**Vorwort:**

Für unsere Aufgabe ist vor allem der Ordner *miniworld_core/miniworld/model/spatial* des bestehenden Projekts wichtig, da in diesem sowohl die schon vorhandenen MovementPattern liegen und hier die allgemeinen Ideen der Bewegungen und Definitionen von Straßen und Umgebungen einer Stadt liegen. Darüber hinaus sind noch der Ordner *miniworld_core/miniworld/management/spartial* (hier liegt der Movement Director) und die Klasse *miniworld_core/miniworld/model/collections/DistanceMatrix.py* (hier ist die Distanzmatrix definiert) wichtig.

**Wichtige Dateien im bestehenden Projekt, die noch nicht vollständig implementiert wurden:**

1. miniworld_core/miniworld/management/spartial/MovementDirectorArma.py
   - ➔ def get_geo_json_for_roads(self)
2. miniworld_core/miniworld/management/spartial/MovementDirectorCoreConfig.py
   - ➔ def get_geo_json_for_connections(self)
   - ➔ get_geo_json_for_nodes(self)
   - ➔ get_geo_json_for_roads(self)
3. miniworld_core/miniworld/management/spartial/MovementDirectorNoMobility.py
   - ➔ get_geo_json_for_nodes(self)
   - ➔ get_geo_json_for_roads(self)
4. miniworld_core/miniworld/model/collections/DistanceMatrix.py
   - ➔ def get_key(self, x, y)
   - ➔ def set_distance(self, x, y, distance)
   - ➔ def set_unlimited_distance(self, x, y)
   - ➔ def get_distance(self, x, y)
5. miniworld_core/miniworld/model/spatial/MovementPattern/AbstractMovementPattern.py
   - ➔ def get_next_map_node(self, crt_map_node, last_map_node)
   - ➔ def get_speed(self)
6. miniworld_core/miniworld/management/spatial/Node/AbstractNode.py
   - ➔ def get_name_of_movement_patter(self, movement_pattern)
   - ➔ def __check_conditions(self)
7. miniworld_core/miniworld/management/spatial/CoreConfigNodes.py
   - ➔ def get_list_of_nodes(self)
   - ➔ def get_node_for_node_id(self, node_id)
   - ➔ def get_coordinates(self)
   - ➔ def get_geo_json(self)
   - ➔ def _walk(self)
   - ➔ class CoreConfigNodesLan(CoreConfigNodes)
   - ➔ def get_distance_matrix(self)
   - ➔ class CoreConfigNodesWiFi(CoreConfigNodes)
   - ➔ def get_distance_matrix(self)
8. miniworld_core/miniworld/management/spatial/Nodes.py
   - ➔ def get_distance_matrix(self)
9. miniworld_core/miniworld/management/spatial/Singleton.py
   - ➔ class Singleton(Resetable)

**Schon vorhandene Movement Pattern:**

Insgesamt gibt es schon vier Movement Patterns. Zwei von Ihnen beziehen sich dabei auf die schon entwickelte OSM-Datenbank für die Stadt Marburg (miniworld_core/osm_marburg.db).

1. **RandomWalk:** Mithilfe der Datenbank osm_marburg.db bewegen sich die Knoten gemäß eines RandomWalk-Ansatzes zufällig durch die Straßen von Marburgs.
   ➔ **Fragen:**
   - Inwieweit ist das schon vorprogrammiert, dass die Knoten sich hier frei bewegen?
   - Können wir unser Movement Pattern auf diesem Ansatz erfolgreich aufbauen?
2. **MoveOnBigStreets:** An sich wie RandomWalk, mit dem Unterscheid, dass die Knoten große Straßen bevorzugen.
   ➔ **Fragen:** analog zu RandomWalk
3. **Arma3/Arma:** Basiert auf dem Spiel Arma3, wobei Koordinaten für jeden Spieler in Arma3 aus dem Spiel gezogen werden. Diese Koordinaten werden anschließend dem MovementDirector übergeben.
   ➔ **Fragen:** Können wir unser Movement Pattern auf diesem Ansatz erfolgreich aufbauen?
4. **CORE:** Dies unterstützt das UI von der ‚CORE emulation platform' um die Knoten auch graphisch zu setzen. Die Idee von diesem MP ist es mehrere ‚topology-files' haben zu können.
   ➔ **Fragen:**
   - Können wir unser Movement Pattern auf diesem Ansatz erfolgreich aufbauen?
   - Was bedeutet in diesem Fall mehrere Topologie-files haben zu können?
   - Ist mit einer Topolgie-file die Anordnung der Geräte in einem Netzwerk gemeint?
   - Patrick sollte, falls dieses MP wichtig oder sehr gut für uns geeignet ist, uns etwas mehr über die Vorgehensweise dieses MP erzählen.
5. Allgemeine Fragen zu den Movement Pattern:
   - Wofür sind die Klassen AbstractMovementPattern.py, ReplayMovementPattern.py und ReplayNode.py? (miniworld_core/miniworld/model/spatial/MovementPattern)

Darüber hinaus gibt es für jeden MovementPattern eine eigene Node-Klasse (miniworld_core/miniworld/model/spatial/Node).

**Ideen:**

- Man kann sich auf OpenSource-Geoinformationssyteme-Software beziehen (GIS), wie zum Beispiel GRASS GIS oder QGIS (https://de.wikipedia.org/wiki/Geoinformationssystem). Die Frage ist, ob diese mit unserem Projekt kompatibel ist, da diese GIS-Software oft eine eigene Art von Distanzmatrix besitzt und somit vielleicht zu viel durcheinander bringen würde, was eigentlich schon funktioniert.
- Weitere GIS-Software ist in Unterpunkt ‚Geoinformationssysteme' auf https://de.wikipedia.org/wiki/Spatial zu finden.
- Bis jetzt habe ich mir drei OpenSource-Games angeschaut. Ich habe zum Beispiel ein zu GTA-ähnliches Spiel gefunden, welches gut eine Art automatischen RandomWalk liefern könnte, womit wir gut die Tagesablaufvariante wiedergeben könnten (https://github.com/rwengine/openrw). Falls wir uns für das Katastrophenszenario entscheiden, würde sich eher SimCity anbieten, da es in diesem Spiel eben um das Verhalten von Knoten bei Katastrophen geht. Ich bin bis jetzt jedoch noch nicht dazu gekommen, mir die einzelnen OpenSource-Games genauer anzuschauen.