Open Source Simulation

Leveraging Open Source Technologies for Rapid Prototyping of Simulation Tools

Presented by:

Eduardo Fonseca

October 2016

Program Description

- Soyuz Docking to ISS Simulator
  - Built with Python and Blender
- Features
  - Open Source Tools
  - Rapid Prototyping

Business Problem

- Soyuz Docking is one of the most dynamic phases of flight in ISS operations
  - Training Flight Controllers is critical for Mission Success and Crew Safety
  - Human Space Flight involves high risk
- Currently Soyuz Docking is not a phase of flight that is trained in an integrated environment due to lack of adequate simulation tools

ISS005E19275

Business Problem

STINGER GHAFFARIAN TECHNOLOGIES

- International Space Station (ISS) EVA Suit Water Intrusion High Visibility Close Call IRIS Case Number: S–2013–199–00005 Recommendations
  - **R17:** The ISS Program should ensure that FMEA/CILs related to fastpaced failure scenarios (visiting vehicles, on-board emergency response, software transition issues, and serious system hardware failures) are regularly updated, studied, and used in training for flight controllers as well as engineering and safety personnel.
  - **R18:** As the success of the ISS Program continues, the ISS Program must institute requirements and behaviors that combat the tendency towards complacency by requiring regular training by all teams in the safety critical aspects of failures related to fastpaced scenarios (visiting vehicles, on-board emergency response, software transition issues, and serious system hardware failures).

Business Problem

- Dynamic phases of flight that are trained in integrated environment
  - Reboost/Debris Avoidance Maneuver
  - HTV/Dragon/Cygnus Rendezvous, Capture and Berthing
  - EVA
  - Major Failures
  - Emergencies
  - ~~*Soyuz/Progress Docking/Undocking*~~

Technology Solution

- A simulation tool to bridge the gap between current simulation capabilities/limitations

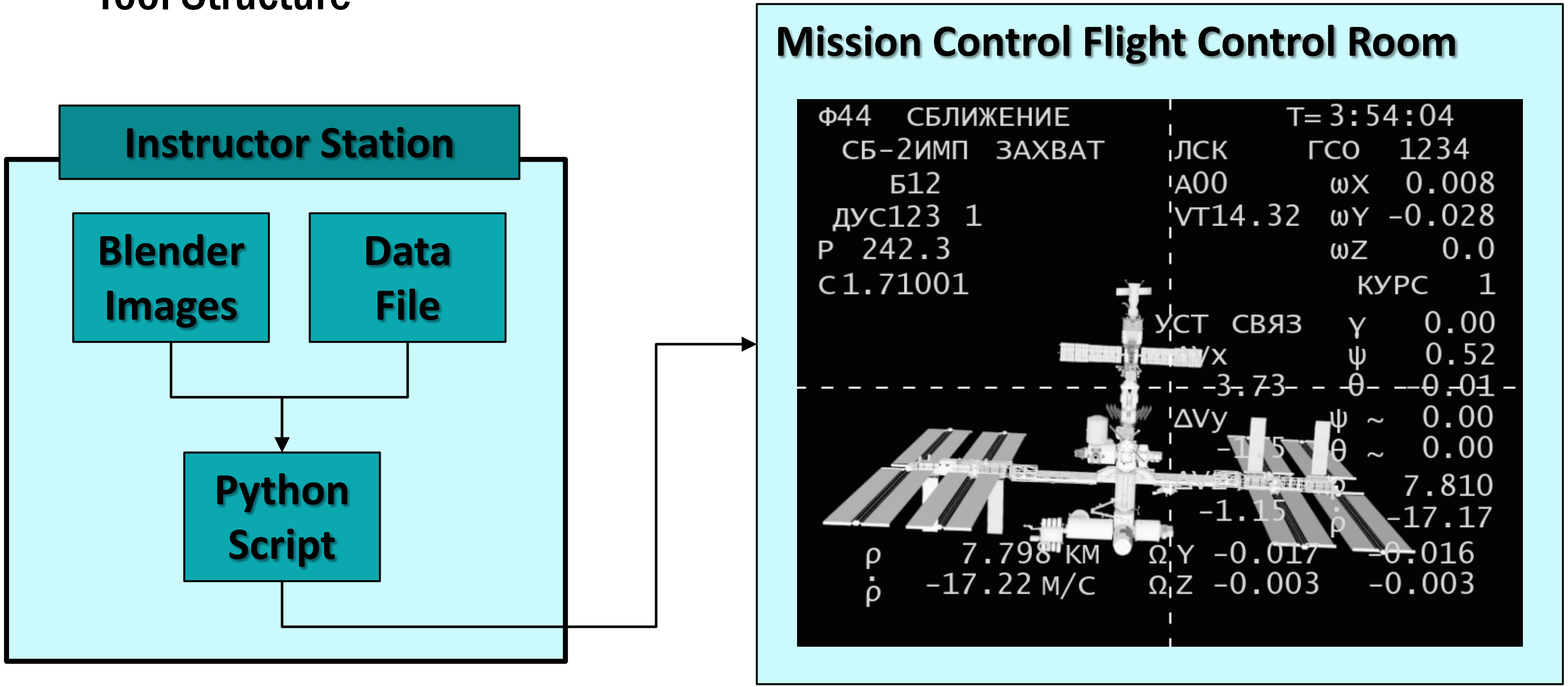
Technology Solution

Technology Solution

- Tools Used
  - Blender: 3D modeling
  - Excel: Data Manipulation
  - Python: Integration of data and 3D models

Technology Solution

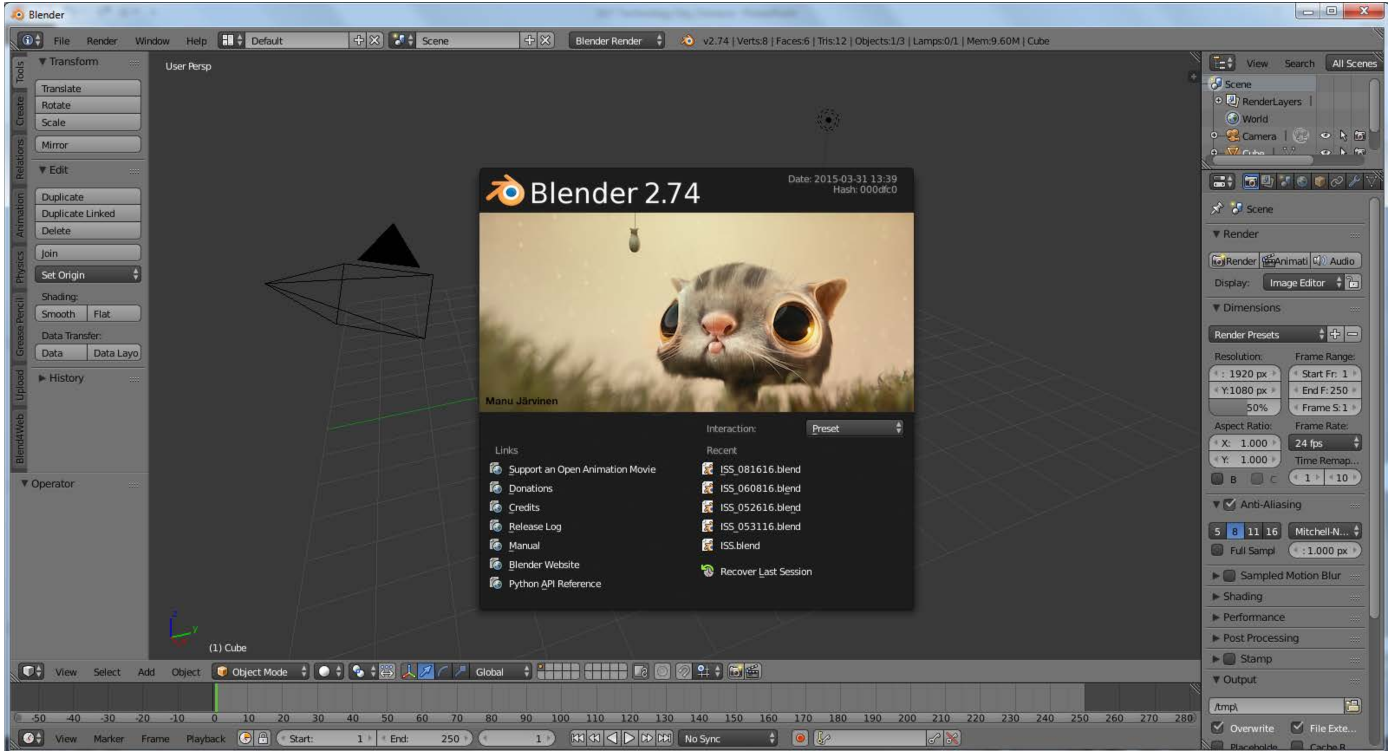
- Tool Structure



Technology Solution

STINGER GHAFFARIAN TECHNOLOGIES

- Blender



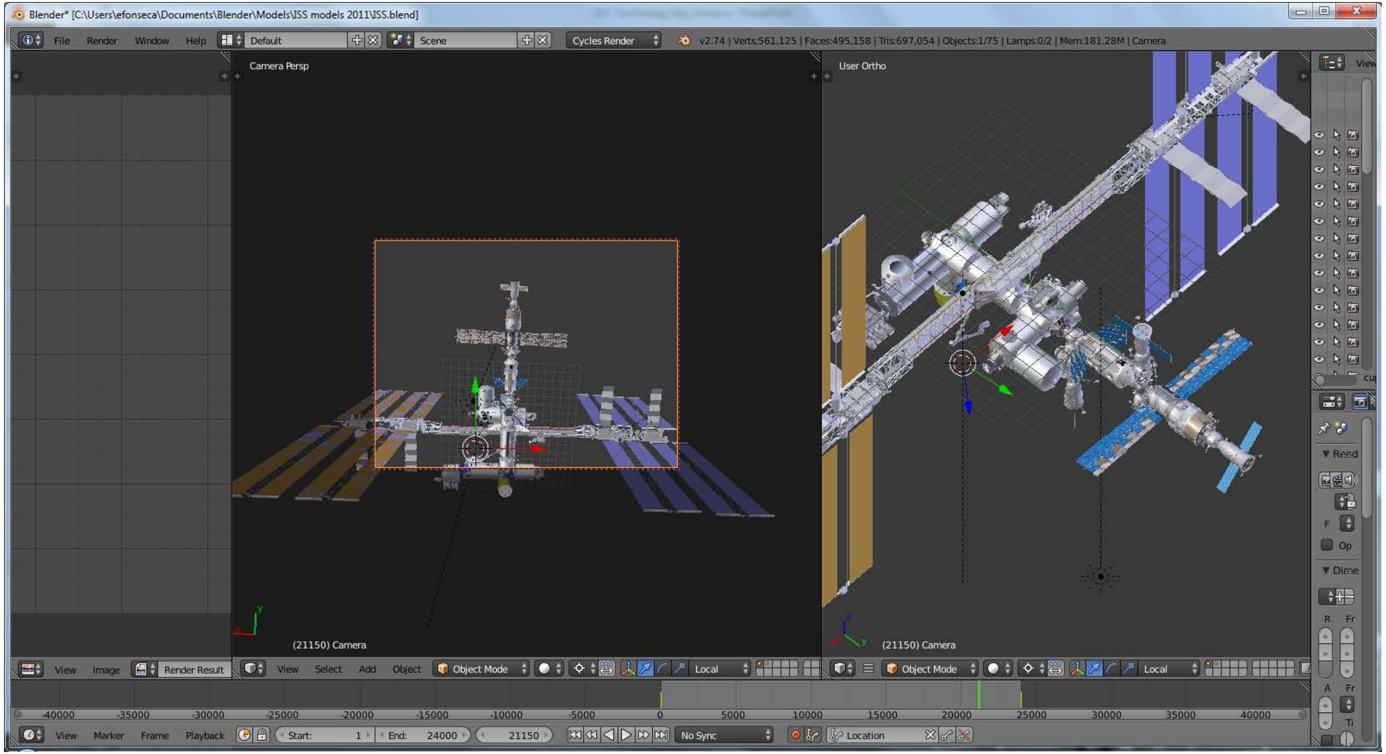
Technology Solution

- Blender
  - Models from NASA 3D Resources (public): https://nasa3d.arc.nasa.gov/
  - Instructors determine the camera views
  - Model / camera / lighting/ flight path created in Blender
  - Rendezvous profile scene rendered
    - Each image requires 30 seconds of render time
    - 40 minutes requires 2400 frames at 1 frame / second
    - Total rendering time approximately 20 hours per machine
  - A text file with location of images is generated by user

Technology Solution

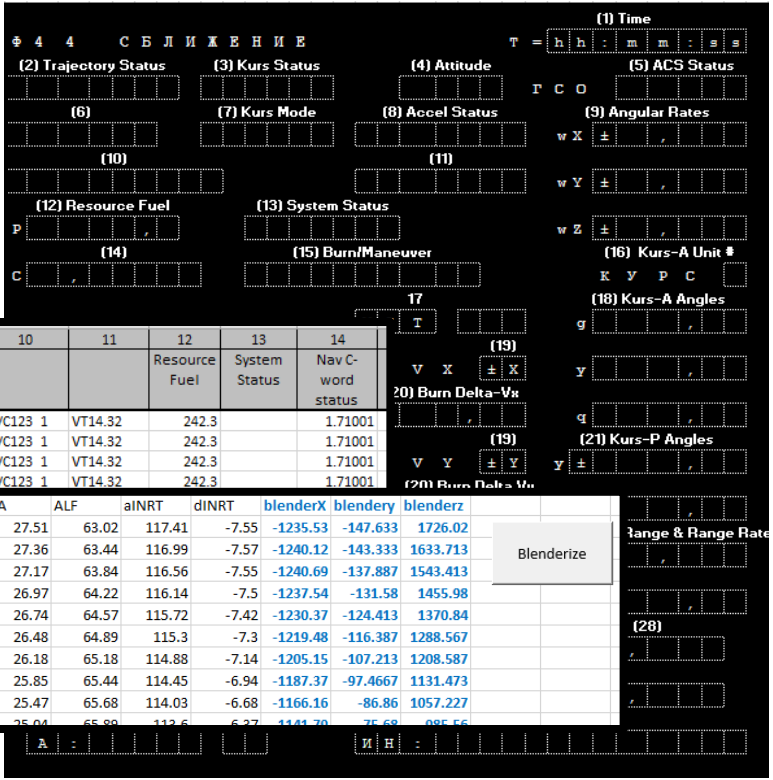
STINGER GHAFFARIAN TECHNOLOGIES

- Blender



Technology Solution

- Excel Data Manipulation
  - VVO provides data that drives the display
  - Data manipulated to be Blender and Python readable



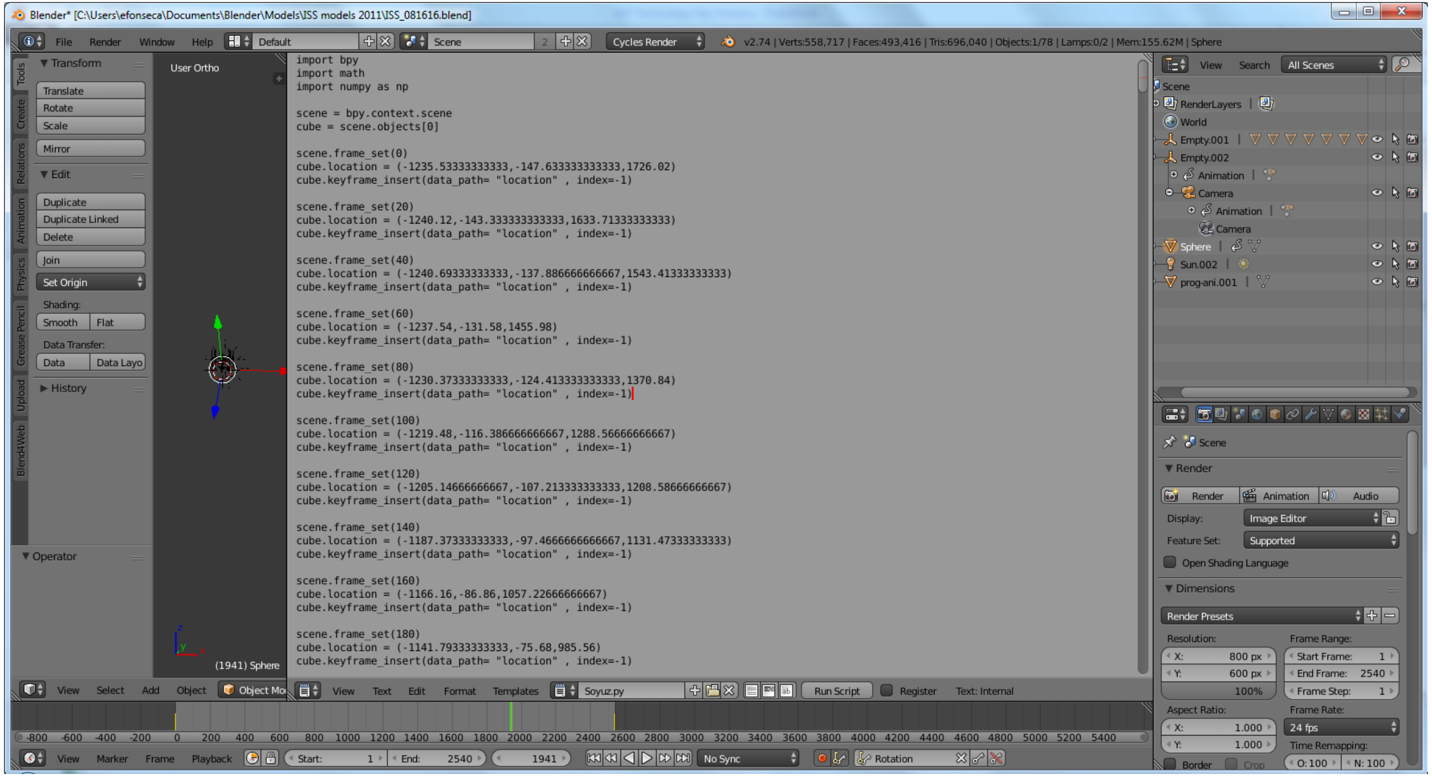
time (real)	time (SGMT)	1 Time	2 Trajectory Status	3 Kurs Status	4 Attitude	5 ACS Status	6 thruster manifold	7 Kurs Mode	8 Accel status	9a Angular Rates (X)	9b Angular Rates (Y)	9c Angular Rates (Z)	10	11	12 Resource Fuel	13 System Status	14 Nav C-word status
0:54:00	0:54:00	3:54:00	СБ-2ИМП	ЗАХВАТ	ЛСК	1234	Б12		A00	0.008	-0.017	-0.008	ДУС123 1	VT14.32	242.3		1.71001
0:54:01	0:54:01	3:54:01	СБ-2ИМП	ЗАХВАТ	ЛСК	1234	Б12		A00	0.008	-0.017	-0.008	ДУС123 1	VT14.32	242.3		1.71001
0:54:02	0:54:02	3:54:02	СБ-2ИМП	ЗАХВАТ	ЛСК	1234	Б12		A00	0.008	-0.017	-0.008	ДУС123 1	VT14.32	242.3		1.71001
0:54:03	0:54:03	3:54:03	СБ-2ИМП	ЗАХВАТ	ЛСК	1234	Б12		A00	0.008	-0.028	0.000	ДУС123 1	VT14.32	242.3		1.71001

TIME				Frame	DIST	VLPV	X ISS	Y ISS	Z ISS	SA	HLA	AZ	EL	MA	ALF	aINRT	dINRT	blenderX	blendery	blenderz
87	0:54:04	0:00:00	0	0	7.423	-12.66	-4.31	-0.515	6.021	111.13	-34.41	352.06	-54.36	27.51	63.02	117.41	-7.55	-1235.53	-147.633	1726.02
87	0:54:24	0:00:20	20	20	7.172	-12.465	-4.326	-0.5	5.699	110.71	-32.8	351.83	-52.76	27.36	63.44	116.99	-7.57	-1240.12	-143.333	1633.713
87	0:54:44	0:00:20	20	40	6.925	-12.291	-4.328	-0.481	5.384	110.29	-31.22	351.58	-51.18	27.17	63.84	116.56	-7.55	-1240.69	-137.887	1543.413
87	0:55:04	0:00:20	20	60	6.681	-12.138	-4.317	-0.459	5.079	109.87	-29.65	351.31	-49.62	26.97	64.22	116.14	-7.5	-1237.54	-131.58	1455.98
87	0:55:24	0:00:20	20	80	6.44	-12.003	-4.292	-0.434	4.782	109.45	-28.11	351.01	-48.1	26.74	64.57	115.72	-7.42	-1230.37	-124.413	1370.84
87	0:55:44	0:00:20	20	100	6.202	-11.885	-4.254	-0.406	4.495	109.03	-26.6	350.69	-46.59	26.48	64.89	115.3	-7.3	-1219.48	-116.387	1288.567
87	0:56:04	0:00:20	20	120	5.966	-11.782	-4.204	-0.374	4.216	108.61	-25.12	350.33	-45.12	26.18	65.18	114.88	-7.14	-1205.15	-107.213	1208.587
87	0:56:24	0:00:20	20	140	5.732	-11.694	-4.142	-0.34	3.947	108.19	-23.66	349.94	-43.68	25.85	65.44	114.45	-6.94	-1187.37	-97.4667	1131.473
87	0:56:44	0:00:20	20	160	5.499	-11.617	-4.068	-0.303	3.688	107.77	-22.24	349.51	-42.27	25.47	65.68	114.03	-6.68	-1166.16	-86.86	1057.227

Blenderize

Technology Solution

STINGER GHAFFARIAN TECHNOLOGIES



Technology Solution

- Python
  - Reads text file with location of Blender images
  - Reads text file with VVO data
  - Loop:
    - Displays specified image from table
    - Displays specified telemetry from table
    - Iterate

Technology Solution

- Python
  - Libraries:
    - NumPy
    - Pandas
    - Matplotlib

Technology Solution

- Python

```python
# For loop that runs for the duration equal to the size of the array.
# For 40 minute video it should be about 2400 loops
while True:
    img = imread(cbook.get_sample_data(frame.FILE[i+j])) # Reads the respective image
    plt.clf() # Clears the screen
    plt.imshow(img, cmap = cm.Greys_r, zorder=0, extent=[0.5, 8.0, 1.0, 7.0]) # Show the image
    plt.xlim(0.5,8) # Set the x-limits on the display
    plt.ylim(1,7) # Set the y-limits on the display
    plt.plot(xhudx,yhudx,"--", color="#ffffff", dashes=(11, 14), linewidth=2) # Plot the horizontal guide
    plt.plot(xhudy,yhudy,"--", color="#ffffff", dashes=(11, 14), linewidth=2) # Plot the vertical guide
    plt.text(0.75,6.70,u'Φ' + '44', fontdict=font) # Add text
    plt.text(1.66,6.70,u'СБЛИЖЕНИЕ', fontdict=font) # Add text
    plt.text(5.55,6.70,'T=', fontdict=font)
    plt.text(6.00,6.70,str(rndz.a1[i]), fontdict=font) # Add text
    plt.text(2.28,6.34,str(rndz.a2[i]), fontdict=font, horizontalalignment='right') # Add text
    plt.text(2.58,6.34,str(rndz.a3[i]), fontdict=font) # Add text
    plt.text(4.40,6.34,str(rndz.a4[i]), fontdict=font) # Add text
    plt.text(5.78,6.34,u'ГСО', fontdict=font) # Add text
    plt.text(6.72,6.34,str(rndz.a5[i]), fontdict=font) # Add text
    plt.text(2.00,5.97,str(rndz.a6[i]), fontdict=font, horizontalalignment='right') # Add text
    if str(rndz.a7[i]) != '.':
        plt.text(3.65,5.97,str(rndz.a7[i]), fontdict=font, horizontalalignment='right') # Add text
    plt.text(4.40,5.97,str(rndz.a8[i]), fontdict=font, horizontalalignment='left') # Add text
    plt.text(6.00, 5.97,u'ω', fontdict=font) # Add text
    plt.text(6.00, 5.61,u'ω', fontdict=font) # Add text
    plt.text(6.00, 5.25,u'ω', fontdict=font) # Add text
    plt.text(6.22, 5.97,'X', fontdict=font) # Add text
    plt.text(6.22, 5.61,'Y', fontdict=font) # Add text
    plt.text(6.22, 5.25,'Z', fontdict=font) # Add text
    plt.text(7.69,5.97,str("%.3f" % rndz.a9a[i]), fontdict=font, horizontalalignment='right') # Add text
    plt.text(7.69,5.61,str("%.3f" % rndz.a9b[i]), fontdict=font, horizontalalignment='right') # Add text
    plt.text(7.69,5.25,str(rndz.a9c[i]), fontdict=font, horizontalalignment='right') # Add text
    plt.text(2.00,5.61,str(rndz.a10[i]), fontdict=font, horizontalalignment='right') # Add text
    plt.text(2.42,5.61,'1', fontdict=font, horizontalalignment='right') # Add text
    plt.text(4.40,5.61,str(rndz.a11[i]), fontdict=font, horizontalalignment='left') # Add text
    plt.text(0.75,5.25,'P', fontdict=font, horizontalalignment='left') # Add text
    plt.text(1.21,5.25,str(rndz.a12[i]), fontdict=font, horizontalalignment='left') # Add text
```

Technology Solution

- Logistics
  - Graphics and telemetry cannot be generated in real time (currently)
  - Number of rendering workstations reduce rendering time
- Cost
  - Open Source (no cost)
  - 10 hours of prototype development

Future Roadmap

STINGER GHAFFARIAN TECHNOLOGIES

- Allow the capability to render 3D model and data in real time
- Investigate integration to existing simulators (Space Station Training Facility / TS21 Simulator)
- Train users on tool operations and improvements
- Apply to other simulators / programs / customers
  - Axiom
  - Boeing
  - Sierra Nevada
  - SpaceX
  - ...
- This is one of many possible solutions to solve the problem

Contact Information

Eduardo Fonseca

Email: **efonseca@sgt-inc.com**

Phone: 281-244-5507 (O) | 281-467-2219 (M)