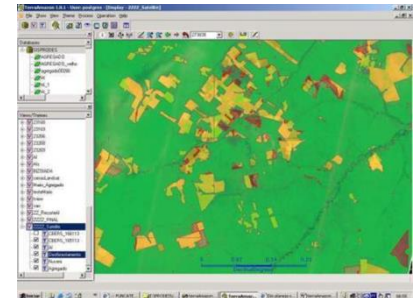
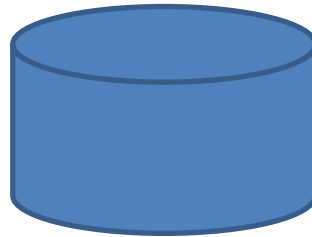
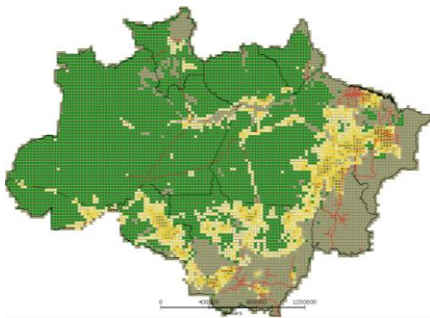


Spatial Databases: Lecture 7+8

Institute for Geoinformatics
Winter Semester 2014



Malumbo Chipofya: room 109

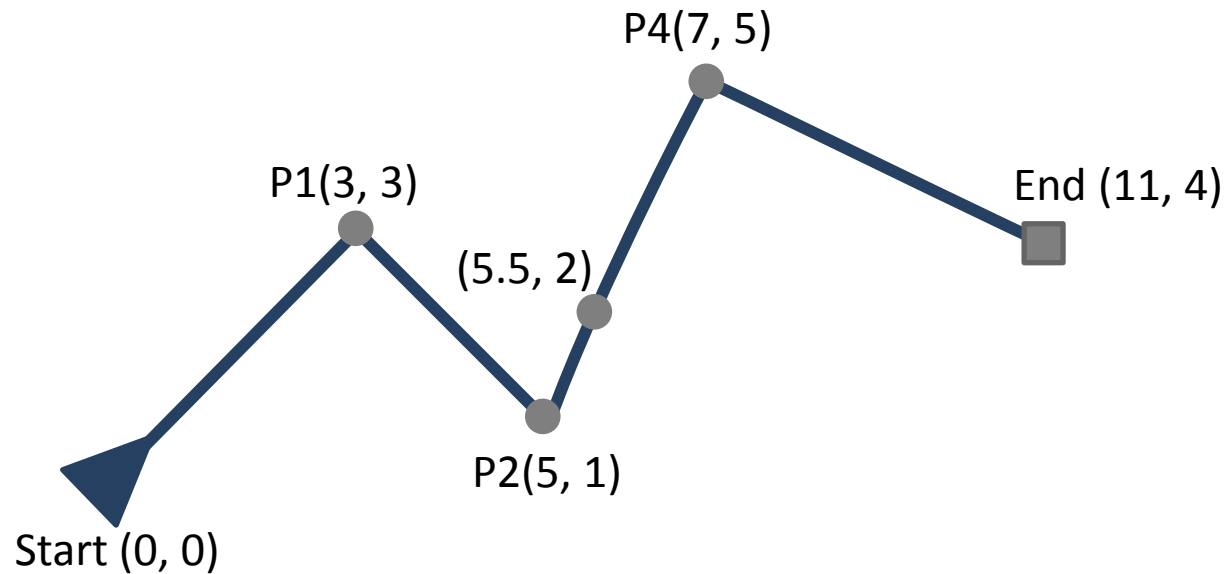
7 Exercises and Discussion

- Spatial relationships
- Spatial joins
- Projections and the Geography type
- Geometries from Geometries
- A note on validity
- Equality

8 Linear Referencing & pgRouting

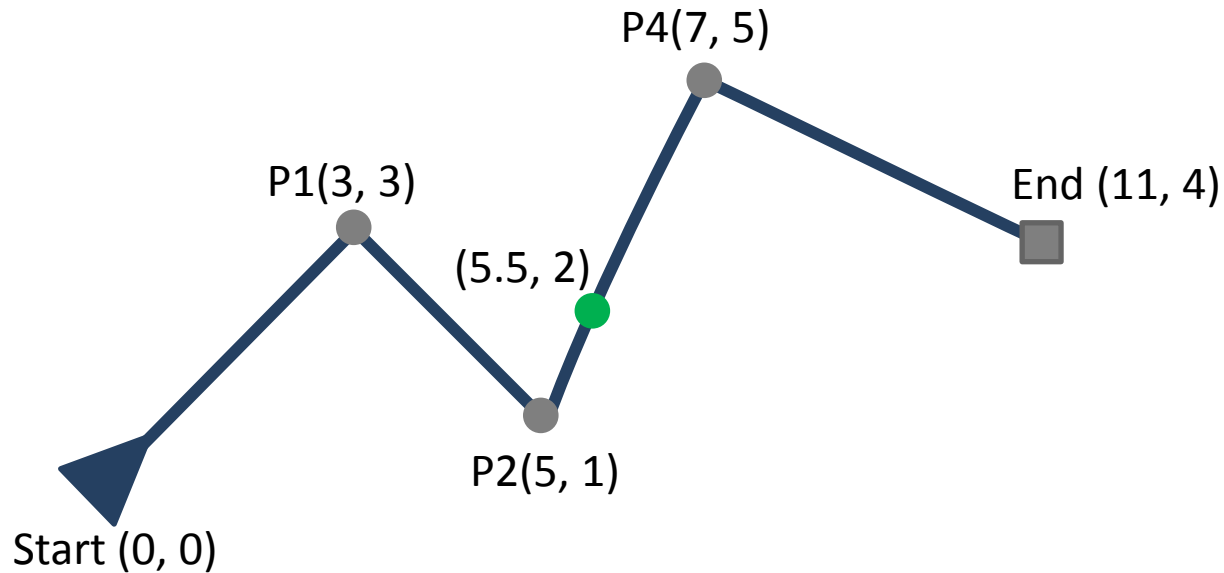
Linear Referencing

- Locate a point along a line



Linear Referencing

```
SELECT ST_LineLocatePoint(  
    'LINESTRING(0 0, 3 3, 5 1, 7 5, 11 4)',  
    'POINT(5.5 2)'  
);
```

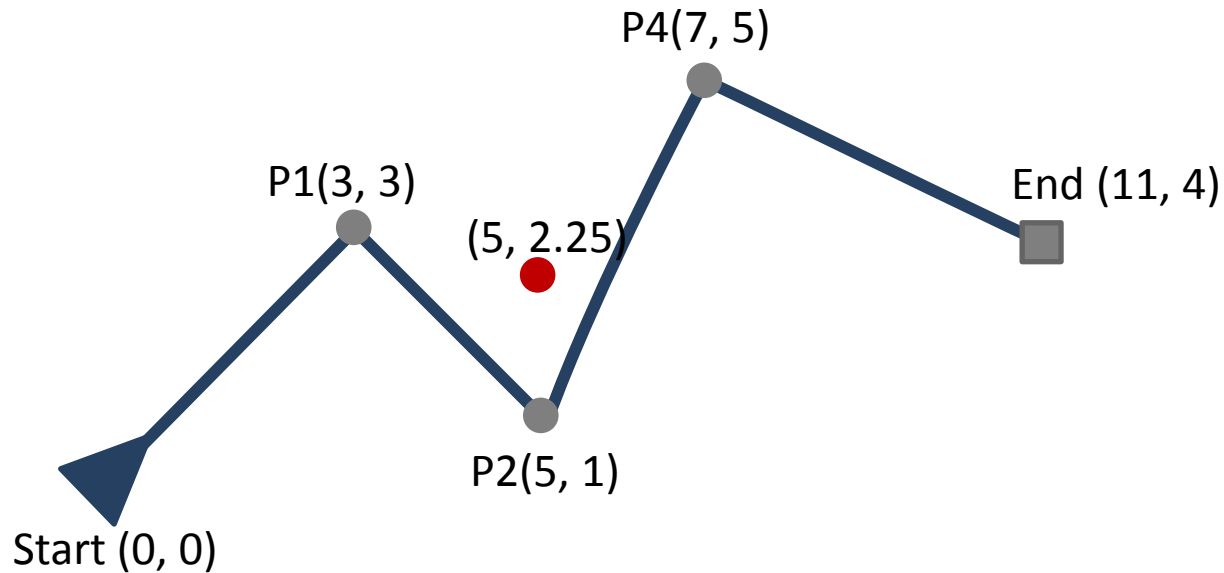


Linear Referencing

```
SELECT ST_LineLocatePoint(  
    'LINESTRING(0 0, 3 3, 5 1, 7 5, 11 4)',  
    'POINT(5 2.25)'  
);
```

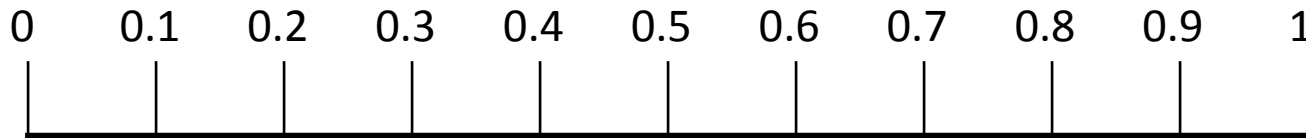
Linear Referencing

```
SELECT ST_LineLocatePoint(  
    'LINESTRING(0 0, 3 3, 5 1, 7 5, 11 4)',  
    'POINT(5 2.25)'  
);
```

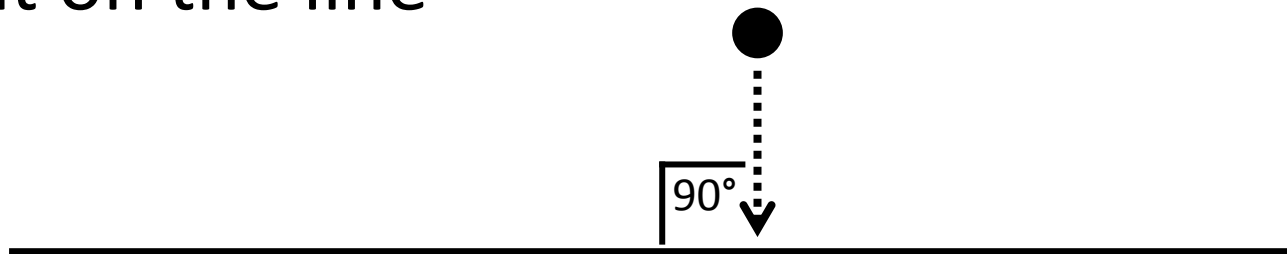


Linear Referencing

- `ST_LineLocatePoint()` returns the fraction of a line traversed from start to the located point



- Points not on line are projected to the nearest point on the line



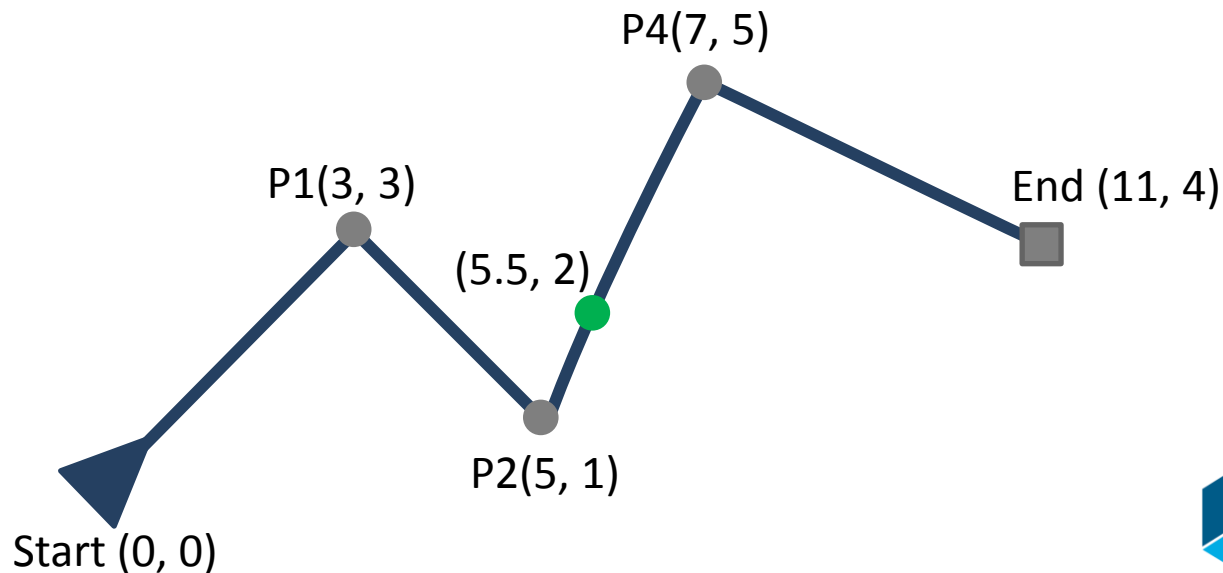
Linear Referencing

```
SELECT ST_AsText(ST_LineInterpolatePoint(  
    'LINESTRING(0 0, 3 3, 5 1, 7 5, 11 4)',  
    0.522720546074802  
));
```

ST_LineInterpolatePoint() returns the point at the specified location along a line

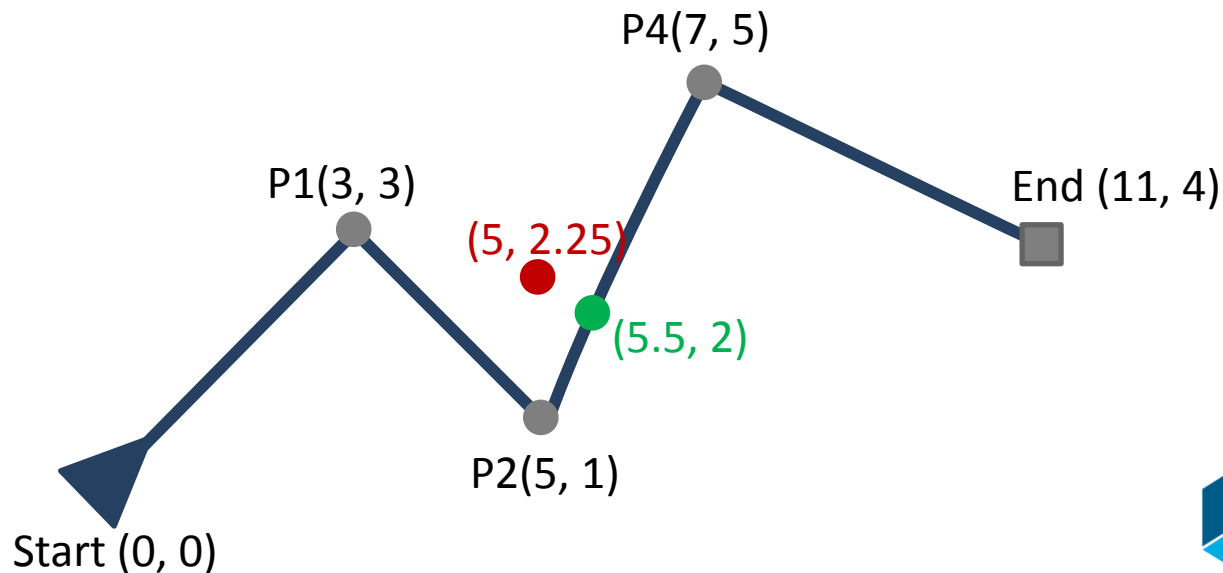
Linear Referencing

```
SELECT ST_AsText(ST_LineInterpolatePoint(  
    'LINESTRING(0 0, 3 3, 5 1, 7 5, 11 4)',  
    0.522720546074802  
));
```



Linear Referencing

```
SELECT ST_AsText(ST_LineInterpolatePoint(  
  'LINESTRING(0 0, 3 3, 5 1, 7 5, 11 4)',  
  ST_LineLocatePoint(  
    'LINESTRING(0 0, 3 3, 5 1, 7 5, 11 4)',  
    'POINT(5 2.25)'  
  )  
));
```



pgRouting

- A postgres/postgis extension for computing routes and routing information
 - To use pgRouting you must first enable it for your database

```
CREATE EXTENSION pgRouting;
```

pgRouting

- Let's take a section of the nyc_streets database

```
SELECT st.* INTO si_streets FROM
```

```
    nyc_streets st,
```

```
    nyc_neighborhoods nb
```

```
WHERE
```

```
    ST_Intersects(st.geom, nb.geom) AND
```

```
    nb.boroname = 'Staten Island';
```

pgRouting

- To do routing we need a network topology:
creating a topology.
 1. Identify start and end points of your streets
 2. Let pgRouting create the topology for you

```
ALTER si_streets ADD COLUMN "source" integer;
```

```
ALTER si_streets ADD COLUMN "target" integer;
```

```
SELECT pgr_createTopology(' si_streets', 0.00001,  
'geom', 'gid');
```

pgRouting

- If not already created you can add indexes for your two new columns (check now)

```
CREATE INDEX si_streets_source_idx ON  
si_streets("source");
```

```
CREATE INDEX si_streets_target_idx ON  
si_streets("target");
```

- Plus we must declare a cost column

```
ALTER si_streets ADD COLUMN "cost" real;
```

pgRouting

- You are now ready to route!
- A story
 - An angry person shot someone dead in Staten Island and fled by car. The event is recorded with id 1408 in the homicides table. In a haste the same person apparently run over another person and killed her as well (recorded with id 2388). The question is: assuming the person took the shortest escape route between the first and second events, which route was it?

pgRouting

- Dijkstra

pgRouting

- An update to the story
 - As we dig for more evidence we encounter a witness who says he saw the getaway car between the times of the two events at Huguenot subway station (id 484). Does this fact change the most probable route?

pgRouting

- kDijkstra

Exercise

- Follow the tutorial at

http://workshops.boundlessgeo.com/postgis-intro/linear_referencing.html

- Outcome: linear referencing of subway stations

Exercise

- Split all street linestrings containing references to subway stations into parts.
 - From the start to the first subway station
 - From each subway station to the next subway station or end of linestring whichever comes first
- Create a new streets table where any split linestrings are replaced by their pieces.
- Make the new streets table ready for pgRouting

Exercise

- Compute the station to station distance for all stations on a given route assuming that consecutive stations are spatially nearest to each other (break ties arbitrarily)
- Create a new table of subway routes with a line string for each route
- Make the new subway routes table ready for pgRouting

Exercise

- For the next tasks assume that the cost of travel by road is the distance and consider three cost scenarios for travel by subway
 - Straight line distance between consecutive stations
 - 0.5 times the straight line distance
 - 2 times the straight line distance

Exercise

- Use Shortest Path Dijkstra to find the shortest routes for the following journeys using a combination of travel modes by car and subway
 - P1 to P2
 - P1 to P3
 - P2 to P4
 - P2 to P3
- Repeat the task above but this time avoid subway stations within 100 meters of a homicide

Exercise

- Repeat the two tasks but this time use the A^* variant of shortest path
- Comment on the differences between the routes under the different cost assumptions and restrictions

References

- <http://postgis.net/docs/>
- <http://workshops.boundlessgeo.com/postgis-intro/index.html>
- <http://pgrouting.org/documentation.html>
- <http://workshop.pgrouting.org/>
 - See especially chapters 5, 7, and 8

That's all for today

Thank you!

Questions?