Eagle Eye

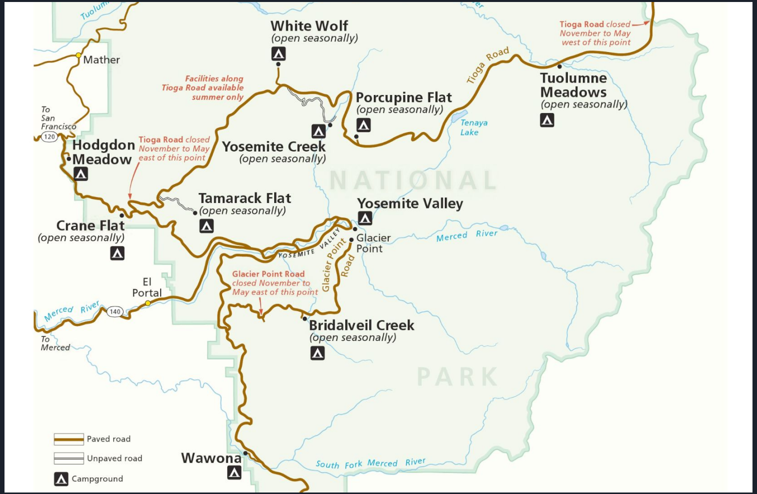
By Ahmed Khan, Adam Freed, Gavin Silva
Team Skynet

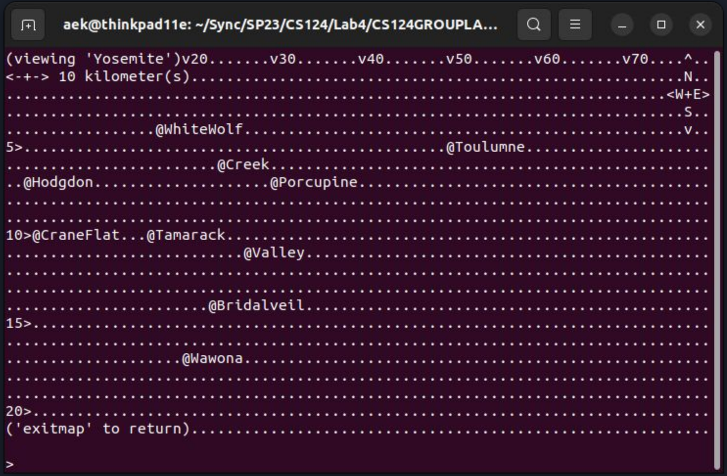
Context and Rationale

- Context: Made to meet UN SDG 15’s “aims to protect, restore, and promote sustainable use of terrestrial ecosystems, sustainably manage forests, combat desertification, and halt and reverse land degradation and halt biodiversity loss” (1)
- Impact: Enables user to visually track of points of interests and their attributes in a nature setting, and find distances and paths to reach them, such as forest fires, water sources, trailheads, etc

What does it do?

- Visualize points of interest within a natural dominion such as campgrounds, peaks, trailheads, wildfires, etc.
- Attribute each point an unbounded set of unique features such as fire risk, flood risk, altitude, wildlife prevalence
- Rank relevant points by feature in ascending order of importance
- Find paths between points in a nonlinear BFS, and euclidean distance based on provided scale for rangers to address needs





```
(viewing 'Yosemite')v20.......v30.......v40.......v50.......v60.......v70....^..
<-+-> 10 kilometer(s)........................................................N..
...........................................................................<W+E>
.............................................................................S..
.................@WhiteWolf..................................................v..
5>................................................@Toulumne.....................
........................@Creek..................................................
..@Hodgdon....................@Porcupine........................................
................................................................................
................................................................................
10>@CraneFlat...@Tamarack.......................................................
...........................@Valley..............................................
................................................................................
................................................................................
.......................@Bridalveil..............................................
15>.............................................................................
................................................................................
....................@Wawona.....................................................
................................................................................
................................................................................
20>.............................................................................
('exitmap' to return)...........................................................

>
```

Work Strategy

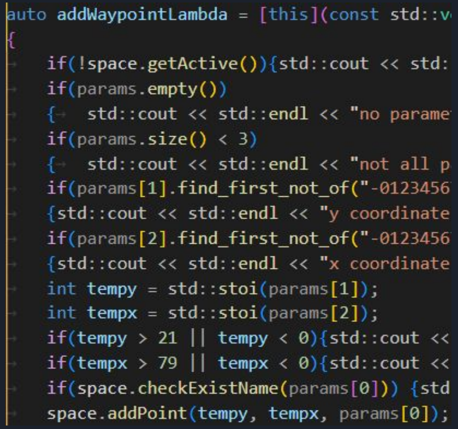
- Github repository allowing delegation of labor, resiliency in case of error, and history of development through commits
- Ahmed: event handler and ascii visualizer
- Gavin: algorithms and presentation
- Adam: data structures and report

General design and algorithms

- Local SC Hashmap for O(1) waypoint reference and check
- STL Vector for O(n) for iteration for visualizer, listing, sorting
- Local LL Stack for undo addition of previous waypoint
- Nested Map Graph of distances for BFS pathfinder
- STL Map for event handler lambdas, with a runtime loop that calls one each time the user specified a command, and input validation and data manipulation occurred in the backend before returning to the next loop to ask user for command

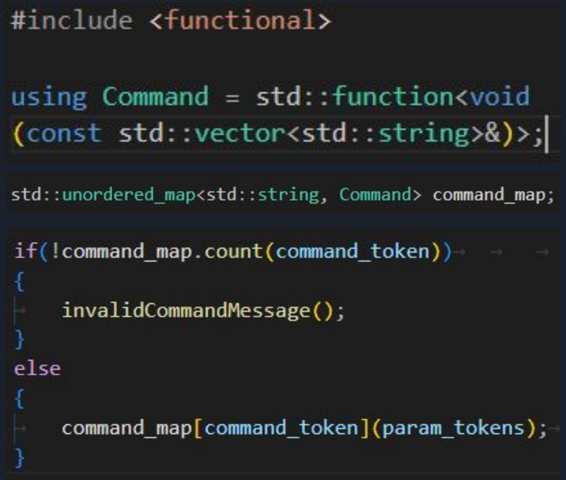
Code Showcase

Robust guard clauses and feedback for every command lambda, purpose is preventing undefined behavior at runtime



```cpp
auto addWaypointLambda = [this](const std::v
{
    if(!space.getActive()){std::cout << std:
    if(params.empty())
    {   std::cout << std::endl << "no parame
    if(params.size() < 3)
    {   std::cout << std::endl << "not all p
    if(params[1].find_first_not_of("-0123456
    {std::cout << std::endl << "y coordinate
    if(params[2].find_first_not_of("-0123456
    {std::cout << std::endl << "x coordinate
    int tempy = std::stoi(params[1]);
    int tempx = std::stoi(params[2]);
    if(tempy > 21 || tempy < 0){std::cout <<
    if(tempx > 79 || tempx < 0){std::cout <<
    if(space.checkExistName(params[0])) {std
    space.addPoint(tempy, tempx, params[0]);
```

Command based referencing map of lambdas to call functions, purpose is to allow user flexibility and options, instead of traversing switch cases with enums, user is free to choose what they want to do with their data.



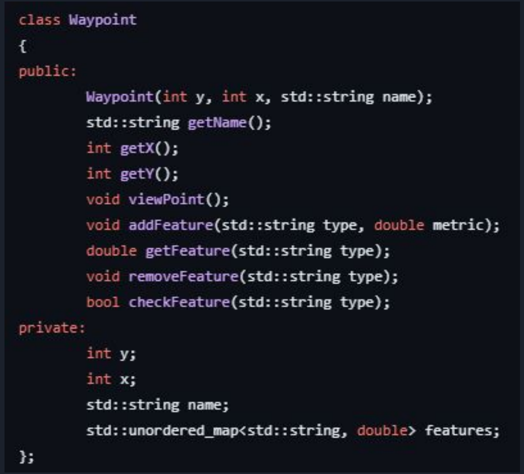
```cpp
#include <functional>

using Command = std::function<void
(const std::vector<std::string>&)>;
```

```cpp
std::unordered_map<std::string, Command> command_map;
```

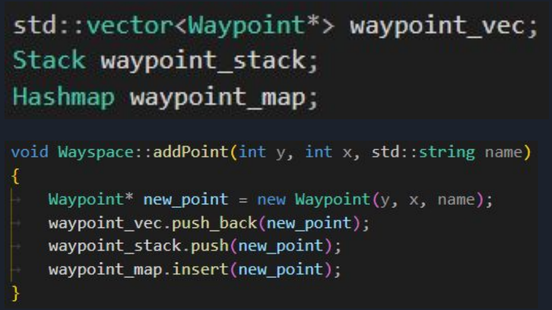
```cpp
if(!command_map.count(command_token))
{
    invalidCommandMessage();
}
else
{
    command_map[command_token](param_tokens);
}
```

Waypoint object, managed by Wayspace class, is core program. All functionality centers around waypoint manipulation. Enables encapsulation and function calls to manipulate waypoints rather than having to alter data.



```cpp
class Waypoint
{
public:
        Waypoint(int y, int x, std::string name);
        std::string getName();
        int getX();
        int getY();
        void viewPoint();
        void addFeature(std::string type, double metric);
        double getFeature(std::string type);
        void removeFeature(std::string type);
        bool checkFeature(std::string type);
private:
        int y;
        int x;
        std::string name;
        std::unordered_map<std::string, double> features;
};
```

Simultaneous data structures, for various purposes such as reference, listing or undo, in sync to prevent any memory leaks or segmentation faults.



```cpp
std::vector<Waypoint*> waypoint_vec;
Stack waypoint_stack;
Hashmap waypoint_map;
```

```cpp
void Wayspace::addPoint(int y, int x, std::string name)
{
    Waypoint* new_point = new Waypoint(y, x, name);
    waypoint_vec.push_back(new_point);
    waypoint_stack.push(new_point);
    waypoint_map.insert(new_point);
}
```

Issues and Challenges

- Changes in plans: there was much deviation from the initial flowchart and presumed functionality based on time and requirement constraints. File IO was scrapped entirely for example, and data structure use cases had to be changed
- Complicated bugs: occasional segmentation faults with no indication of where they were occurring required closer looks
- Communication: coordination, keeping on schedule, and delegation of labor led to crunch time for particular sections

References

https://sdgs.un.org/topics/forests

https://www.nps.gov/yose/planyourvisit/campgrounds.htm