Strategy Pattern Exercise

Geographical Information Systems (GIS)

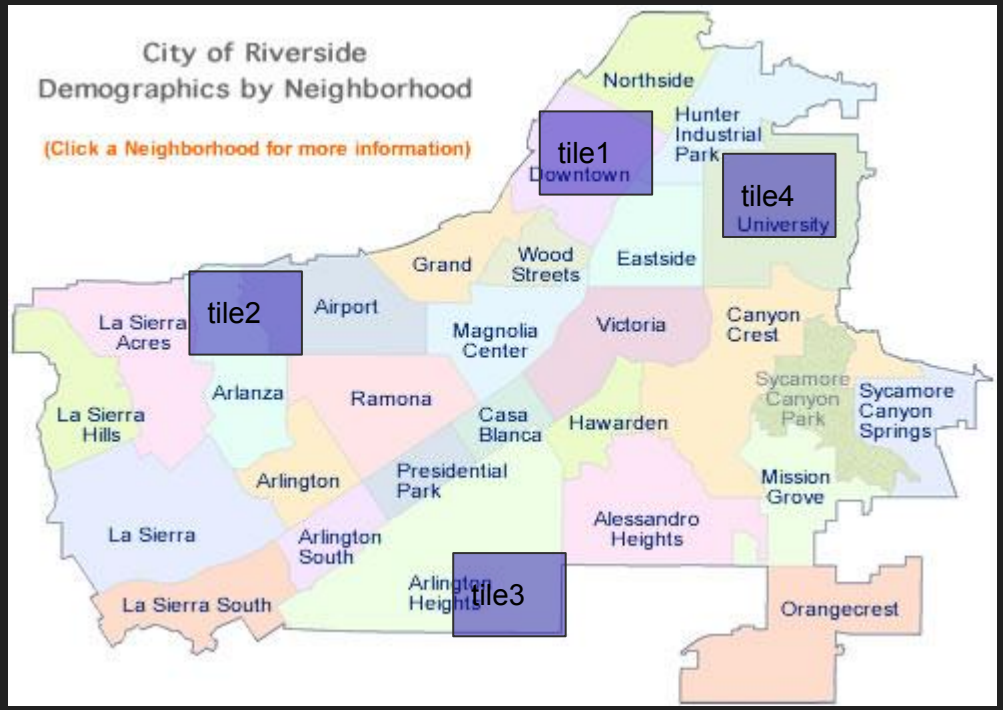
Remember find_poi()?

```
class Area : public Tile {
    ….
    Public:
        Point find_poi(string name,Point location) {
            Point current_closest = Point(-1,-1);
            double current_distance = DBL_MAX;
            for(unsigned i = 0;i < this->tiles.size();i++) {
                Point new_location = this->tiles.at(i).find_poi(name,location);
                if (new_location != Point(-1,-1)) {
                    if (distance(location,new_location) < current_distance || current_closest == (-1,-1)) {
                        current_closest = new_location;
                        current_distance = distance(location,new_location);
                    }
                }
            }
            return current_closest;
        }
    ….
```

Currently we search blindly

Lots of area to search

Even just within the city



Every tile needs to be searched

- UCR encompasses 3 sq miles
- Riverside: 97 sq miles
- California: 163,696 sq miles
- The USA: 3,796,742 sq miles
- Planet Earth: 197,000,000 sq miles

We would need to search **197 million** tiles to get our answer

How can we improve the way we are searching?

Directed Search

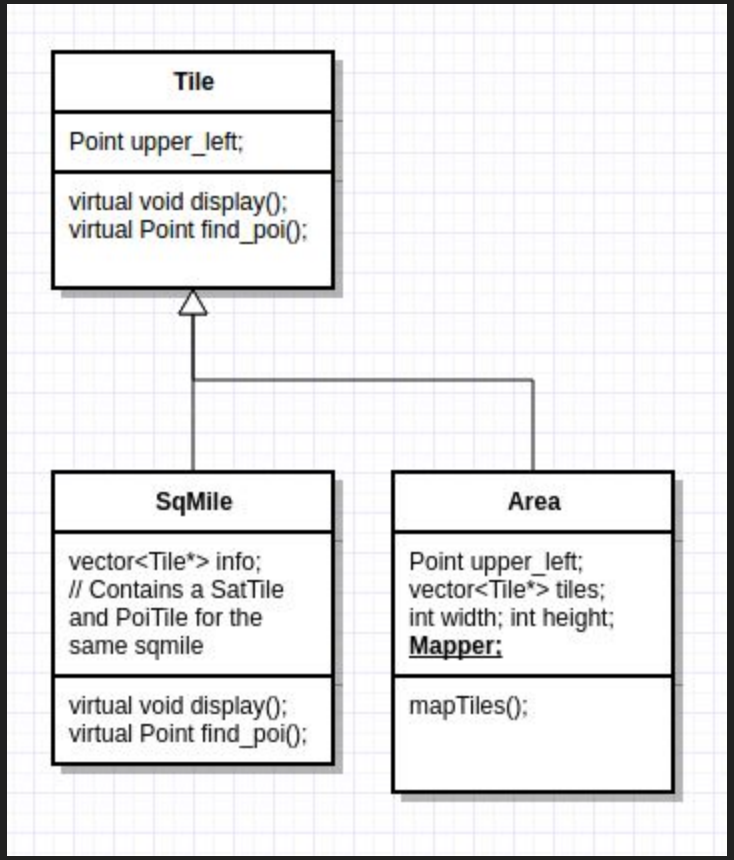
```
Class Area : public Tile {
  Public:
    Point find_poi(string name,Point location) {
      Point current_closest = Point(-1,-1);
      double current_distance = -1;
      tiles.sort_by_distance_to(location);  // we can direct the search so we can early abandon
      for(unsigned i = 0;i < this->tiles.size();i++) {
        Point new_location = this->tiles.at(i).find_poi(name,location);
        if (new_location != (-1,-1)) {
          return new_location;
        }
      }

     return Point(-1,-1);
    }
};
```

How efficient is this search?

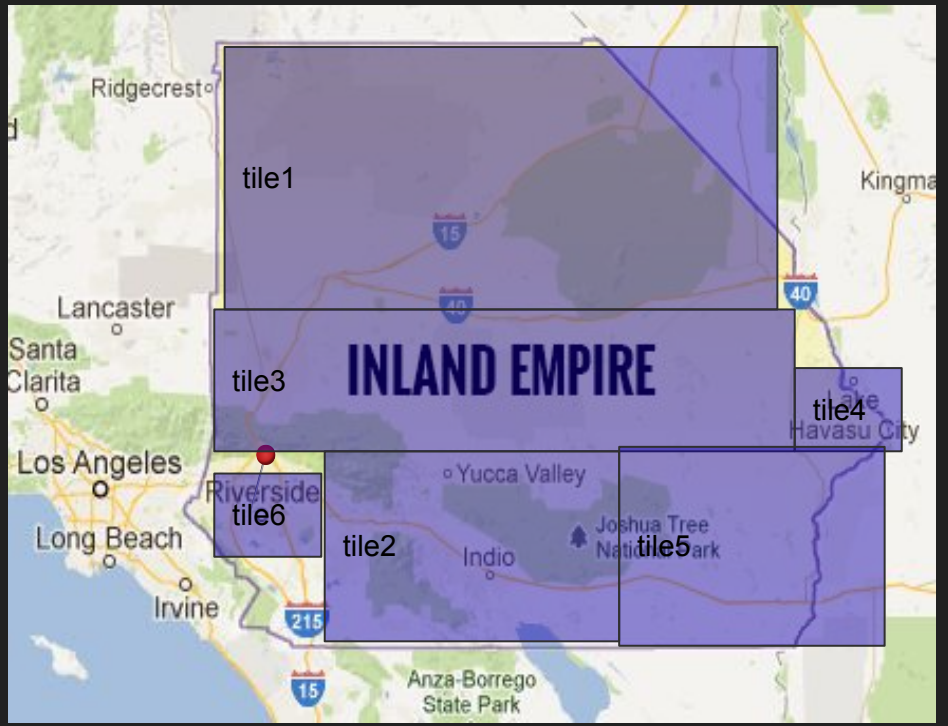
What can we do to make it more efficient?

Hierarchy & Organization

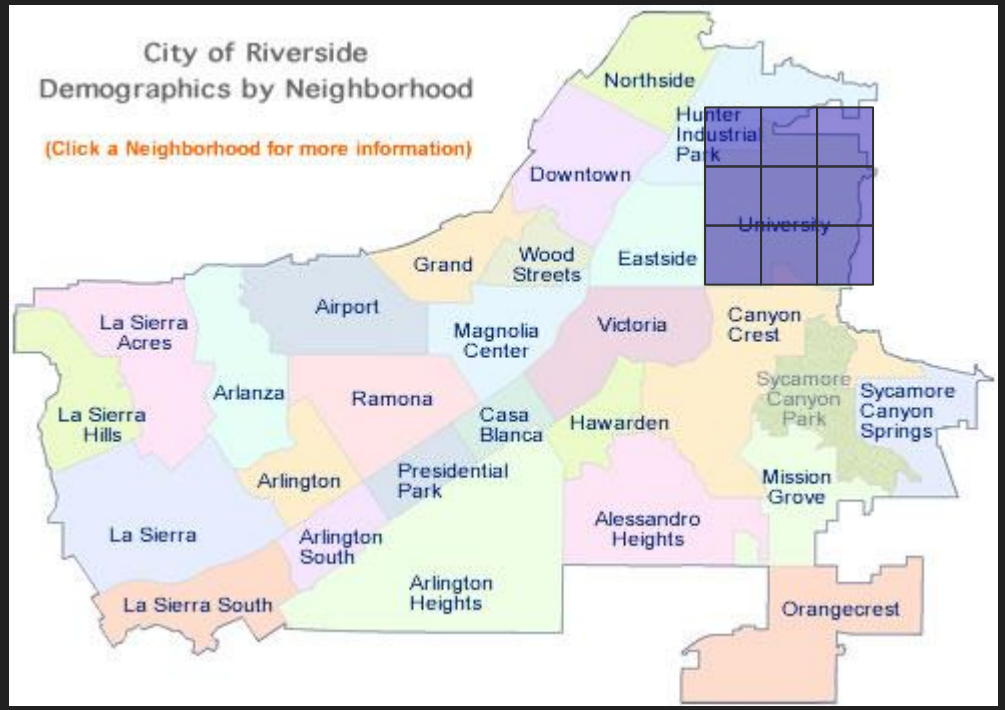

Tile
Point upper_left;
virtual void display();
virtual Point find_poi();
SqMile
vector<Tile*> info;
// Contains a SatTile
and PoiTile for the
same sqmile
virtual void display();
virtual Point find_poi();
Area
Point upper_left;
vector<Tile*> tiles;
int width; int height;
Mapper:
mapTiles();

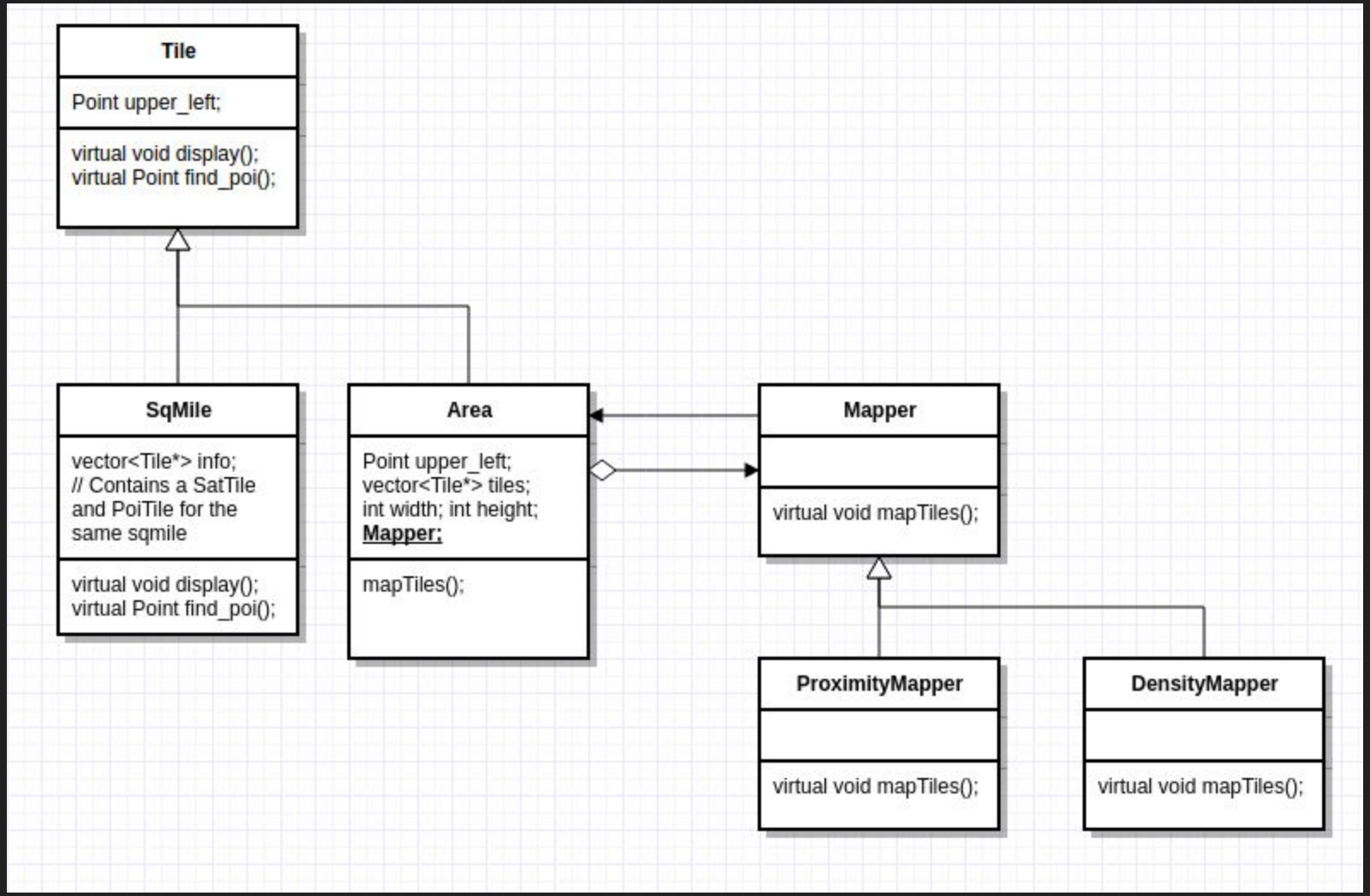
Proximity

Density

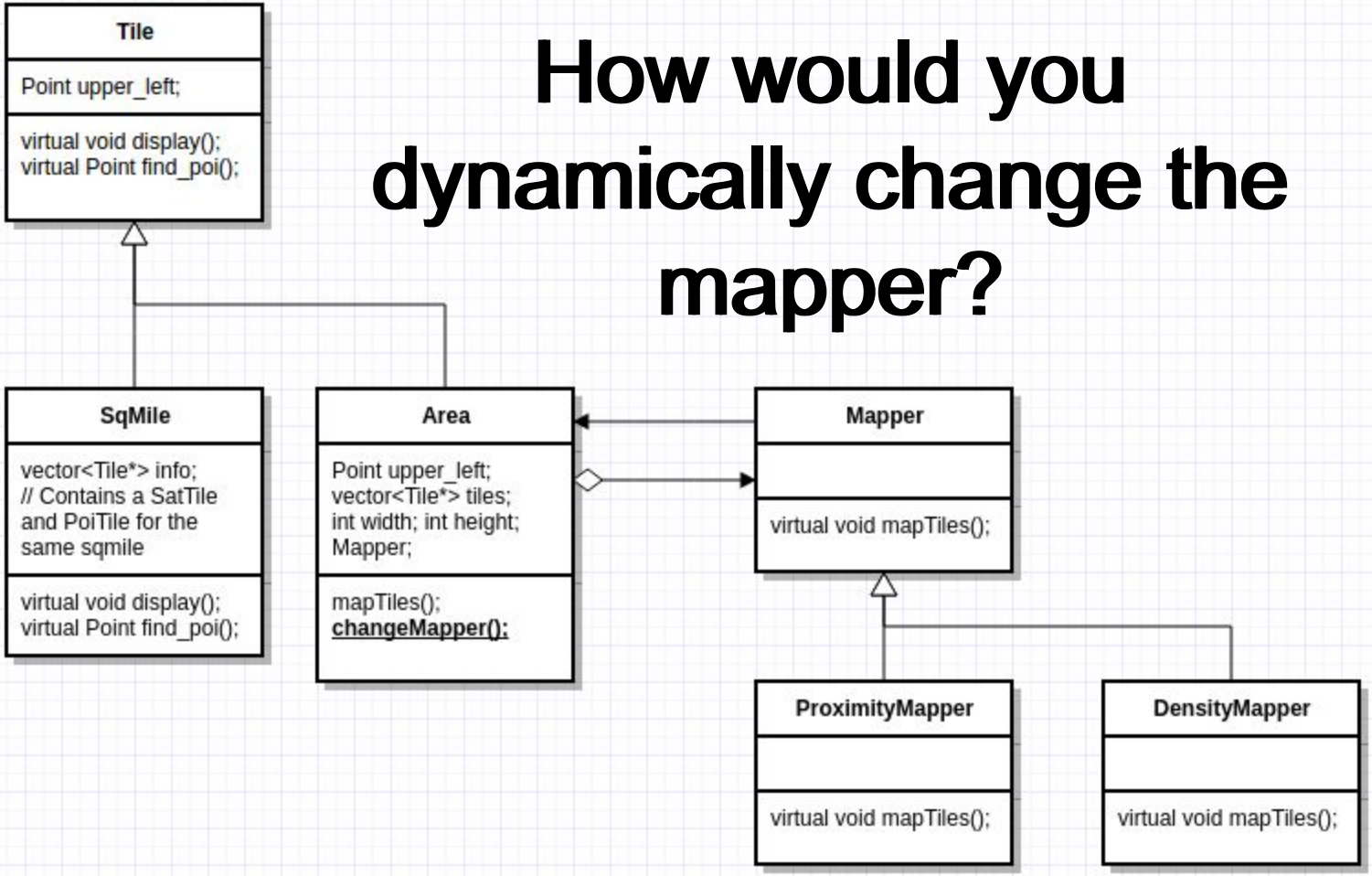


Using proximity organization




Tile
Point upper_left;
virtual void display();
virtual Point find_poi();
SqMile
vector<Tile*> info;
// Contains a SatTile
and PoiTile for the
same sqmile
virtual void display();
virtual Point find_poi();
Area
Point upper_left;
vector<Tile*> tiles;
int width; int height;
Mapper:
mapTiles();
Mapper
virtual void mapTiles();
ProximityMapper
virtual void mapTiles();
DensityMapper
virtual void mapTiles();

How would you dynamically change the mapper?



How would you add a mapping strategy?