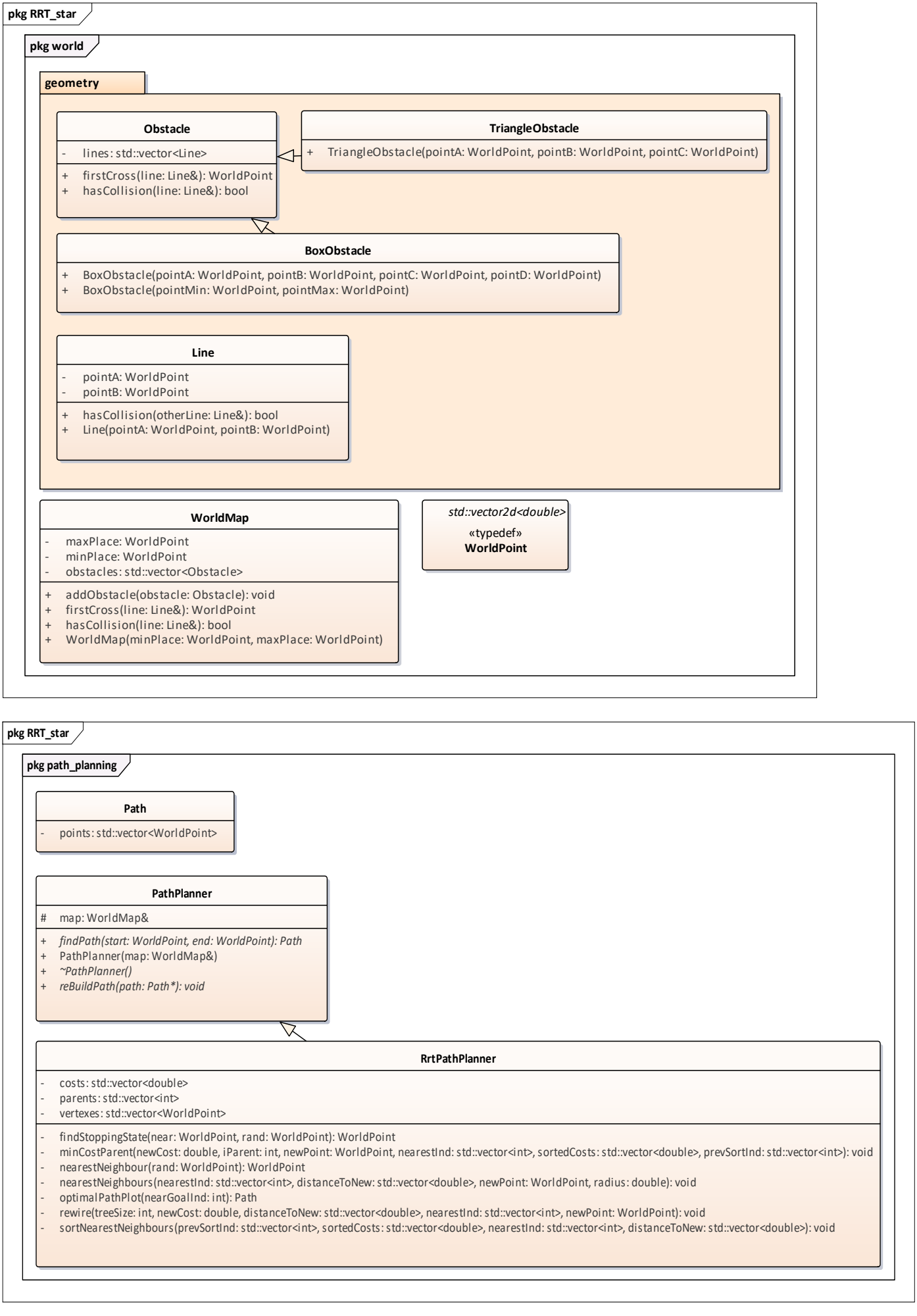
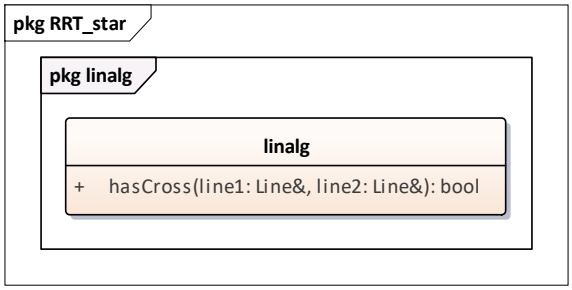
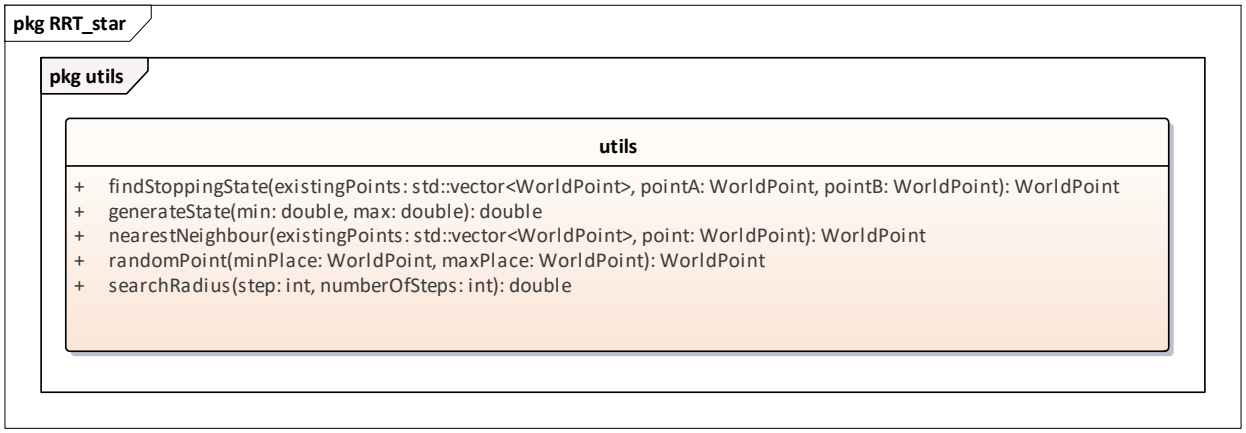

pkg RRT_star
pkg world
geometry
Obstacle
- lines: std::vector<Line>
+ firstCross(line: Line&): WorldPoint
+ hasCollision(line: Line&): bool
TriangleObstacle
+ TriangleObstacle(pointA: WorldPoint, pointB: WorldPoint, pointC: WorldPoint)
BoxObstacle
+ BoxObstacle(pointA: WorldPoint, pointB: WorldPoint, pointC: WorldPoint, pointD: WorldPoint)
+ BoxObstacle(pointMin: WorldPoint, pointMax: WorldPoint)
Line
- pointA: WorldPoint
- pointB: WorldPoint
+ hasCollision(otherLine: Line&): bool
+ Line(pointA: WorldPoint, pointB: WorldPoint)
WorldMap
- maxPlace: WorldPoint
- minPlace: WorldPoint
- obstacles: std::vector<Obstacle>
+ addObstacle(obstacle: Obstacle): void
+ firstCross(line: Line&): WorldPoint
+ hasCollision(line: Line&): bool
+ WorldMap(minPlace: WorldPoint, maxPlace: WorldPoint)
std::vector2d<double>
«typedef»
WorldPoint

pkg RRT_star
pkg path_planning
Path
- points: std::vector<WorldPoint>
PathPlanner
map: WorldMap&
+ findPath(start: WorldPoint, end: WorldPoint): Path
+ PathPlanner(map: WorldMap&)
+ ~PathPlanner()
+ reBuildPath(path: Path*): void
RrtPathPlanner
- costs: std::vector<double>
- parents: std::vector<int>
- vertexes: std::vector<WorldPoint>
- findStoppingState(near: WorldPoint, rand: WorldPoint): WorldPoint
- minCostParent(newCost: double, iParent: int, newPoint: WorldPoint, nearestInd: std::vector<int>, sortedCosts: std::vector<double>, prevSortInd: std::vector<int>): void
- nearestNeighbour(rand: WorldPoint): WorldPoint
- nearestNeighbours(nearestInd: std::vector<int>, distanceToNew: std::vector<double>, newPoint: WorldPoint, radius: double): void
- optimalPathPlot(nearGoalInd: int): Path
- rewire(treeSize: int, newCost: double, distanceToNew: std::vector<double>, nearestInd: std::vector<int>, newPoint: WorldPoint): void
- sortNearestNeighbours(prevSortInd: std::vector<int>, sortedCosts: std::vector<double>, nearestInd: std::vector<int>, distanceToNew: std::vector<double>): void


pkg RRT_star
pkg linalg
linalg
+ hasCross(line1: Line&, line2: Line&): bool


pkg RRT_star
pkg utils
utils
+ findStoppingState(existingPoints: std::vector<WorldPoint>, pointA: WorldPoint, pointB: WorldPoint): WorldPoint
+ generateState(min: double, max: double): double
+ nearestNeighbour(existingPoints: std::vector<WorldPoint>, point: WorldPoint): WorldPoint
+ randomPoint(minPlace: WorldPoint, maxPlace: WorldPoint): WorldPoint
+ searchRadius(step: int, numberOfSteps: int): double