Fixing recursive header calls

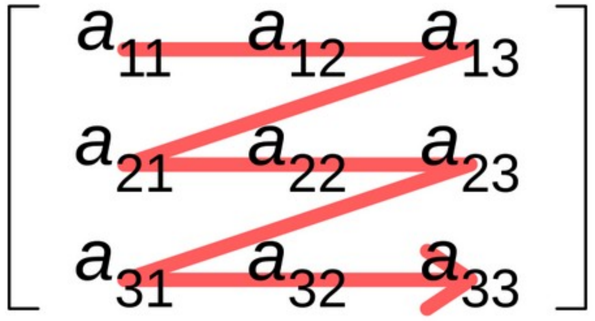
Goals

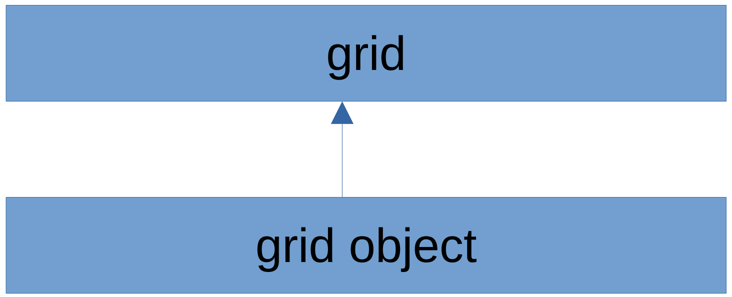
- Have a grid with pointers to objects
- Objects themselves can make a move on the grid
  - Implies including of grid header

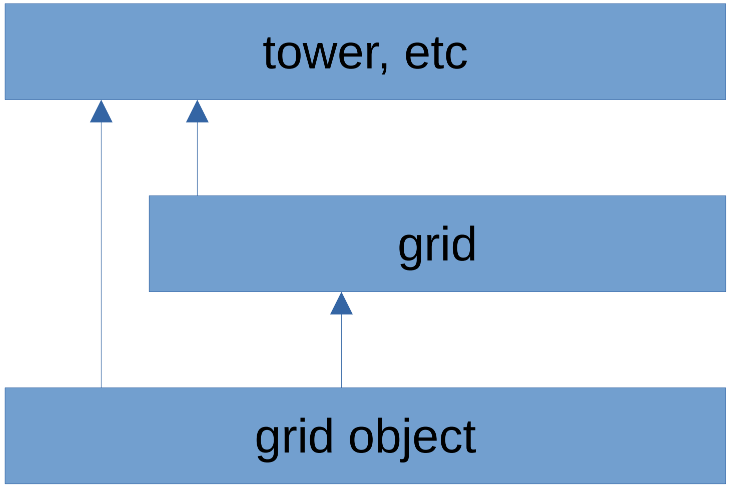
```
class Grid {
    Public:


    std::vector<std::vector<grid_object*>> grid;
```

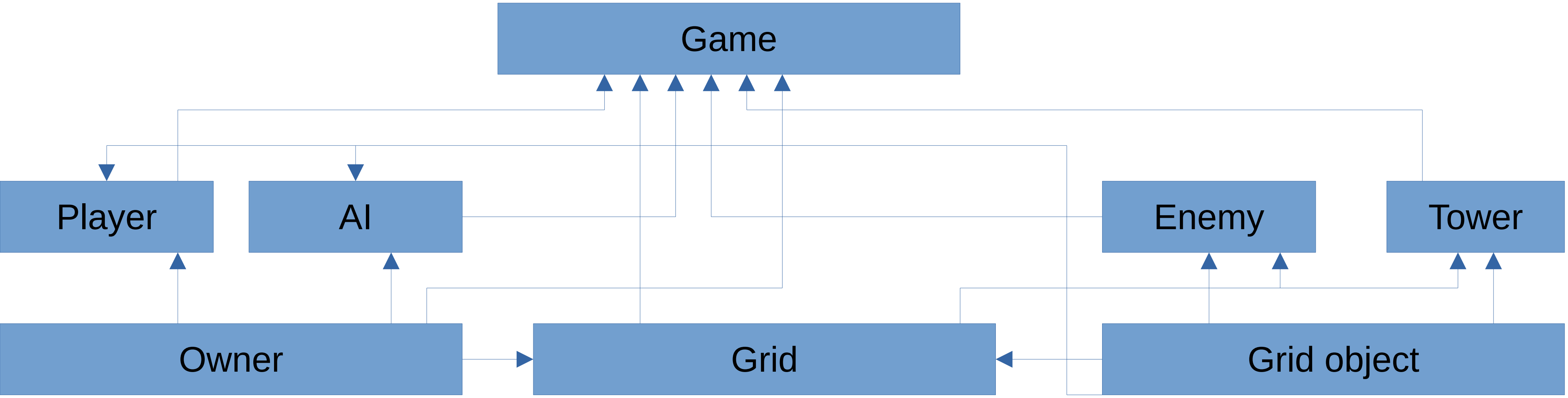
```
class Grid {
    Public:


    std::vector<grid_object*> grid;


    T* operator()(int row, int col) {
        return grid[row * cols + col];
    }
```

Row-major order


a_{11} a_{12} a_{13}
a_{21} a_{22} a_{23}
a_{31} a_{32} a_{33}


grid
grid object


tower, etc
grid
grid object


Game
Player
AI
Enemy
Tower
Owner
Grid
Grid object

- What kind of datatype should be on the grid?
  - Towers, enemies, castle?
  - Grid object?
  - Bananas??? Firetrucks???

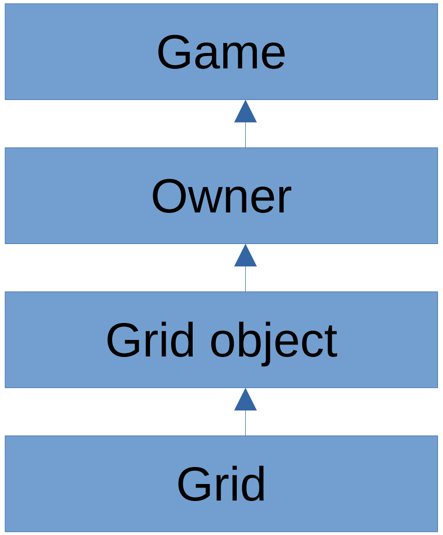
Templates!!

```
template <typename T>
class Grid {

    std::vector<T*> grid;
```

```
grid = Grid<grid_object>(rows, cols);
```

```
template<typename T>
void add_obj(int row, int col, Owner<GO>* owner, Grid<GO>* grid) {
        int index = owner->return_first_empty_index();
        T* ptr = new T(row, col, grid);
        this->grid(row, col) = ptr;
        owner->operator()(index) = ptr;
    }
}
```

```
   this->add_obj<Tower>(5, 3, &player, &grid);
```


Game
Owner
Grid object
Grid

- Using templates allows for more general and modular code
-