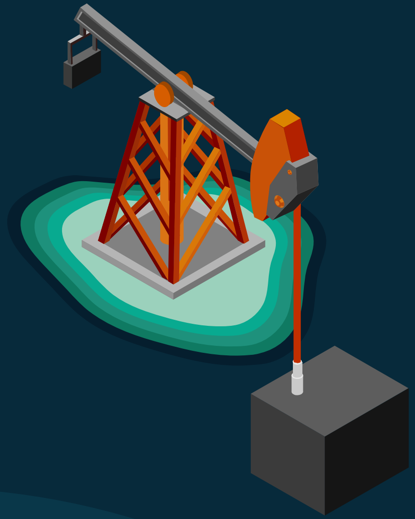
F.I.N. OffShore Resource Drill

Team Temple

Alexander Mot
Manraj Singh Rai
Christopher Tesar
Pardeep Singh Bhattal

Offshore drilling harms the environment

But, we still need resources. So how can we balance resource extraction while minimizing environmental harm?



Preservation Path

Mapping the Way to Sustainable Seas!

What is Preservation Path?

- Maps out preservation sites, including Coral Reefs, Habitats of Endangered Species, and Historical Shipwrecks.
- Plans and optimizes offshore resource drilling activities.
    - Factors in location of preservation sites, calculates most efficient paths for resource extraction while avoiding or minimizing disturbance to these sites.
- The UI presents data in a visually appealing and comprehensible manner, utilizing interactive map and dashboards.

o

o

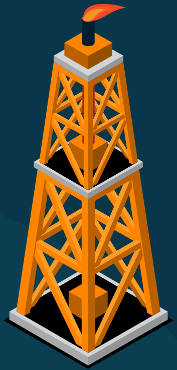
Design Process

Define Project Objectives

Clearly articulate the objectives and scope of the preservation path

Define the problem it aims to address and the desired outcomes it intends to achieve, such as optimizing offshore resource drilling while minimizing environmental impact.

Identify types of algorithms we could use for this system



Development Process

We broke the process down step by step!

1. We developed a UI to visualize our data.
    a. This was to help our algorithm as we could see how everything works step by step
2. We got to work on the algorithm.
3. Finally we built upon our earlier data visualizations to visually display the algorithm for route finding

Research and presentation progression was concurrently done.

Data Selection

- World Map
- Obtain:
  - Oil
  - Precious Metals
  - Helium
  - Shipwreck
- Preserve:
  - Coral Reef
  - Endangered Species
  - Shipwrecks
- Informational
  - Algal

Use of Data

The data was quite messy

We had to normalize and interpret the data. With the Numpy library

We assigned a score to every tile

o

This allowed for us to prioritize visiting certain spaces over others. Higher the score, higher the priority.

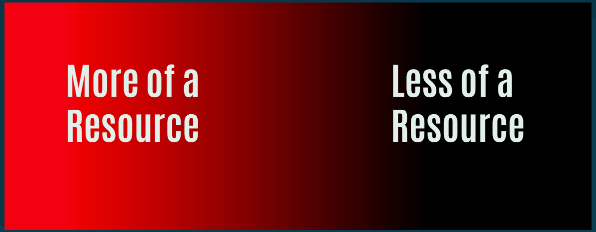
(s1) · Obtain – (s2) · Preserve + (s3) · Information

= Total Score

*Where s1,s2,s3 are scalar values

How does the user sees the data:

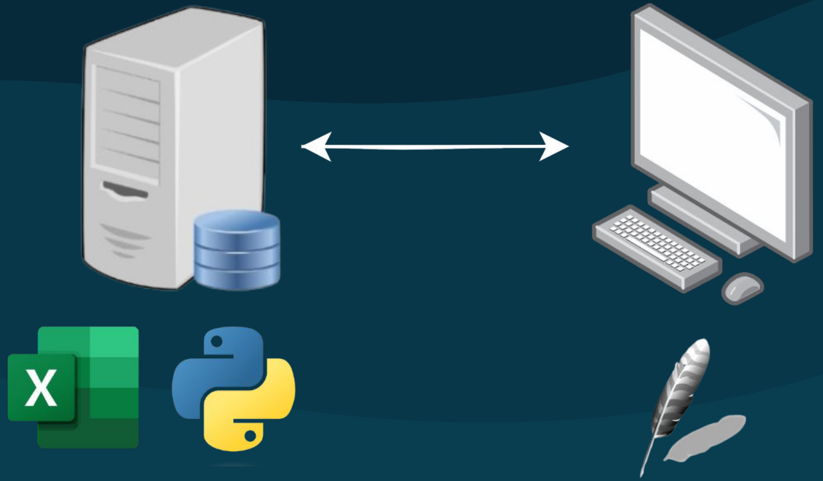
With our normalized data, we assigned scaled hex values with the specific data points for each coordinate. This allowed for us to have a gradient heatmap for each specific parameter.



Software Implementation

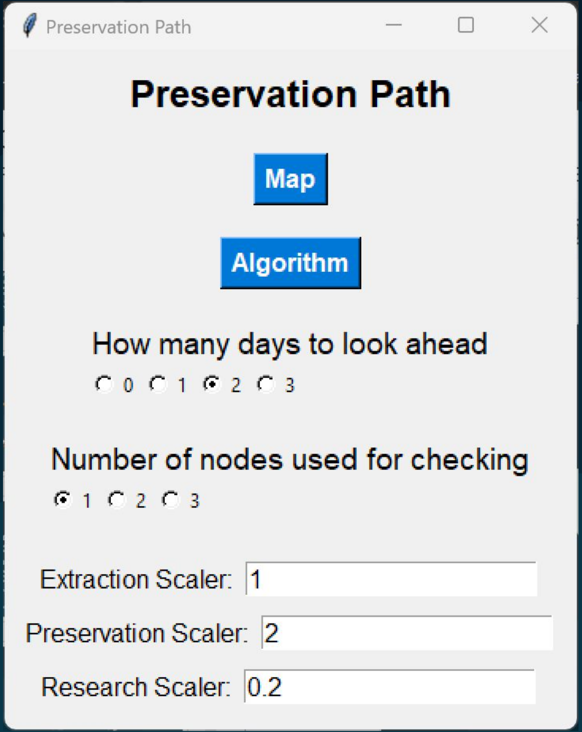
o

o



UI Design

o



Preservation Path

Preservation Path

Map

Algorithm

How many days to look ahead

○ 0 ○ 1 ● 2 ○ 3

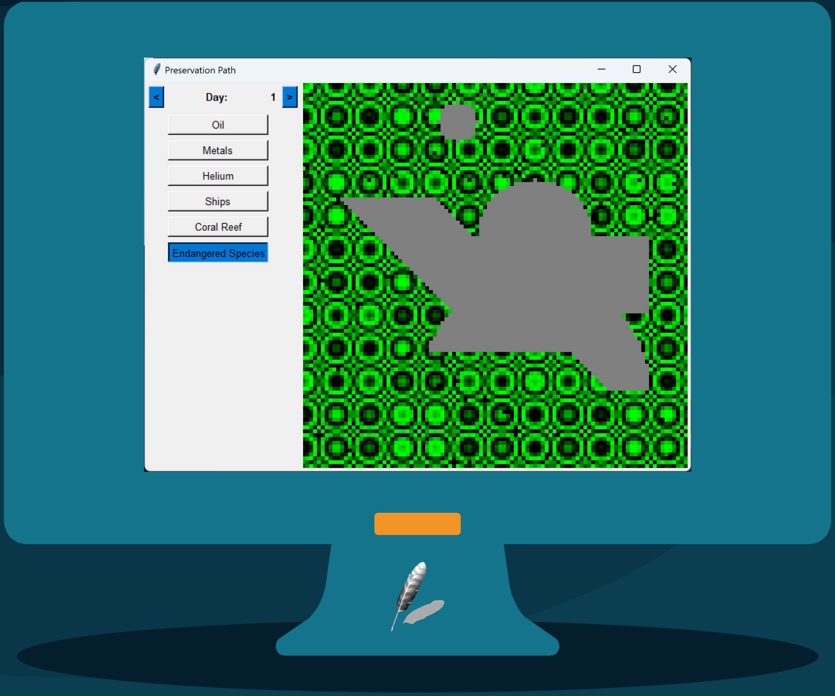
Number of nodes used for checking

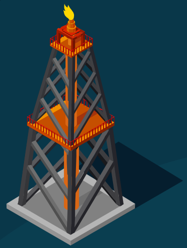
● 1 ○ 2 ○ 3

Extraction Scaler: 1

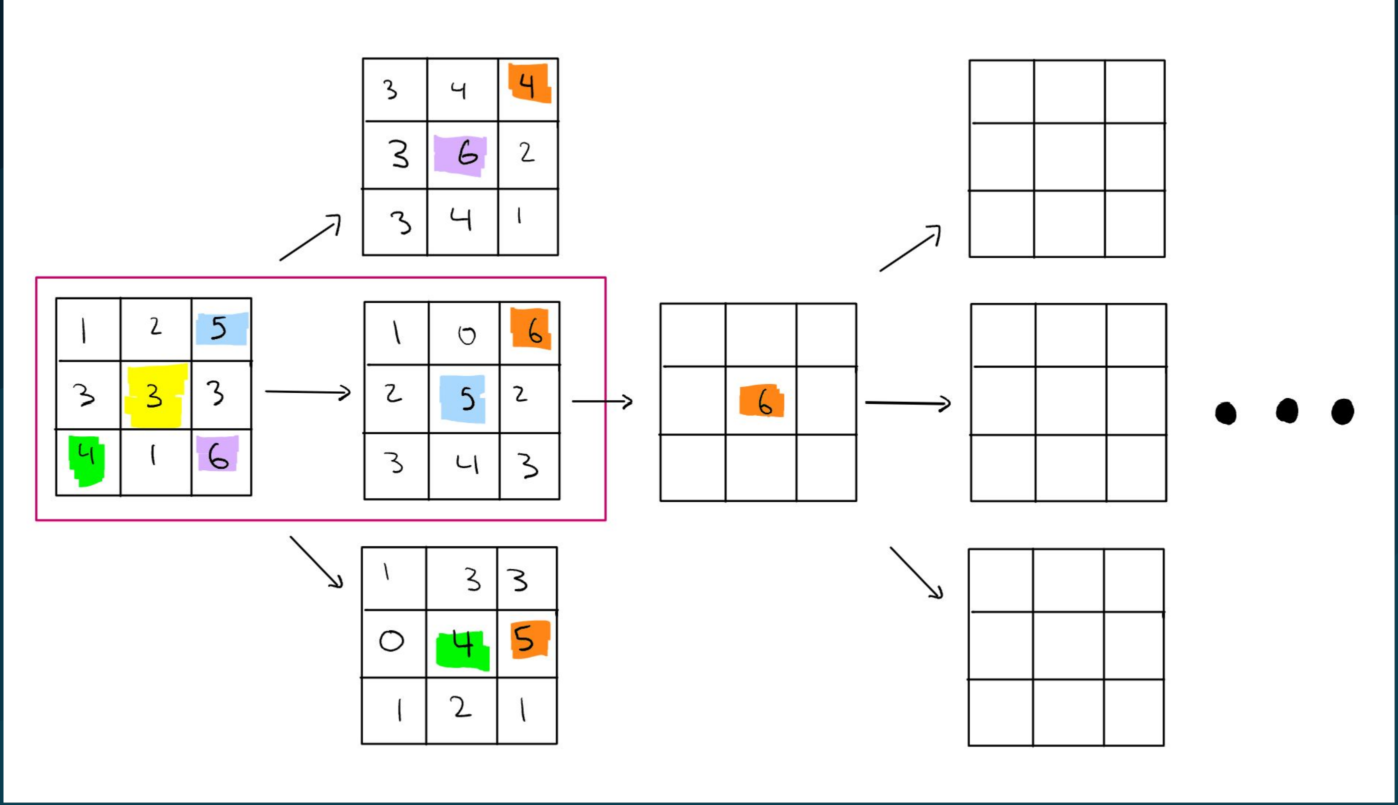
Preservation Scaler: 2

Research Scaler: 0.2





Algorithm Development



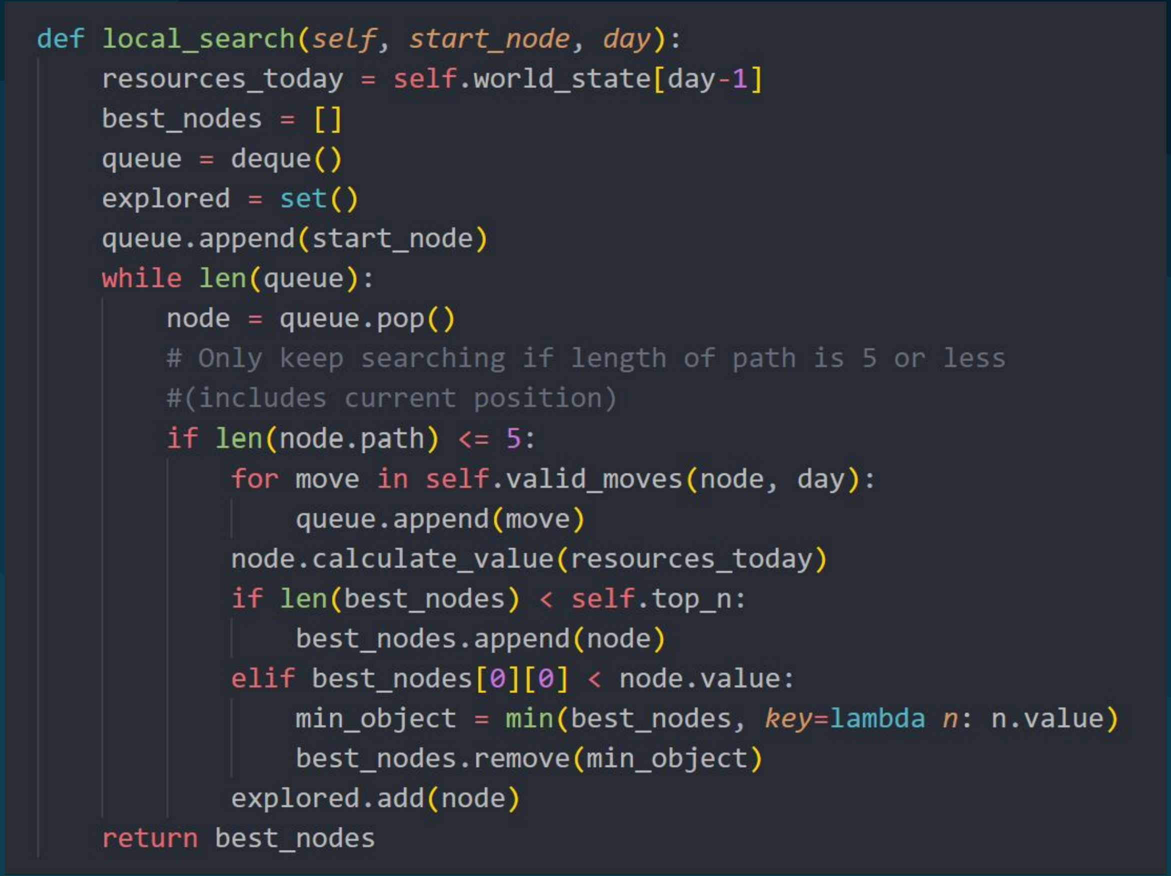
1	2	5
3	3	3
4	1	6

3	4	4
3	6	2
3	4	1

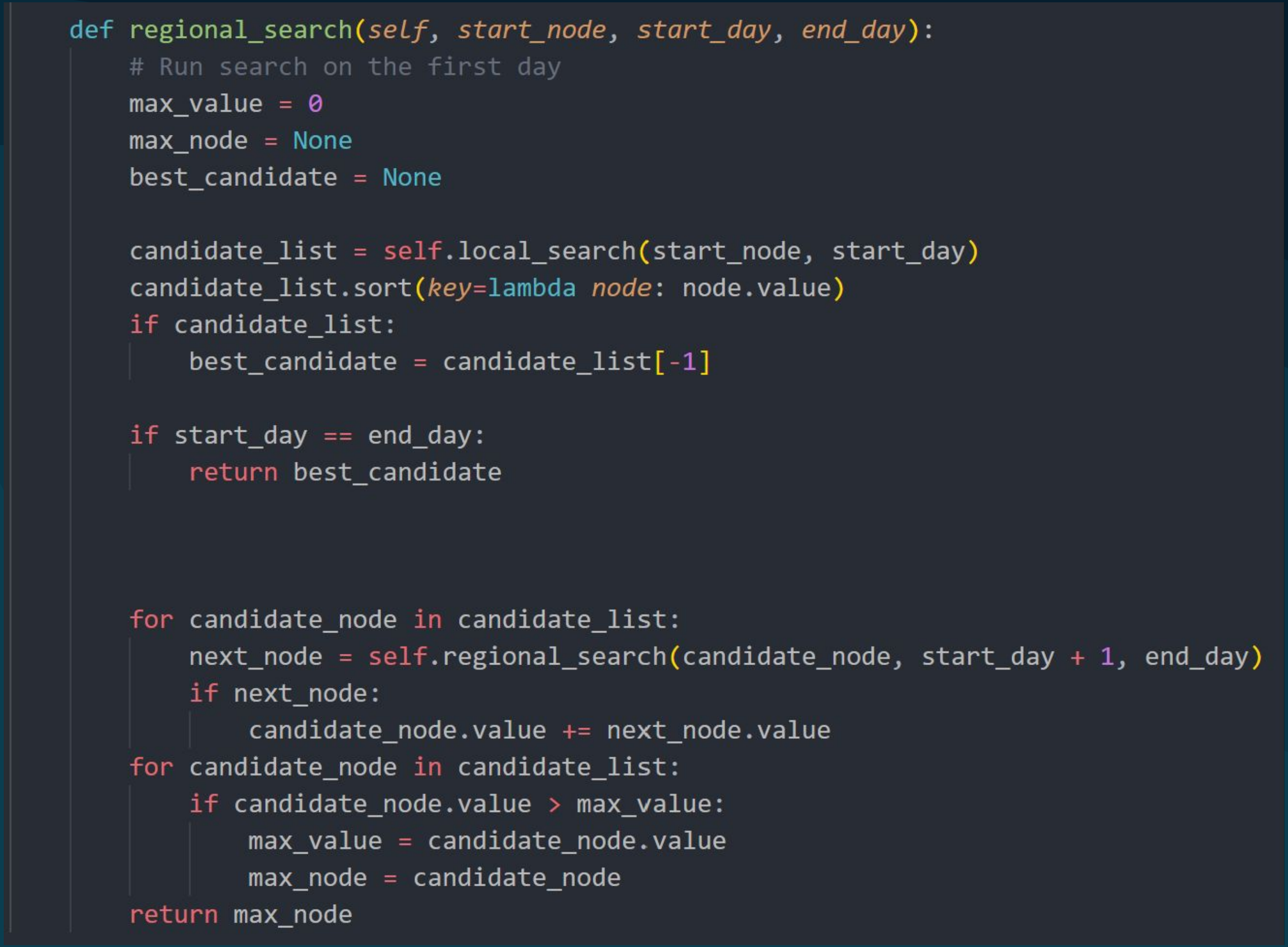
1	0	6
2	5	2
3	4	3

1	3	3
0	4	5
1	2	1

	6	



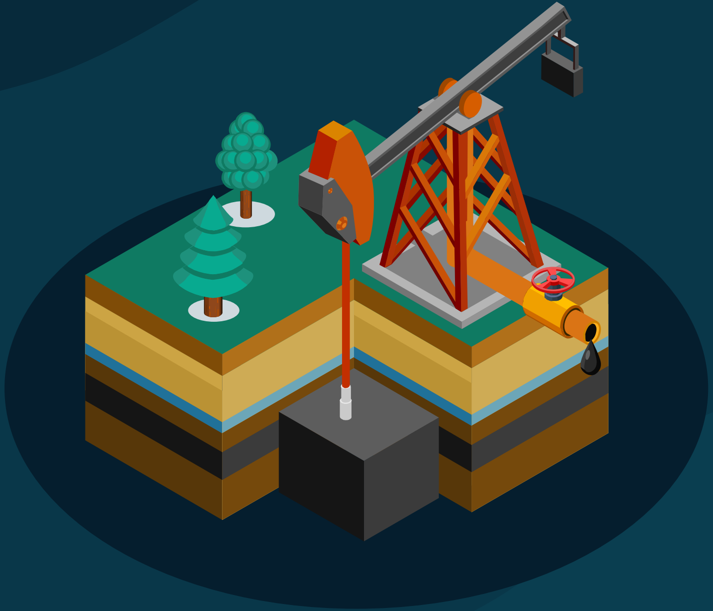
```python
def local_search(self, start_node, day):
    resources_today = self.world_state[day-1]
    best_nodes = []
    queue = deque()
    explored = set()
    queue.append(start_node)
    while len(queue):
        node = queue.pop()
        # Only keep searching if length of path is 5 or less
        #(includes current position)
        if len(node.path) <= 5:
            for move in self.valid_moves(node, day):
                queue.append(move)
            node.calculate_value(resources_today)
            if len(best_nodes) < self.top_n:
                best_nodes.append(node)
            elif best_nodes[0][0] < node.value:
                min_object = min(best_nodes, key=lambda n: n.value)
                best_nodes.remove(min_object)
            explored.add(node)
    return best_nodes
```



```python
    def regional_search(self, start_node, start_day, end_day):
        # Run search on the first day
        max_value = 0
        max_node = None
        best_candidate = None

        candidate_list = self.local_search(start_node, start_day)
        candidate_list.sort(key=lambda node: node.value)
        if candidate_list:
            best_candidate = candidate_list[-1]

        if start_day == end_day:
            return best_candidate



        for candidate_node in candidate_list:
            next_node = self.regional_search(candidate_node, start_day + 1, end_day)
            if next_node:
                candidate_node.value += next_node.value
        for candidate_node in candidate_list:
            if candidate_node.value > max_value:
                max_value = candidate_node.value
                max_node = candidate_node
        return max_node
```

Future Implementations

- Further frontend development to make it even more appealing for users
- More robust data visualization
    - Displaying heatmaps of all resources selected simultaneously
- Implement more advanced algorithms
    - Machine Learning Algorithms:
        - SVM, Random Forest, Clustering, etc.
- Accommodate both professionals in their everyday tasks and researchers exploring new avenues of inquiry.



DEMO

Thank You

Do you have any questions?

CREDITS: This presentation template was created by Slidesgo, and includes icons by Flaticon, and infographics & images by Freepik