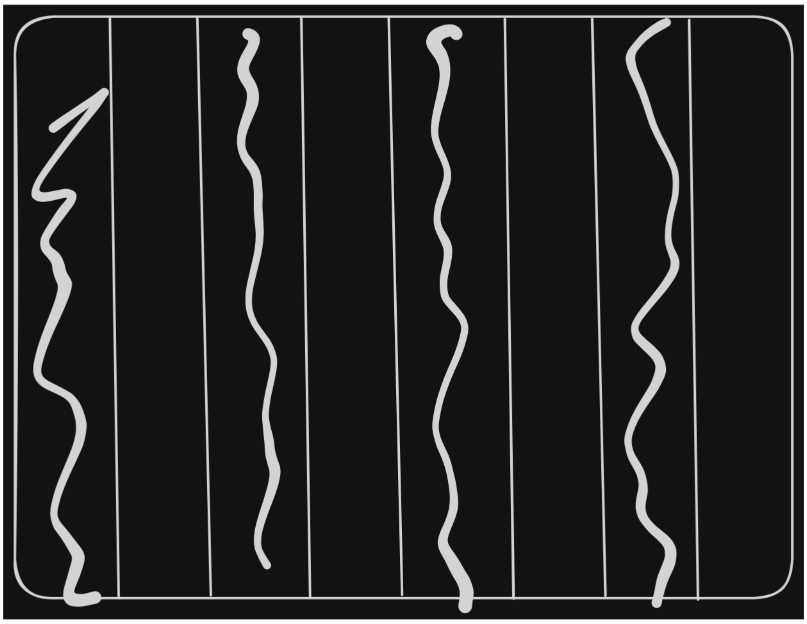
Blender Sim Report

Drone Movement and navigation

Algorithm

- The Drone divides fly zone into alternative strips.



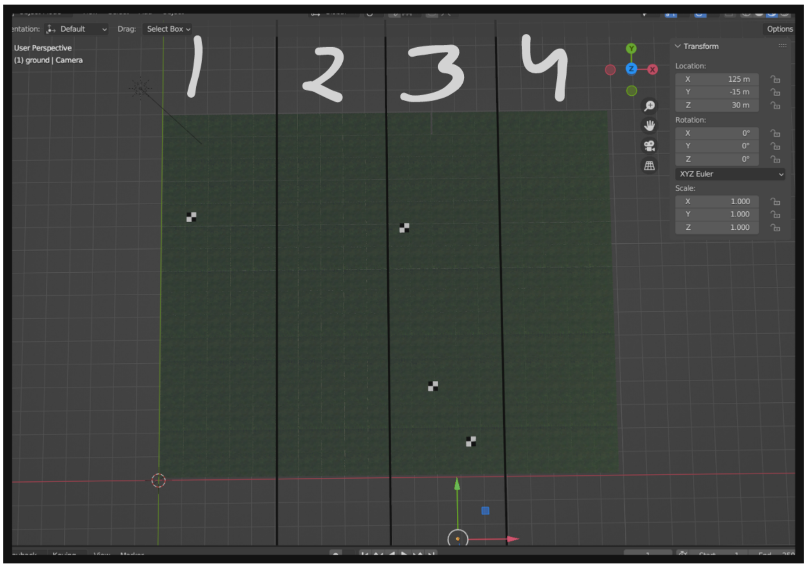
- The Drone first flies through the first set of "columns", and scans for hotspots.
- The will capture the area as it goes, storing the images for locally processing, or processing in real time.
- If the drone finds all hotspots in first set of columns, well and good, else it will move to other set of columns.

```
import random
import bpy

scene = bpy.context.scene
scene.render.image_settings.file_format = 'PNG'
scene.frame_set(1)
fp = scene.render.filepath

targets = ["Plane","Plane.001","Plane.002","Plane.003"] #get a list of all
targets
x,y = [], []
for i in range(len(targets)):
    cord = random.uniform(6, 190) #while generating co-ords, we
                                                            #take a buffer of 1
meter,
                                                            #half of the "target"
side
```

```
    while cord in x:
        cord = random.uniform(6, 190)
    x.append(cord)

    cord=random.uniform(7, 153)
    while cord in y:
        cord=random.uniform(7, 153)
    y.append(cord)

for i in range(len(targets)):
    bpy.data.objects[targets[i]].location[0] = x[i]
    bpy.data.objects[targets[i]].location[1] = y[i]


cam = bpy.data.objects["Camera"]
cam.location = [25,15,30] #position of cam is not exactly 0 as there is no ground apart
#from given area

#from here we can see that the camera will cover approximately 3 tiles. A bit less,
#but since the unchecked area is less that size of a hotspot, we will overlook it.
#so the size of 3 tiles is 3*10, i.e., 30m. So, the camera will that a step of 30m.
y_step = 30
#same for x, 50m steps
x_step = 50

def move_cam(sum):
    if sum=="ahead":
        cam.location[1]+=y_step
    else:
        cam.location[1]-=y_step

while cam.location[0]+100<200:
    while cam.location[1]+30 < 145: #hardcoding the limit again
        scene.render.filepath = r"C:\Users\2002t\Documents\sae\blender-renders\\"+str(cam.location[0])+str(cam.location[1])+".png"
        bpy.ops.render.render(write_still=True)
        move_cam("ahead")
    cam.location[0] +=x_step*2
    while cam.location[1] > 0: #hardcoding the limit again
        scene.render.filepath = r"C:\Users\2002t\Documents\sae\blender-renders\\"+str(cam.location[0])+str(cam.location[1])+".png"
        bpy.ops.render.render(write_still=True)
        move_cam("back")
    956
```

The above code takes a camera object (as a drone), and moves it through the flying zone, in the first set of columns.

For example:



ɴtation: Default
Drag: Select Box

Transform
Location:
X 125 m
Y -15 m
Z 30 m
Rotation:
X 0°
Y 0°
Z 0°
XYZ Euler
Scale:
X 1.000
Y 1.000
Z 1.000

In this fly zone, the drone will search the columns 1 and 3. Since there are only 4 columns, it will scan the two columns first and stop. Then it will process the images to calculate the number of hotspots it can identify.
In this case we can identify all 4 of the hotspots. Hence, the drone will stop. If it cannot identify, or partially identifies a hotspot, it will scan the remaining column.

Image recognition implementation

Available options for Image:

- Tiny Yolo Darknet - locally
- Single Shot detector - locally
- Full AI Model - Ground station
- No AI model - instead search for specific color values and see if they meet the minimum requirement of a hotspot to be present in a picture.