Lee James Nestor

Computer Engineer, Mathematician

Design Portfolio

Figure 1: NeRF Rendering of HVAC System

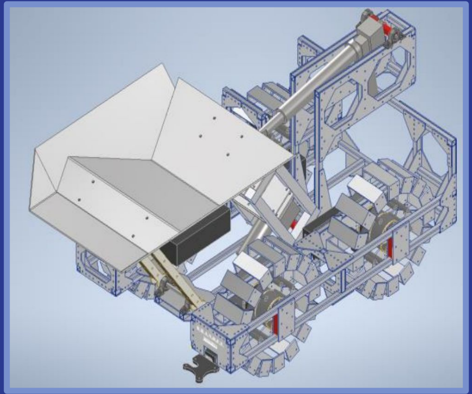

Figure 2: CAD Model of a Mining Robot

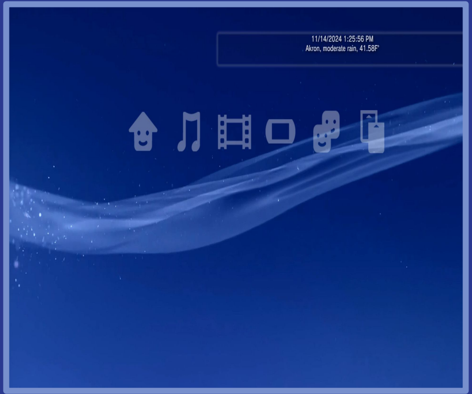

Figure 3: Recreation of XMB Home Screen

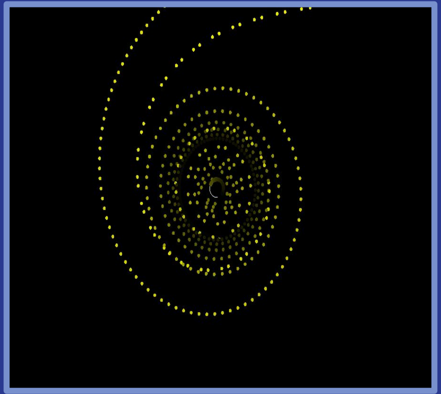

Figure 4: Particle Simulation of bodies about a black hole

About me

I’m an engineer and mathematician whose passions range from Computer Architectures to Relativistic Physics. I have experience with embedded systems such as Microcontrollers, Single Board Computers, and FPGAs. I have worked with a wide plethora of programming languages such as C, C++, Python, Matlab, and more. I have also done work with mathematical modeling, numerical system simulation and analysis.

Engineering Work and Designs

Autonomous Photogrammetric Data Collection and Analyzation

- **Problem**
  - Structural Damage or Environmental Hazard making the area unsafe for human investigation
- **Solution**
  - Navigation using conventional Simultaneous Localization and Mapping Techniques
  - Photogrammetric Data Processing with AI, using Neural Radiance Fields or (NeRF) combined with COLMAP (Structure-from-Motion and Multi-View Stereo REconstruction)
  - Analyzation of the generated model using simple Finite Element Analysis (FEA) for Structural Integrity
- **Source**
  - https://github.com/colmap/colmap
  - https://github.com/NVlabs/instant-ngp

Figure 5: NeRF Rendering Text on a Coke-a-Cola Can

Figure 6: NeRF Rendering reflective surface on a Coke-a-Cola Can

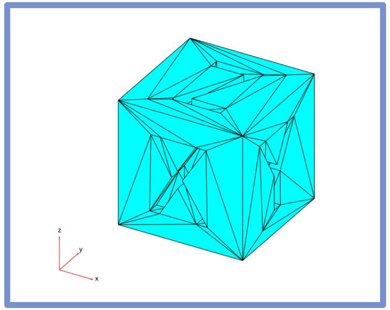

Figure 7: Matlab display of an STL Model

Figure 8: NeRF rendering of an HVAC System

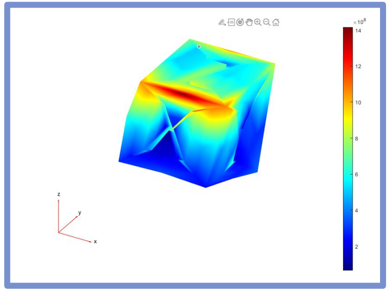

Figure 9: FEA of a cube with pressure applied

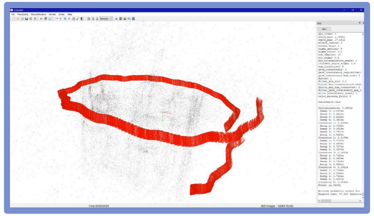

Figure 10: COLMap Program with generated camera positions and angles

Autonomous Damage and Structure Scanning Drone

- **Problem**
  - Structural Damage or Environmental Hazard making the area unsafe for human investigation
- **Solution**
  - Development and design of a quadcopter drone
  - Navigation using an embedded system featuring a Raspberry Pi 3b+ SBC and a Mateksys F405-VTOL working together with Ardupilot firmware
  - Data collection with an on-board camera and installed LiDAR for positional sensing
  - Implemented network for data transmission back to a base computer for processing and manual drone control
  - Base program for receiving this data and presenting it to the user
- **Source**
  - https://github.com/ArduPilot/ardupilot

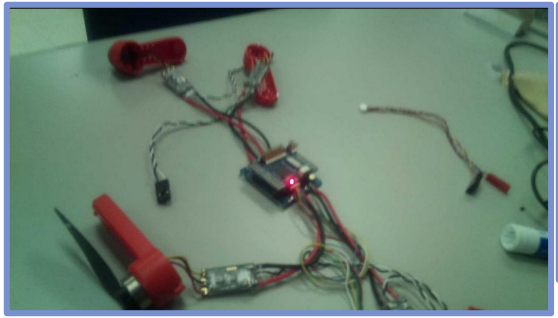

Figure 11: Picture of the flight controller and connected motors taken with the on board camera

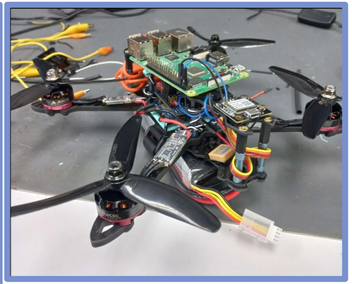

Figure 12: Complete prototype drone

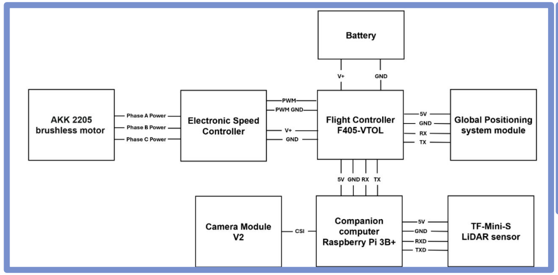

Figure 13: Block Diagram of the Hardware structure

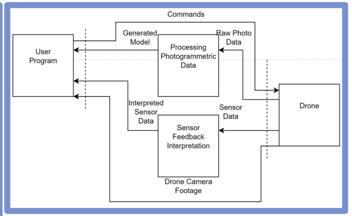

Figure 14: Block diagram the data processing system

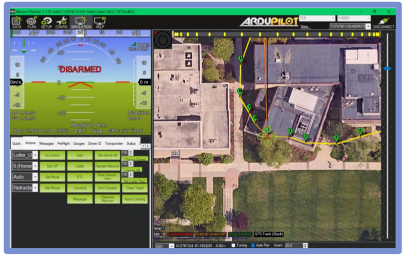

Figure 15: Ardupilot autopilot route for around an University building

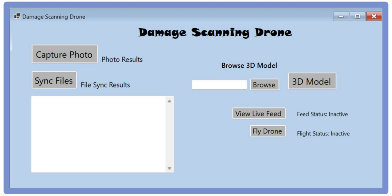

Figure 16: User Interface

The University of Akron NASA Robotics Team Software

- **Problem**
  - To construct a robot to autonomously navigate an arena simulating the surface of the moon and proceed to dig soil and deposit to construct a berm
- **Solution**
  - Development of a wireless and robust communications network for controlling the robot
  - Development of an embedded system, here a Jetson Nano, to interface with the wireless controller and the electrical components of the robot
  - Analysis of the design requirements to properly create the software to work within
  - Research and development into autonomous systems and mapping techniques using Robotic Operating System or ROS
- **Source**
  - https://github.com/UA-NASA-Robotics
  - https://github.com/ros/ros

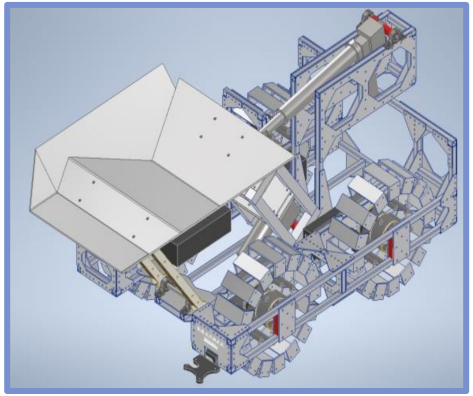

Figure 17: CAD Rendering of UA NASA Robotics Mining Team 2023-2024 Robot, Software written by Software Team

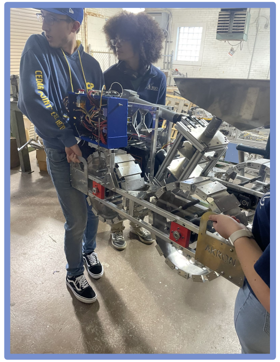

Figure 18: Team members carrying the robot for testing

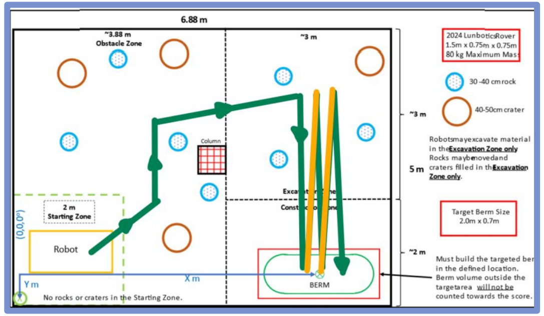

Figure 19: Planned routing for navigating the arena

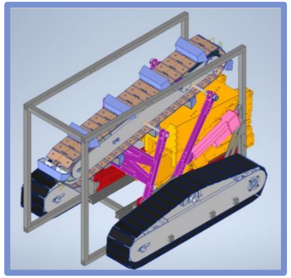

Figure 20: CAD Rendering of UA NASA Robotics Mining Team 2022-2023 Robot, software written by Lee Nestor

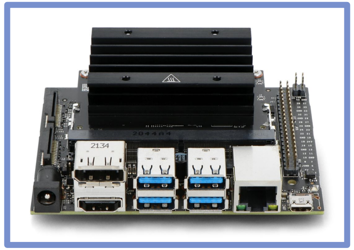

Figure 21: Jetson Nano SBC

ROS

Figure 22: Robotic Operating System used for governing autonomy

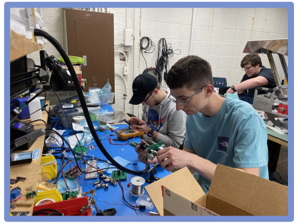

Figure 23: Electrical/Software Members Soldering

Figure 24: Software Member working on SBC

Figure 25: Software Team photo

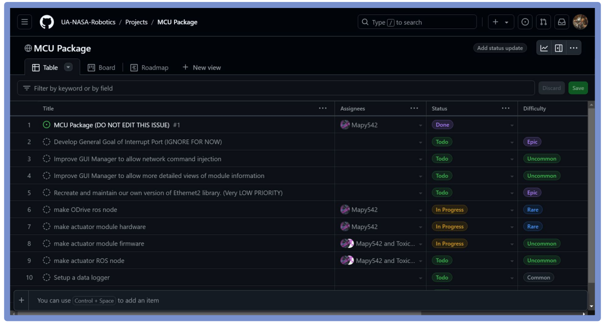

Figure 26: Github project organization

Figure 27: Team Members transporting the Robot

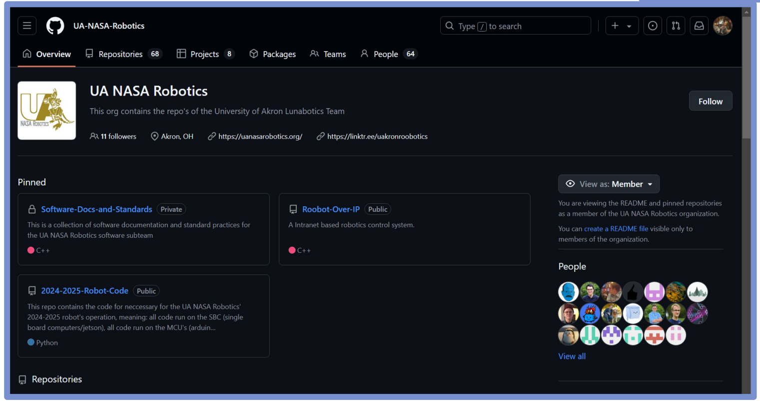

Figure 28: University of Akron NASA Robotics Mining Team Github PAge

The University of Akron NASA Robotics Team Software Head Lead

- **Problem**
  - Managing and coordination of a team of 20+ people while also engaging them and properly utilizing their time
- **Solution**
  - Regular stand-up meetings for coordination and re-alignment of goals and development processes
  - Team meetings for group organization and presentation of current status
  - Development of extensive documentation and employment of software engineering techniques
  - Using github and setting up a system for team members to be kept aware of the current projects and their potential difficulty
- **Source**
  - https://github.com/UA-NASA-Robotics
  - https://github.com/orgs/UA-NASA-Robotics/projects?query=is%3Aopen

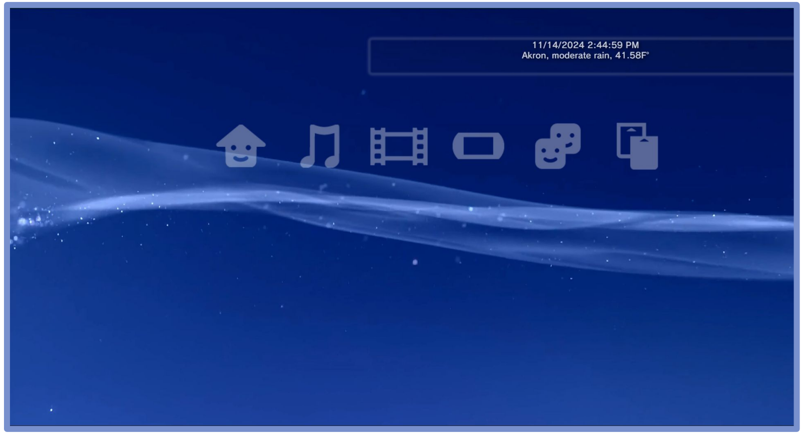

Figure 29: Screenshot of the startpage, written in HTML, CSS, and JS

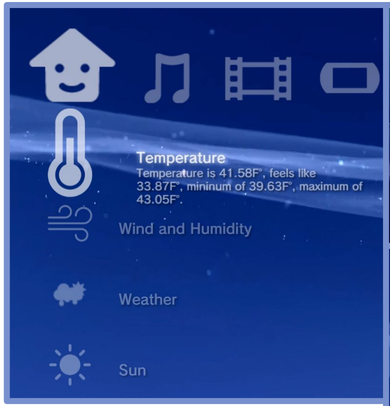

Figure 30: Weather data pulled from the internet

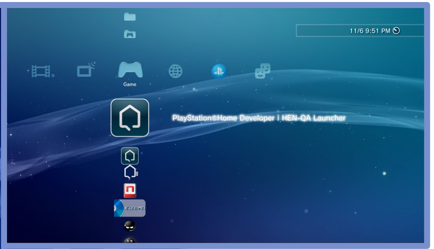

Figure 31: Original home screen the design is based off of

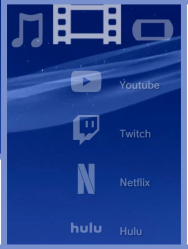

Figure 32: Icon hovered over with mouse cursor presents clickable links

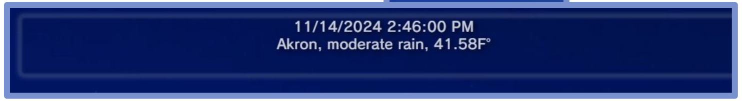

Figure 33: Time, Data, and Weather data presented

Custom Playstation 3 Inspired Webpage

- **Problem**
  - There is a need for a homepage for web browsing that has quick access for a wide variety of links while also presenting relevant information
- **Solution**
  - Utilizing HTML, CSS and JS to design an implement a web page that can incorporate all of these requirements
  - Use the Playstation 3 home screen as inspiration for a way to organize and present different links
  - Allow the icons to expand with user input such as a mouse hover
- **Source**
  - https://github.com/LeeJamesNestorAkron/Playstation3StartPage

Schwarzschild Metric Based Black Hole Simulation

- **Problem**
  - A lack of understanding of general relativistic effects around a black hole, or an object whose Schwarzschild Radius is outside its physical body
- **Solution**
  - Develop and design a numerical simulation to solve and display a particle affected by a black hole
  - Derive and understand Schwarzschild geodesics in order to create a set of differential equations to solve numerically
  - Implement an 4th order Runge-Kutta numerical solver to evaluate the system of equations
  - Display and render the particles using OpenGL rendering
- **Source**
  - https://github.com/LeeJamesNestorAkron/schwarzchildBlackhole
  - https://en.wikipedia.org/wiki/Schwarzschild_geodesics

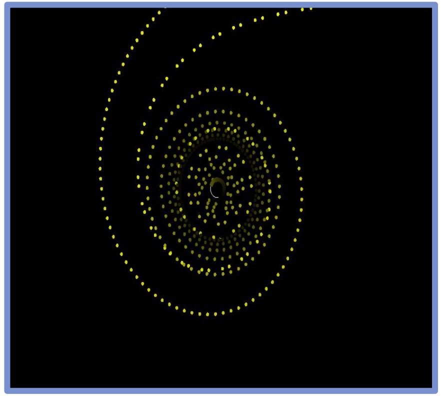

Figure 34: Example simulation of large mass particles orbiting a black hole

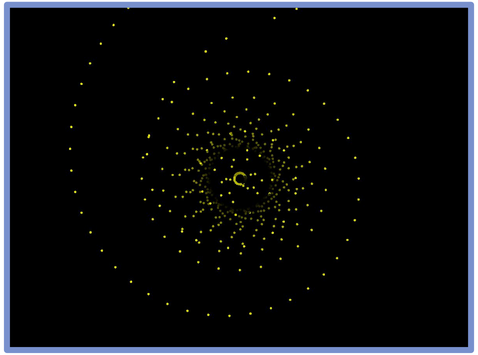

Figure 35: Second example of large mass particles orbiting a black hole

```python
def runge_kutta_4th_order(f, tau, IC, h_func, M):
    """
    Implements 4th-order Runge-Kutta method to solve differential equations.

    Preconditions:
    - f: Function that returns the derivatives
    - tau: Time array (non-negative)
    - IC: Initial conditions (array containing [t, phi, r, L, E])
    - h_func: Function to calculate time step
    - M: Mass of the central object

    Postconditions:
    - Returns array of integrated values over time
    """
    xSize = len(IC)
    tphirEL = np.zeros((len(tau), xSize))
    tphirEL[0] = IC
    rPrev = []


    dr_dtau_sqr_initial = IC[3]**2 - (1 - (SR / IC[2])) * (1 + (IC[4]**2 / IC[2]**2))
    if dr_dtau_sqr_initial > 0:
        rPrev.append(0)
        rPrev.append(1)
    else:
        rPrev.append(1)
        rPrev.append(0)

    for i in range(1, len(tau)):
        h = h_func(tau[i - 1])
        k1 = h * f(tau[i - 1], tphirEL[i - 1], M,rPrev)
        k2 = h * f(tau[i - 1] + h/2, tphirEL[i - 1] + k1/2, M,rPrev)
        k3 = h * f(tau[i - 1] + h/2, tphirEL[i - 1] + k2/2, M,rPrev)
        k4 = h * f(tau[i - 1] + h, tphirEL[i - 1] + k3, M,rPrev)
        tphirEL[i] = tphirEL[i - 1] + (1/6) * (k1 + 2*k2 + 2*k3 + k4)
        rPrev.append(tphirEL[i-1,2])

    return tphirEL
```

Figure 36: Python Implementation of a Runge-Kutta 4th Solver

```python
def schwarzschildGeodesicEq(tau, tphirEL, SR,rPrev):
    """
    Schwarzschild geodesic equations to determine the time, angular, and radial motion of the object.

    Preconditions:
    - tau: Proper time (positive)
    - tphirEL: Array containing [t, phi, r, E, L] (all non-negative)
    - SR: Schwarzschild radius (positive)

    Postconditions:
    - Returns an array [dt_dtau, dphi_dtau, dr_dtau, E, L] containing time, angular, and radial derivatives.
    """
    t, phi, r, E, L = tphirEL
    epsilon = 1e-8  # Small tolerance for floating-point comparisons
    # Time, angular, and radial motion derivatives
    dt_dtau = E / (1 - (SR / r))
    dphi_dtau = L / (r**2)
    # Effective potential and radial motion
    Veff = (1 - (SR / r)) * (1 + (L**2 / r**2))
    dr_dtau_sqr = E**2 - Veff
    dL_dtau = -Gamma_L * L
    dE_dtau = -Gamma_E * E
    # Ensure radial derivative is real and non-negative
    dr_dtau = np.sqrt(np.maximum(dr_dtau_sqr, 0))
    if E**2 > Veff + epsilon:
        dr_dtau = -abs(dr_dtau)  # Ensure proper inward motion
    elif E**2 <= Veff:
        dr_dtau = abs(dr_dtau)  # Ensure proper outward motion
    if dr_dtau_sqr < 0:
        dr_dtau_sqr =0
    # If inside the Schwarzschild radius, motion should stop
    if r < (SR + epsilon):
        dt_dtau = dphi_dtau = dr_dtau = 0.000001
        t = r = phi = 0

    return np.array([dt_dtau, dphi_dtau, dr_dtau, dL_dtau, dE_dtau])
```

Figure 37: Python Implementation of Schwarzschild geodesic equations

Contact

Lee James Nestor
leenestor.akron@gmail.com
330-696-6327
www.LeeJamesNestor.com

Figure 38: Jean-Pierre Luminet's Simulation of a Black Hole 1978