {} will be the properties of the element. Do not name 2 elements as the same name, elements must follow as element1, element2 etc as the image demonstrates.

Example networklevel.lua:

Note the level = 0, also player and player2 are a necessity. Online levels only support 3 objects. Going higher does not guarantee the other player will receive the level.

**Software to make levels:**

Any text editor that can work with .lua is sufficient, recommended is notepad++.