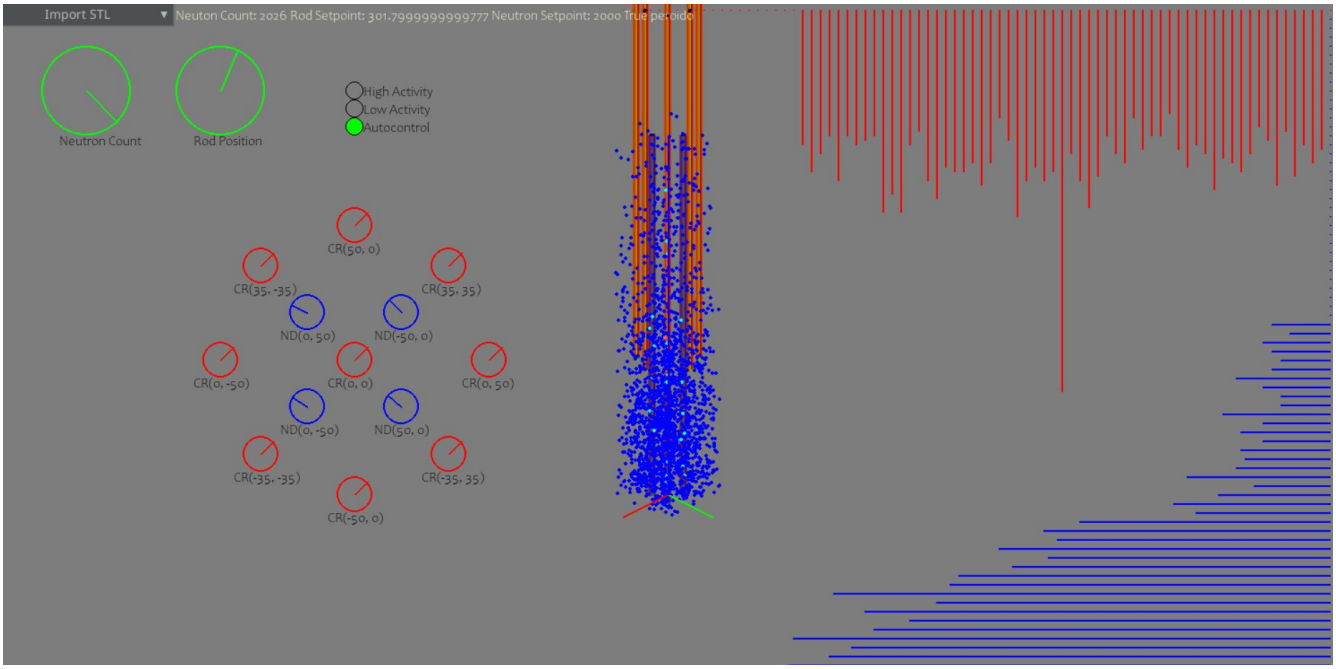
The documentation



View:

The viewpoint is fixed in an isometric mode, and the camera cannot be rotated. Do to a lack of raytracing the most recenrly imported STL will always be placed in front of others on the screen. Neutrons however are always placed in front of the STLs. Custom UI elements can be placed to do whatever you want such as Dials and Lights. These elements are defined by the UIELEMENTS.py file and are usually placed in the UI function.

Importing Objects:

Only ASCII format STLs can be imported at this time. To do so you select: File >> Import STL >> Select File. From here the STL will automatically be placed with default settings, at coordinates 0,0,0, as a “None” type, with no name.
To modify the objects open the Edit >> Modify STLs dialog box which will allow you to movify anything to your hearts desire.

List formats:

For editing the autocontrol, manualcontrol, or UI functions you may need to know this.

STLS

i[0] Metadata
i[0][0] ID
i[0][1] Class
i[0][2] Name
i[1] positon

i[1][0-2] x,y,z
i[2][x] faces
i[2][x][0] normal
i[2][x][1] triangles
i[2][x][1][0-2] verticies
i[2][x][1][0-2][0-2] verticiesx,y,z
i[3] boundries for optimization
i[4] additional data

Neutrons
i[0][0-2] position x,y,z
i[1][0-2] velocity x,y,z

UI elements are tbd idk

Neutron Logic:
Inside of STLs classified as “Control Rod” the neutron will be immeditly deleted. Inside of “Neutron Detector” the neutron will have a low chance of being deleted, but its movement will also be measured and saved inside the STL list in position [4][1]. Inside “fuel” for every frame there will be a small chance of fission where the neutron will be deleted and 3 fast neutrons will take its place with random vectors and speeds. “Moderator” will act to slow down the neutrons which will increase the fission chance inside the “fuel”. “Coolant” acts as moderator does but also takes measure of the themeral energy. I want you to just guess what “Reflector” does.

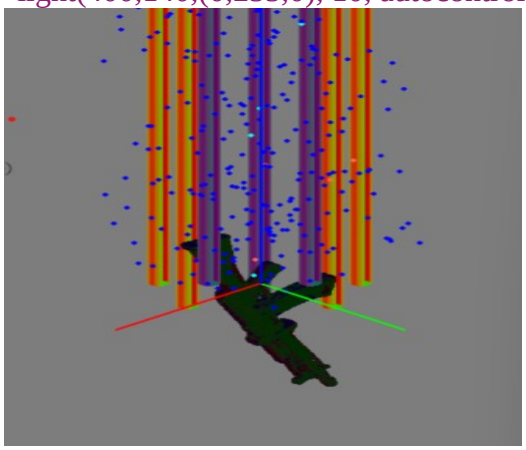
Saving/Loading:
Selecting File >> Save will maintain neutron positions, STLS, and UI elements.
Big surprise, you can reopen these with the open function

Controls:
By default, Z acts to toggle autocontrol, Q/A will zoom, E/D will scroll up and down. W/S are userdefined in the autocontrol or manualcontrol functions, usually to set rod position or neutron setpoint.

Example functions:
```
import pygame
from UIelements import dial
from UIelements import light
```

```
neutroncount =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
#Yes I know I could have done this better don’t judge me

def autocontrol(neutrons,importedSTLs, neutronsetpoint,rodsetpoint):
    neutroncount.pop(0)
    neutroncount.append(len(neutrons))
    rateofchange = neutroncount[len(neutroncount)-1]-neutroncount[len(neutroncount)-51]
    setrateofchange = (neutronsetpoint - len(neutrons))
    if  len(neutrons) > neutronsetpoint+500:
        if rateofchange < -30 and rodsetpoint < 300:
            rodsetpoint += .1
        elif rateofchange > -30 and rodsetpoint > 0:
            rodsetpoint -= .1
    if  len(neutrons) < neutronsetpoint-500:
        if rateofchange < 30 and rodsetpoint < 800:
            rodsetpoint += .1
        elif rateofchange > 30 and rodsetpoint > 0:
            rodsetpoint -= .1

    if  len(neutrons) > neutronsetpoint:
        if rateofchange < -10 and rodsetpoint < 300:
            rodsetpoint += .1
        elif rateofchange > -10 and rodsetpoint > 0:
            rodsetpoint -= .1
    if  len(neutrons) < neutronsetpoint:
        if rateofchange < 10 and rodsetpoint < 800:
            rodsetpoint += .1
        elif rateofchange > 10 and rodsetpoint > 0:
            rodsetpoint -= .1
    return rodsetpoint



def manualcontrol(neutrons,importedSTLs,rodsetpoint):

    return rodsetpoint

def controls(autocontrol, rodsetpoint, neutronsetpoint, neutrons,importedSTLs, mdisplay):
    dial(100,100,50,len(neutrons),True,mdisplay, "Neutron Count",(0,255,0))
    dial(250,100,50,rodsetpoint/8*3,True,mdisplay, "Rod Position",(0,255,0))
    NDs = []
    CRs = []
    for i in importedSTLs:
        if i[0][1] == "Neutron Detector":
            NDs.append(i[0][0])
        if i[0][1] == "Control Rod":
            CRs.append(i[0][0])
```

```
    #print(NDs)
    for i in range(0,len(CRs)):
        dial(importedSTLs[CRs[i]][1][0]*3+400,-importedSTLs[CRs[i]][1]
[1]*3+400,20,importedSTLs[CRs[i]][1][2]/800*360,True,mdisplay, f"CR{importedSTLs[CRs[i]][1]
[1],importedSTLs[CRs[i]][1][0]}",(255,0,0))
    for i in range(0,len(NDs)):
        #print(importedSTLs[NDs[i]][4][0][0])
        dial(importedSTLs[NDs[i]][1][0]*3+400,-importedSTLs[NDs[i]][1]
[1]*3+400,20,sum(importedSTLs[NDs[i]][4][0])/10,True,mdisplay, f"ND{importedSTLs[CRs[i]][1]
[1],importedSTLs[CRs[i]][1][0]}",(0,0,255))


    light(400,100,(255,0,0), 10, len(neutrons) - 300 > neutronsetpoint, mdisplay, "High Activity")
    light(400,120,(0,0,255), 10, len(neutrons) + 300 < neutronsetpoint, mdisplay, "Low Activity")
    light(400,140,(0,255,0), 10, autocontrol == 1, mdisplay, "Autocontrol")
```



Unexpected firearm control “rod”