Career Paths: Math Outside of Academia

Bloomberg

Engineering

ICERM's Geometry Labs United Conference
July 17, 2020

Ryan Hoban
Senior Engineer

TechAtBloomberg.com

About Me

- Graduate of University of Maryland (Student of Goldman, 2009)
- Maryland Experimental Geometry Lab

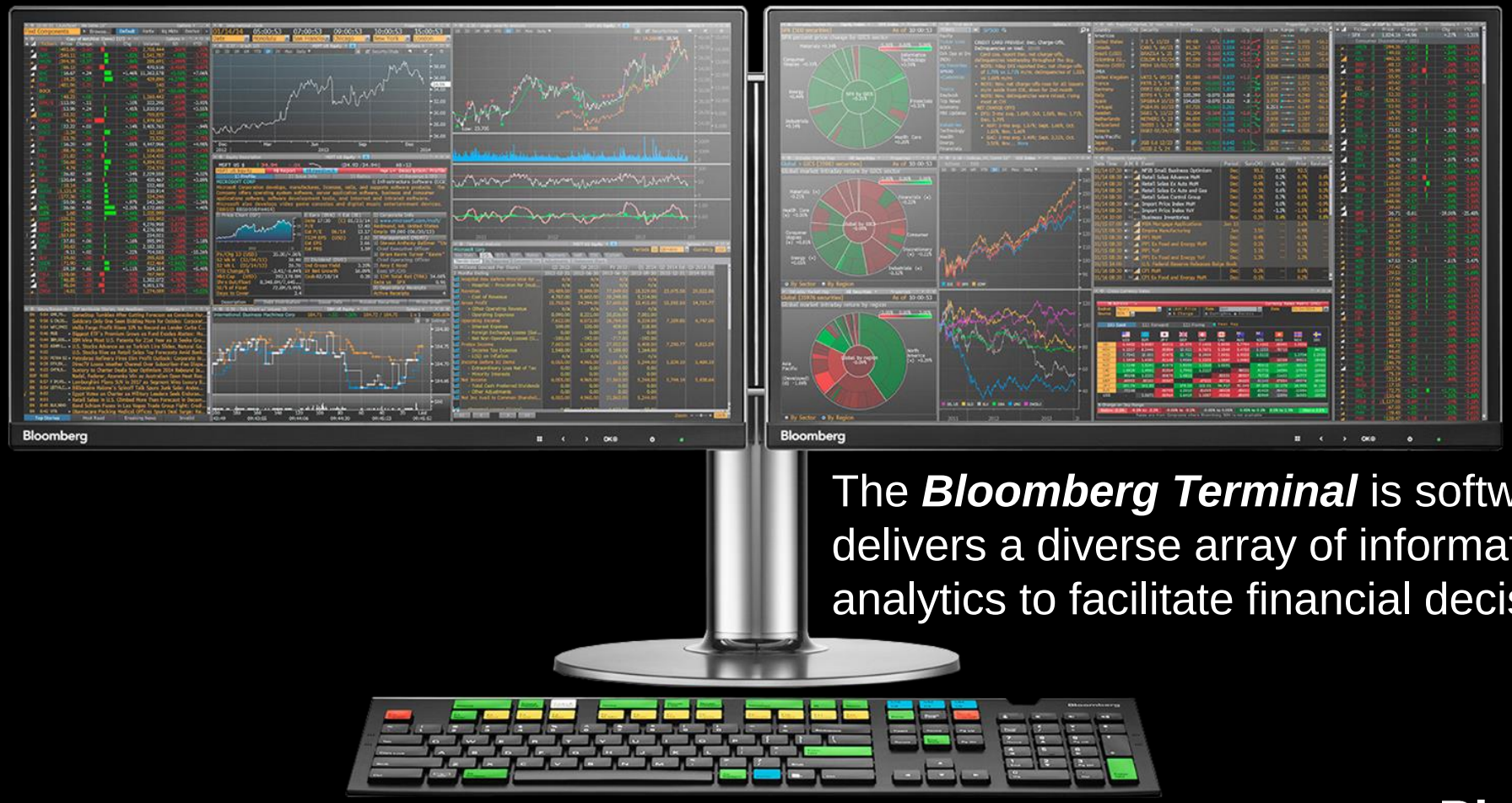
- Senior Engineer at Bloomberg
- Bloomberg's Automated Testing Guild Leadership

TechAtBloomberg.com

Bloomberg

Engineering

About Bloomberg



The ***Bloomberg Terminal*** is software that delivers a diverse array of information, news and analytics to facilitate financial decision-making.

TechAtBloomberg.com

Bloomberg

Engineering

Bloomberg Engineering by the numbers

- **6,000+** software engineers
- **150+** technologists and data scientists devoted to machine learning
- **Degrees:** Bachelors, Masters, Ph.D.
- **Majors:** Computer Science, Engineering, Physics, Chemistry, Mathematics

TechAtBloomberg.com

Bloomberg

Engineering

Constant Learning

- New Hire training is extremely extensive, multiple tracks
  — Many continuing education classes on both tech and finance

- In-house lectures and guest speakers every week
  — Opportunity to share your own work

- Developed **Integration Testing with Python**, now part of the New Hire program
  — Teach several classes on **Test Driven Development (TDD)**

Where do we use real math?

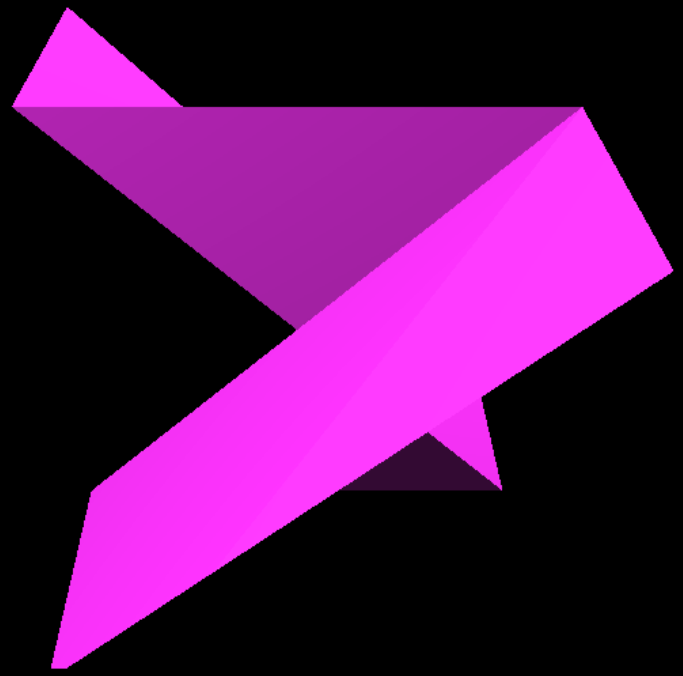
- Current team **Fixed Income Real-time Pricing**
  — Bond math - price/yield/spread calculations
- Present value calculations
  — Curve calculations, Kalman filtering
- Data Science: **Foundations of Machine Learning**
  https://TechAtBloomberg.com/FOML
- **Software testing:** Software metrics, code analysis

What is software testing?

- Systematic process to help ensure software behaves as desired
  - Want to **prove** that our software behaves the way we want
  - We want to **maximize** how much code is tested
  - We want to **minimize** the amount of effort to do it
  - We **quantify** the extent to which our software is tested

- **Automated Testing Guild**: Help every engineer to adopt automated testing in their daily workflow at Bloomberg
  - Training, tooling, measurements

Testing Example: A Ray Traced Crooked Plane



TechAtBloomberg.com

Bloomberg

Engineering

Intersect Ray with Crooked Plane

```python
def intersect(ray):
    '''Find the intersections of ray with the standard crooked plane'''
    intersections = []

    # Find intersection with stem
    if ray.direction.x != 0:  # Not parallel to stem
        t = -ray.origin.x / ray.direction.x
        intersection_point = ray.origin + t * ray.direction

        if minkowski_norm(intersection_point) < 0:
            intersections.append(t)

    # Find intersection with wing1
    if ray.direction.y != - ray.direction.z:  # Not parallel to wing1
        t = - (ray.origin.y + ray.origin.z)/(ray.direction.y + ray.direction.z)
        intersection_point = ray.origin + t * ray.direction

        if intersection_point.x >= 0:
            intersections.append(t)

    # Find intersection with wing2
    if ray.direction.y != ray.direction.z:  # Not parallel to wing2
        t = - (ray.origin.y - ray.origin.z)/(ray.direction.y - ray.direction.z)
        intersection_point = ray.origin + t * ray.direction

        if intersection_point.x <= 0:
            intersections.append(t)

    return intersections
```

Bloomberg

Engineering

How to test this code

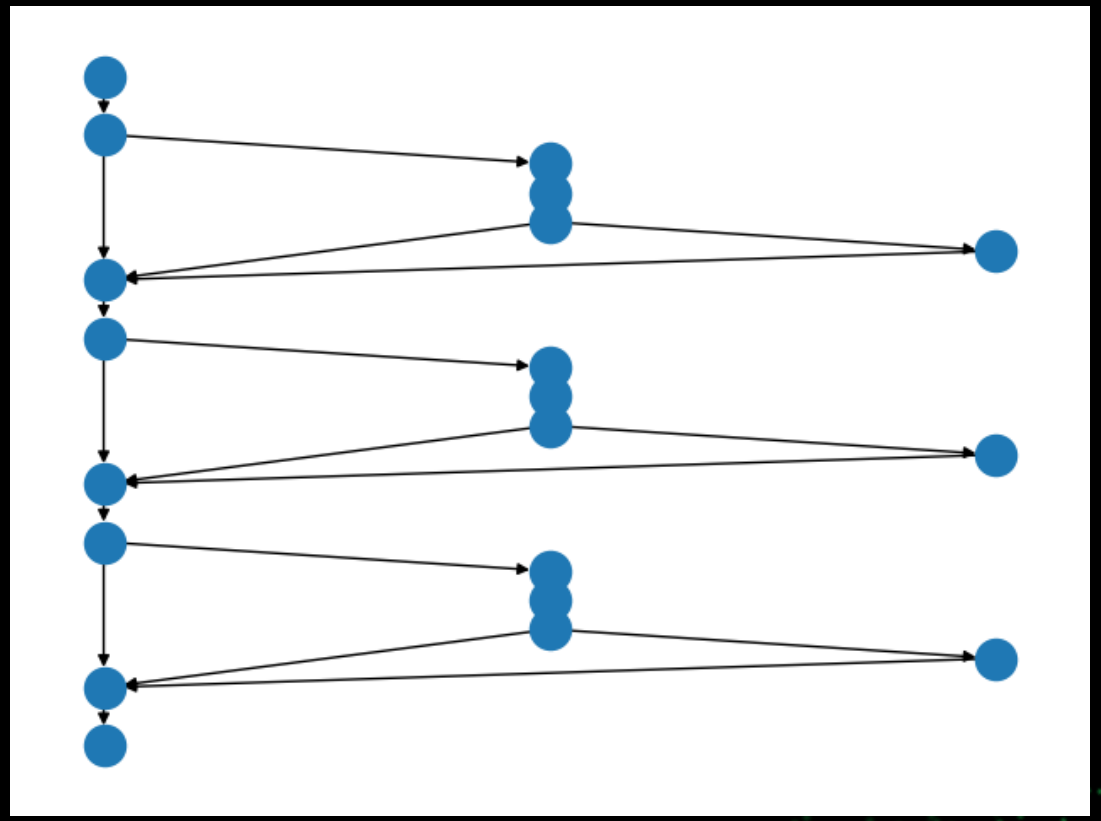
```python
def intersect(ray):
    '''Find the intersections of ray with the standard crooked plane'''
    intersections = []

    # Find intersection with stem
    if ray.direction.x != 0:  # Not parallel to stem
        t = -ray.origin.x / ray.direction.x
        intersection_point = ray.origin + t * ray.direction

        if minkowski_norm(intersection_point) < 0:
            intersections.append(t)

    # Find intersection with wing1
    if ray.direction.y != - ray.direction.z:  # Not parallel to wing1
        t = - (ray.origin.y + ray.origin.z)/(ray.direction.y + ray.direction.z)
        intersection_point = ray.origin + t * ray.direction

        if intersection_point.x >= 0:
            intersections.append(t)

    # Find intersection with wing2
    if ray.direction.y != ray.direction.z:  # Not parallel to wing2
        t = - (ray.origin.y - ray.origin.z)/(ray.direction.y - ray.direction.z)
        intersection_point = ray.origin + t * ray.direction

        if intersection_point.x <= 0:
            intersections.append(t)

    return intersections
```

- Input various rays and check it gives the right output
- How many tests are needed?

```python
def test_ray_parallel_to_stem():
    ray = Ray(Point(-1, 0, 1), Vector(0, 1, 1))
    assert [] == intersect(ray)


def test_ray_intersects_stem():
    ray = Ray(Point(0, 0, -1), Vector(1, 0, 0))
    assert [0] == intersect(ray)


def test_ray_misses_stem():
    ray = Ray(Point(-1, 1, 0), Vector(1, 0, 0))
    assert [] == intersect(ray)
```

Control Flow Graph

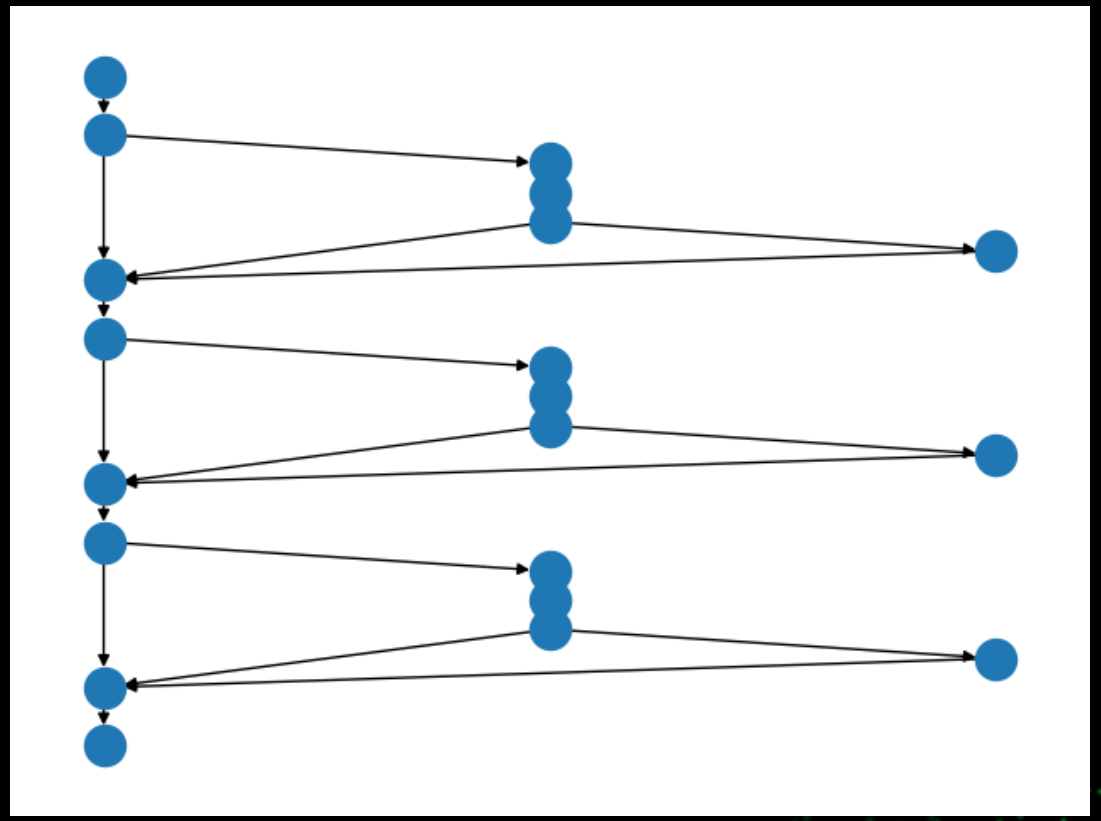
```python
def intersect(ray):
    '''Find the intersections of ray with the standard crooked plane'''
    intersections = []

    # Find intersection with stem
    if ray.direction.x != 0:  # Not parallel to stem
        t = -ray.origin.x / ray.direction.x
        intersection_point = ray.origin + t * ray.direction

        if minkowski_norm(intersection_point) < 0:
            intersections.append(t)

    # Find intersection with wing1
    if ray.direction.y != - ray.direction.z:  # Not parallel to wing1
        t = - (ray.origin.y + ray.origin.z)/(ray.direction.y + ray.direction.z)
        intersection_point = ray.origin + t * ray.direction

        if intersection_point.x >= 0:
            intersections.append(t)

    # Find intersection with wing2
    if ray.direction.y != ray.direction.z:  # Not parallel to wing2
        t = - (ray.origin.y - ray.origin.z)/(ray.direction.y - ray.direction.z)
        intersection_point = ray.origin + t * ray.direction

        if intersection_point.x <= 0:
            intersections.append(t)

    return intersections
```



Control Flow Graph

- **Cyclomatic Complexity:** Number of linearly independent paths through the CFG
  — Gives a lower bound on the number of tests needed to cover all branches
  — Needs only 1 test for 100% statement coverage (exists a Hamiltonian path)
  — Needs at least 7 tests for 100% branch coverage
  — (Cyclomatic Complexity is 7)
  — Needs at least 27 tests for 100% path coverage

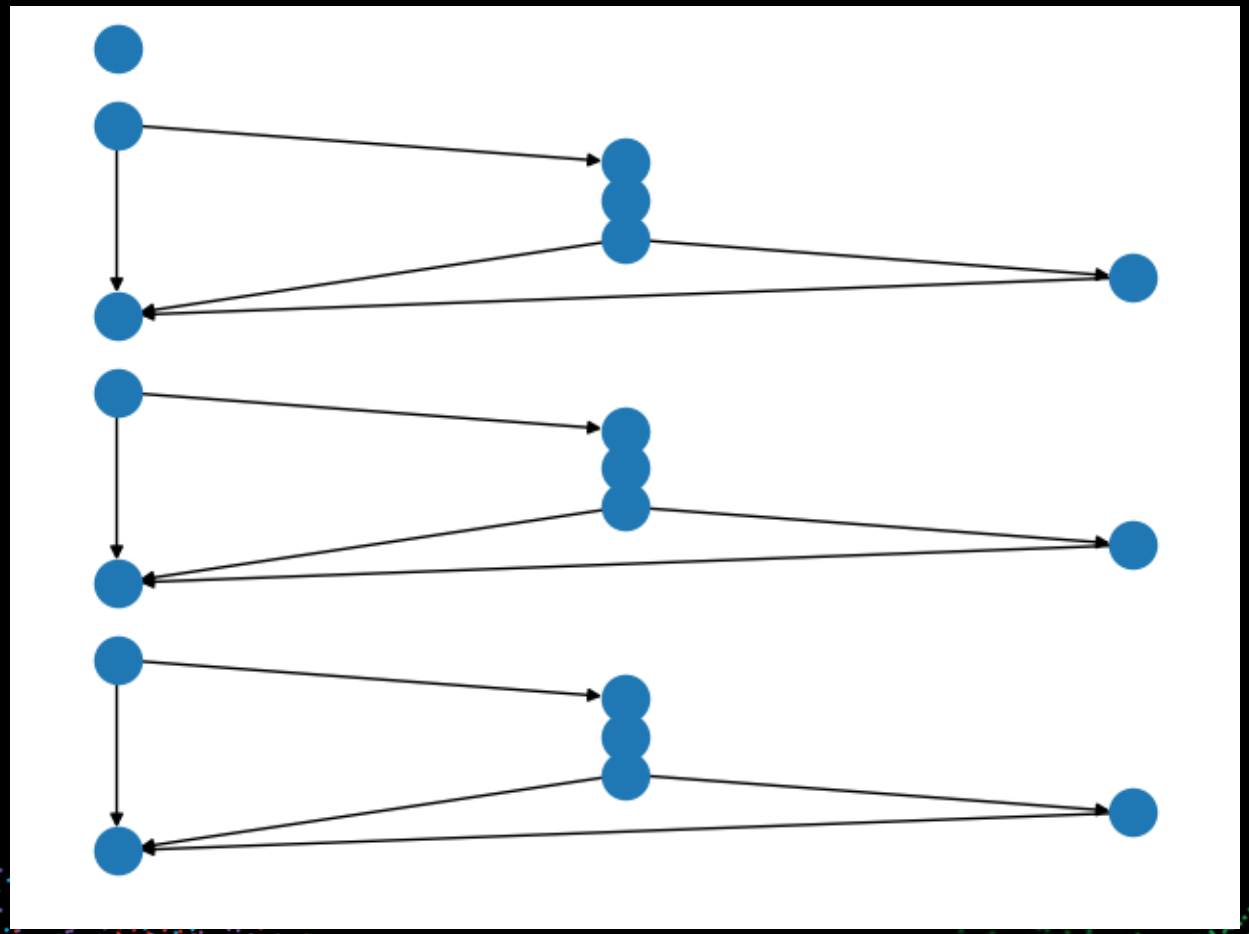
- CFG makes clear ways to improve the code



TechAtBloomberg.com

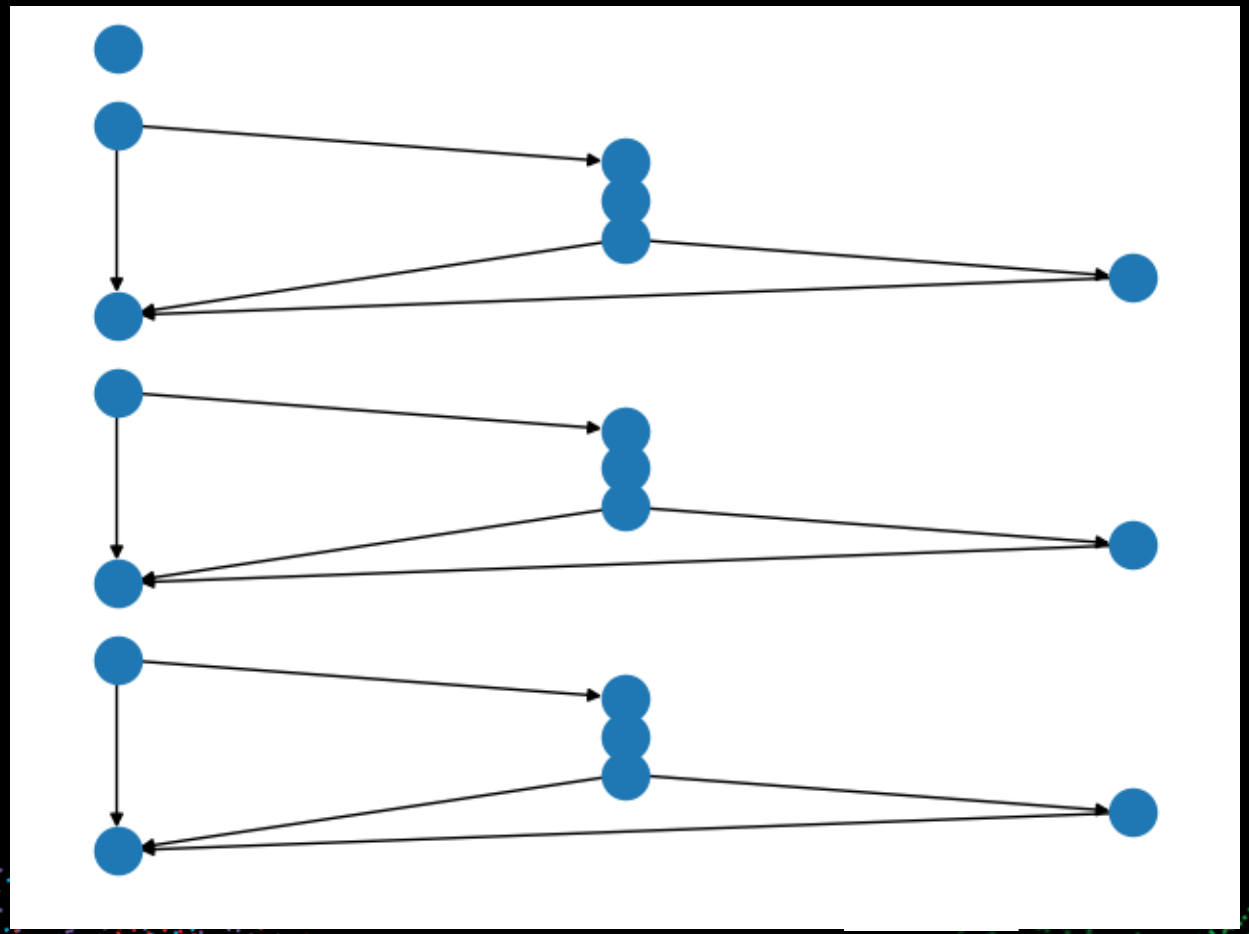
Refactoring opportunities

```python
def intersect(ray):
    '''Find the intersections of ray with the standard crooked plane'''
    return intersect_stem(ray) + intersect_wing1(ray) + intersect_wing2(ray)

def intersect_stem(ray):
    intersections = []
    if ray.direction.x != 0:  # Not parallel to stem
        t = -ray.origin.x / ray.direction.x
        intersection_point = ray.origin + t * ray.direction

        if minkowski_norm(intersection_point) < 0:
            intersections.append(t)
    return intersections

def intersect_wing1(ray):
    intersections = []
    if ray.direction.y != - ray.direction.z:  # Not parallel to wing1
        t = - (ray.origin.y + ray.origin.z)/(ray.direction.y + ray.direction.z)
        intersection_point = ray.origin + t * ray.direction
        if intersection_point.x >= 0:
            intersections.append(t)
    return intersections

def intersect_wing2(ray):
    intersections = []
    if ray.direction.y != ray.direction.z:  # Not parallel to wing2
        t = - (ray.origin.y - ray.origin.z)/(ray.direction.y - ray.direction.z)
        intersection_point = ray.origin + t * ray.direction
        if intersection_point.x <= 0:
            intersections.append(t)
    return intersections
```



Refactoring opportunities

- Initial example had 1 function with complexity 7
- Refactored code has 4 functions and an average complexity of 2.5
- Less complex code is easier to read and maintain



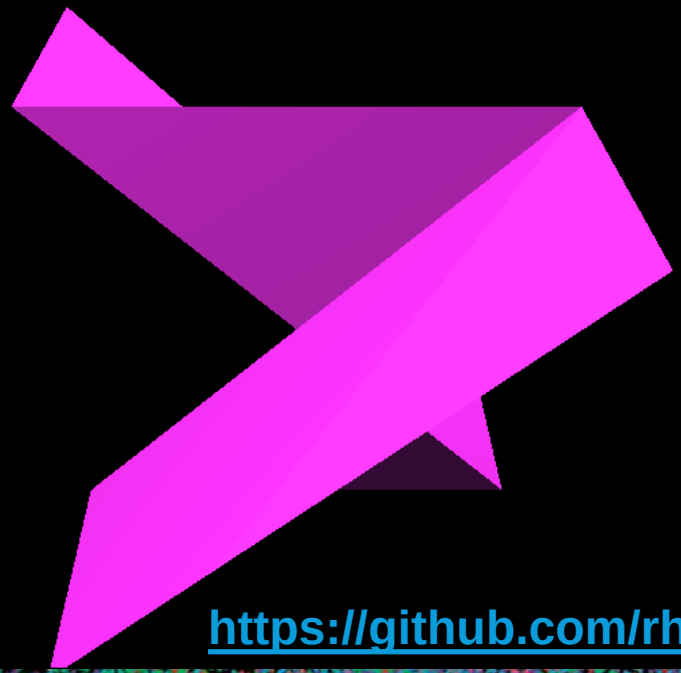
TechAtBloomberg.com

We are hiring!

https://www.bloomberg.com/careers
https://www.bloomberg.com/company/engineering-student-application-process/

Bloomberg

Engineering



https://github.com/rhoban13/ray_tracer

TechAtBloomberg.com