Rundle Retrievers

By: Hetarthi Soni, Hamnah Qureshi, Matteo Golin, and Grant Achuzia

Table of contents

01 Project Overview

02 Target Audience

03 Design Process

04 Solution

05 Missing Functionality

06 Citations

Project Overview

GOAL: To optimize the extraction of coastal resources without destroying the environment

Project Overview

Datasets:

- Coastal resources
- Coastal preservation priorities
- Land map
- Environmental conditions

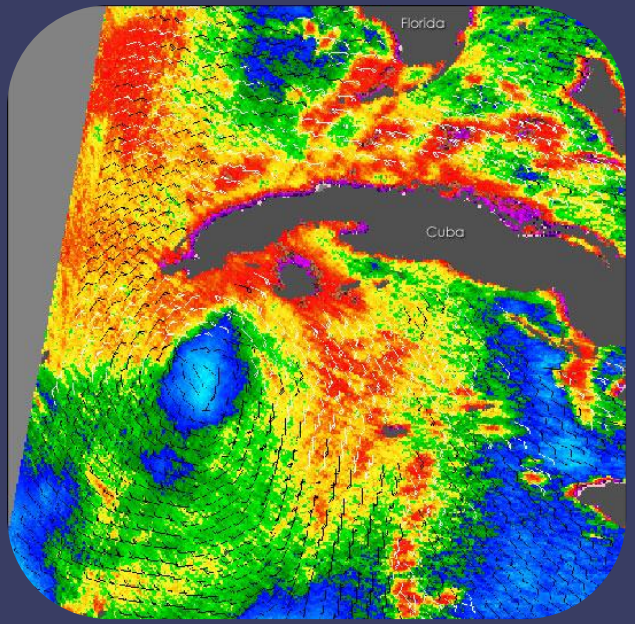

Figure 1: Wind patterns

Project Overview

- Creating an algorithm and user interface
- Algorithm must provide the optimal location for each day
- The interface must visualize optimal locations

Figure 2: Map [6]

Target Audience

- WANBIS Corp.
  - Help WANBIS create software that finds the optimal rig location
- Fictitious Island (F.I.N)
  - Ensure that we protect F.I.N's environment and its people by prioritizing sustainability

Figure 3 : An island like F.I.N. [3]

Design Process

- Resource Obtained: **Oil**
  - Assumption: Oil produces the most profit out of all the resources
- Resource Preserved: **Coral Reefs**
  - Is a diverse ecosystem which includes many living organisms
  - Highest impact if lost

Figure 4: Coral ecosystem [4]

Design Process

Backend:

- Informational data considered irrelevant
- Difficult to optimize a best path without considering all scenarios (moving resources in time)
- Heuristics need to be chosen for optimal calculation

Design Process

Backend:

- Inspiration from A*
- Strategy pattern
- Implemented 2 algorithms (Max and Random)
- Context from the tiles surrounding drill
- Extensible

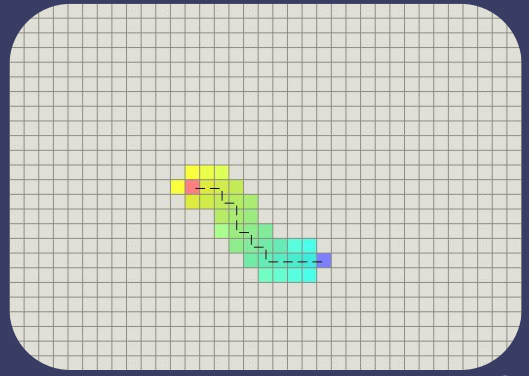

Figure 5: A* algorithm [1]

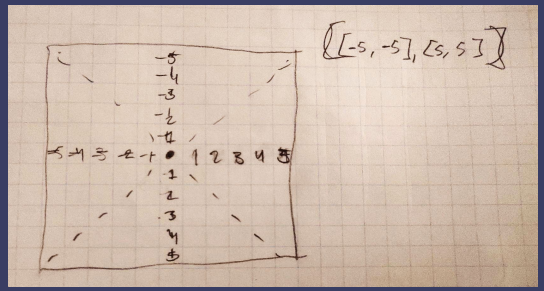

Figure 6: Rig movement radius

Design Process

Backend:

- Resource values normalized
- Account for sustainability in path-finding
- Mask out land
- Reduce values by resource to preserve

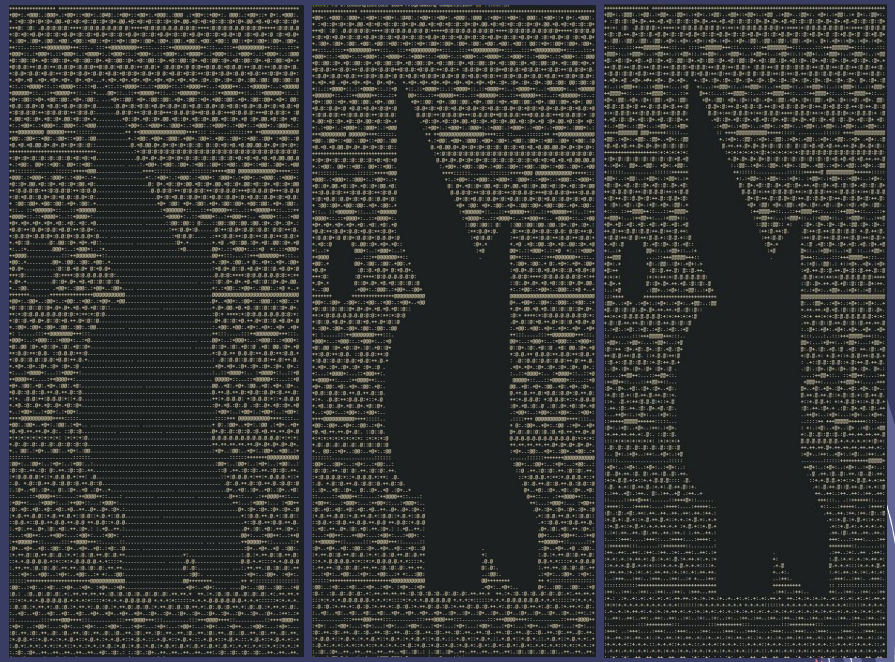

Figure 7: Oil map before and after land mask application

Design Process

Backend:

- State machine-like API
  - Computes next drill state
  - Collects next resources
  - Cycles back to 0 after completion

Design Process

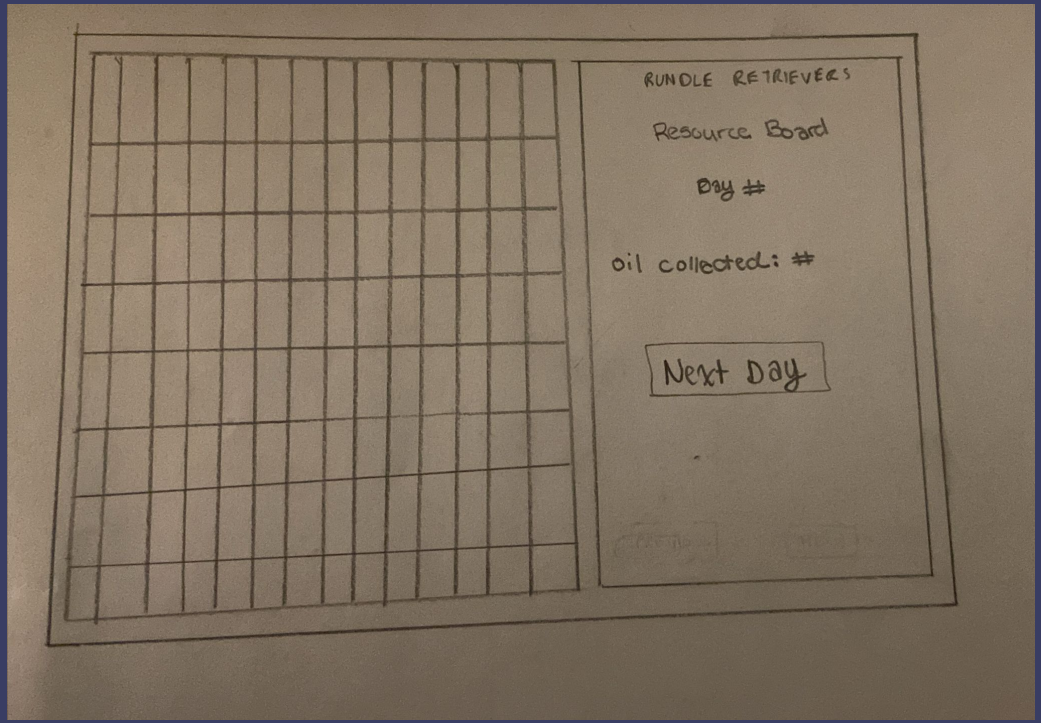

Figure 8: Brainstormed UI

Design Process

Frontend:

- HTML/CSS, JavaScript, and Python
- Map canvas developed using JavaScript
- Generated map rendered on the browser using Flask
- Displays retrieved statistics with HTML and Python

Solution

- Easy graphical representation of the algorithm

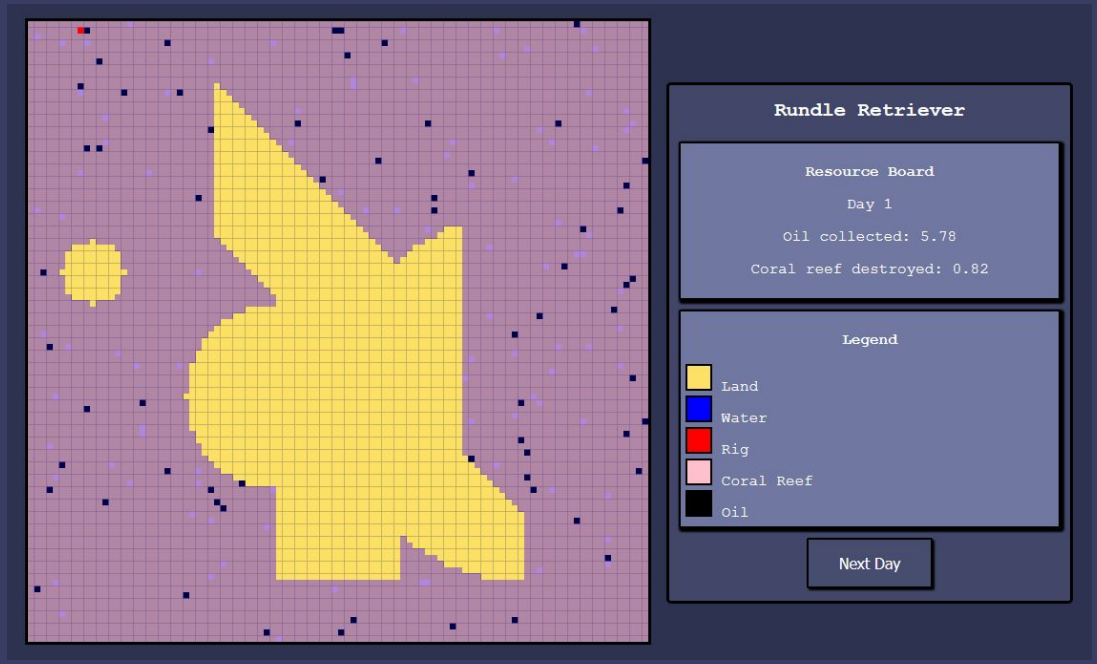

Figure 9: User interface

Missing Functionality

- Does not consider any resource besides coral and oil
- Could make assumptions about utility of information data
- Implement more strategies
- Allow strategy selection from the UI

```
# TODO: Features to implement
features = [
  "more strategies",
  "UI dropdown",
  "consider all resources",
  "use information data"
]
```

Citations

[1] A. Patel, “Introduction to A*,” Stanford.edu, 2019. http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

[2] NOAA, “Coral reef ecosystems | National Oceanic and Atmospheric Administration,” www.noaa.gov, Feb. 01, 2019. https://www.noaa.gov/education/resource-collections/marine-life/coral-reef-ecosystems#:~:text=Benefits%20of%20coral%20reef%20ecosystems

[3]“The 15 Best Islands in the US,” TripSavvy. https://www.tripsavvy.com/best-islands-in-the-us-6979593 (accessed Mar. 02, 2024)

[4]“A Probiotic to Protect Caribbean Corals,” The Scientist Magazine®. https://www.the-scientist.com/a-probiotic-to-protect-caribbean-corals-71204 (accessed Mar. 02, 2024).

[5]“What is biodiversity and why is it so important?,” Great Barrier Reef Foundation, Sep. 01, 2023. https://www.barrierreef.org/news/explainers/what-is-biodiversity-and-why-is-it-so-important#:~:text=Why%20is%20that%20important%3F

[6] A_Y_N, Topographic map and direction marker abstract background. Accessed: Mar. 02, 2024. [Online]. Available: https://stock.adobe.com/ca/images/topographic-map-and-direction-marker-abstract-background-outline-cartography-landscape-topographic-relief-map-modern-cover-design-with-wavy-lines-vector-card-with-weather-map-outline-pattern/445930090

Appendix

```python
class Strategy(ABC):
    """An abstract class to define the interface that all movement strategies must implement."""

    @abstractmethod
    def next(self, neighbourhood: Neighbourhood) -> Coord:
        """Given a neighbourhood of resources within the drill's movement range, returns the next move."""
        pass
```

Appendix

```python
class MaxStrat(Strategy):
    """Moves the drill to the maximum valued resource in its neighbourhood."""

    def next(self, neighbourhood: Neighbourhood) -> Coord:
        """
        Returns the coordinates of the maximum valued resource in the neighbourhood.
        Args:
            neighbourhood: The neighbourhood to chose a move from.
        Returns:
            The coordinate that the drill should move to.
        """

        max_val: float = float("-inf")
        max_coord: Coord = (0, 0)

        for coords, tile in neighbourhood:
            if tile.value > max_val:
                max_val = tile.value
                max_coord = coords

        return max_coord
```

Appendix

```python
class ResourceType(StrEnum):
    """Possible resource types on the map that can be preserved or exploited."""

    OIL = "Oil"
    HELIUM = "Helium"
    PRECIOUS_METALS = "Metal"
    SHIPWRECK = "Shipwreck"
    CORAL = "Coral"
    ENDANGERED = "Endangered"
```

Appendix

```python
@dataclass
class Resource:
    """Represents a resource of a specific type and value."""

    rtype: ResourceType
    value: float

    def __str__(self) -> str:
        """Clear string representation of a resource with value and type."""
        return f"({self.rtype.value}, {self.value})"

    __repr__ = __str__
```

Appendix

```python
def load_resource(filepath: str, rtype: ResourceType) -> ResourceMap:
    """
    Loads resource data from a CSV file into a resource map.
    Args:
        filepath: The path to the CSV file containing the resource data.
        rtype: The resource type that is being loaded.
    Returns:
        A 100x100 2D array containing resources at their respective locations.
    """

    map: ResourceMap = [[None for _ in range(WORLD_WIDTH)] for _ in range(WORLD_HEIGHT)]

    with open(filepath, "r") as file:
        reader = csv.reader(file)
        _ = next(reader)  # Skip over headers always in the order of ID, x, y, value

        for _, x, y, value in reader:

            # If there is missing data, assign the value to be negative infinity
            # This makes unknown locations our lowest priority unless no other tiles in a 5 tile radius contain
a
            # resource
            if not value:
                value = float("-inf")

            # Convert string values into their numerical representation
            x = int(x)
            y = int(y)
            value = float(value)

            map[y][x] = Resource(rtype, value)

    normalize_resources(map)
    return map
```

Appendix

```python
def normalize_resources(resources: ResourceMap) -> None:
    """
    Normalizes all resource values in the resource map to have a minimum value of 0.
    Args:
        resources: The resource to normalize.
    """

    # Get the minimum resource value
    minimum = float("inf")
    for row in resources:
        for tile in row:
            if tile is not None and tile.value != float("-inf") and tile.value < minimum:
                minimum = tile.value

    # Add the absolute value of the minimum resource to bring all resource values to >= 0
    for row in resources:
        for tile in row:
            if tile is not None:
                tile.value += abs(minimum)
```