

AN AGENT-BASED
APPROACH
TO THE SIMULATION OF
PEDESTRIAN MOVEMENT
AND FACTORS THAT
CONTROL IT

1. Why another model?

Planned as part of a modular model able to simulate rent rate / land value / land use changes in informal shopping centres.

Why now?

Because urban agenda is constantly shifting towards old city centres (see Urban Task Force reports)

We need to know the effects of intervention on urban environment

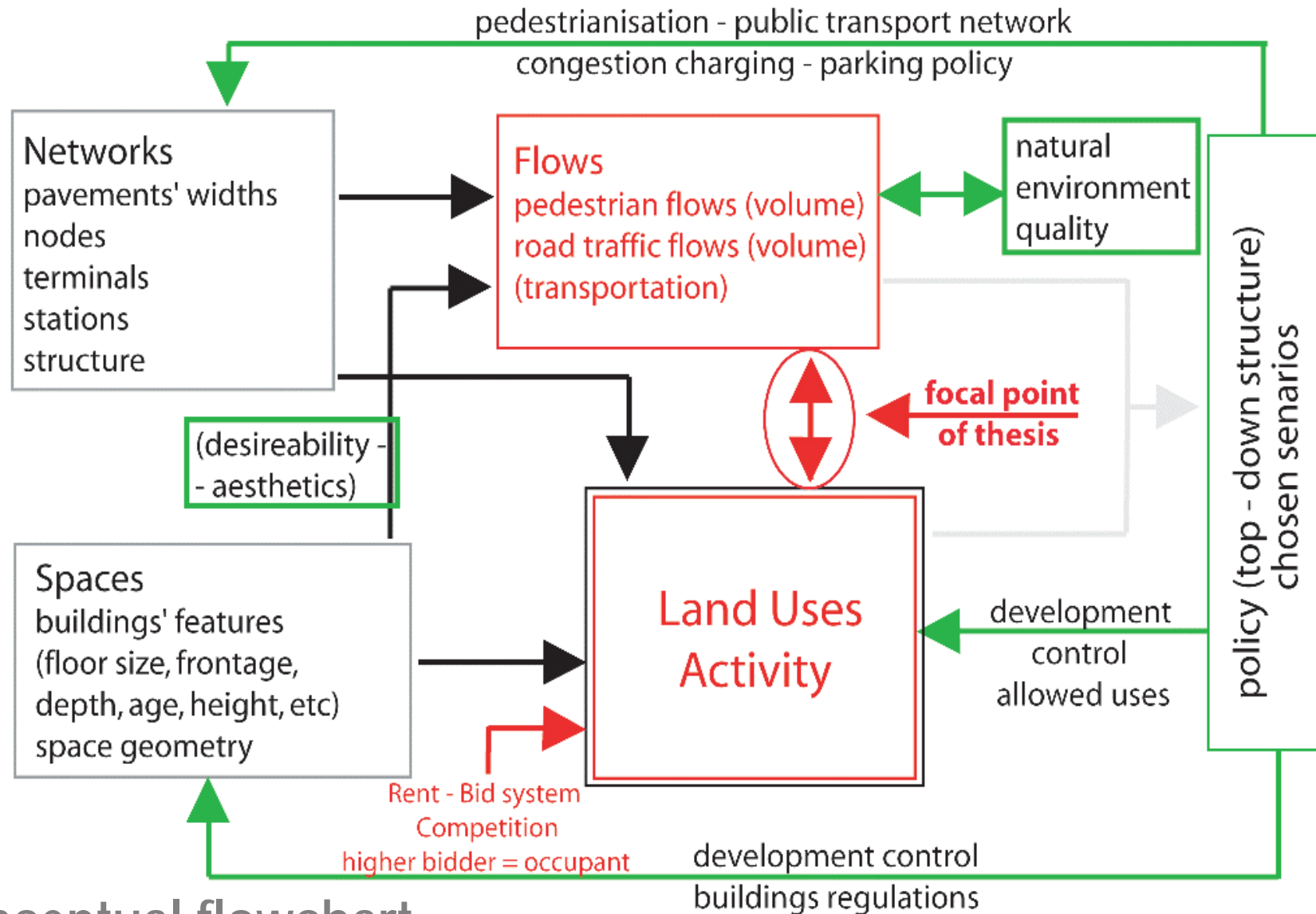
Modular model focuses in microscale land use change due to decisions that are likely to have an impact on trips' demand and/or offer

Microscale...

1. Discrete units (buildings, activities/uses)
2. Observable network morphology

Why microscale?

- a. Unusually high cost of land in shopping centres - precision
- b. Urban centres complex – lack of uniformity
- c. Importance of location/sensitivity of retailing uses to location



Conceptual flowchart

2. Why this model?

Focal Point: pedestrians ↔ retailing uses interface

- a. Why pedestrian movement more important in this case?

- b. Why the model should adopt an agent-based approach?

Walking vs. other transport modes

a. Pedestrians affect rent rates

pedestrian flows \uparrow then rent rates \uparrow Retailing theory (Brown 1992)

i. indirect promotion / increased awareness of units location

not as important in motor transport – duration of stimulation significantly shorter

ii. possibility to engage in shopping activity

not likely while using other transport modes

b. Private motor traffic low or restricted in shopping centres

c. Public transport has minimal negative effects and positive effects are not independent (increased accessibility)

Why an agent-based approach?

- a. **Bottom up systems** individual choices shape outcome

Emergence...

- b. **Microscale**

Aggregates not so effective – they lose ability to predict urban phenomena when microlevel activities are involved

- c. **Small numbers**

Occasionally, numbers of involved actors too low or small number of actors can have asymmetrical impact on system

Inadequacy of statistic methods

Emergent Phenomena

- a. communication – interaction
- b. “predictable” individual behaviour (rule based vs. erratic)
- c. lack of hierarchy
- d. Inclination to collectivistic behaviour

Complex systems → Complexity theory

Emerging outcome cannot be predicted by examination of individual cases. Reductionism insufficient

3. Model strategy

A. Inputs

- i. Built form
- ii. Walkable environment (future classification - walkability)
- iii. Land use map (points representing entrances)
- iv. Street network (street segments and nodes)
- v. Gates
(entrances to study area – tube stations, bus stops, boundaries)
- vi. Visibility map (produced from dense set of points)

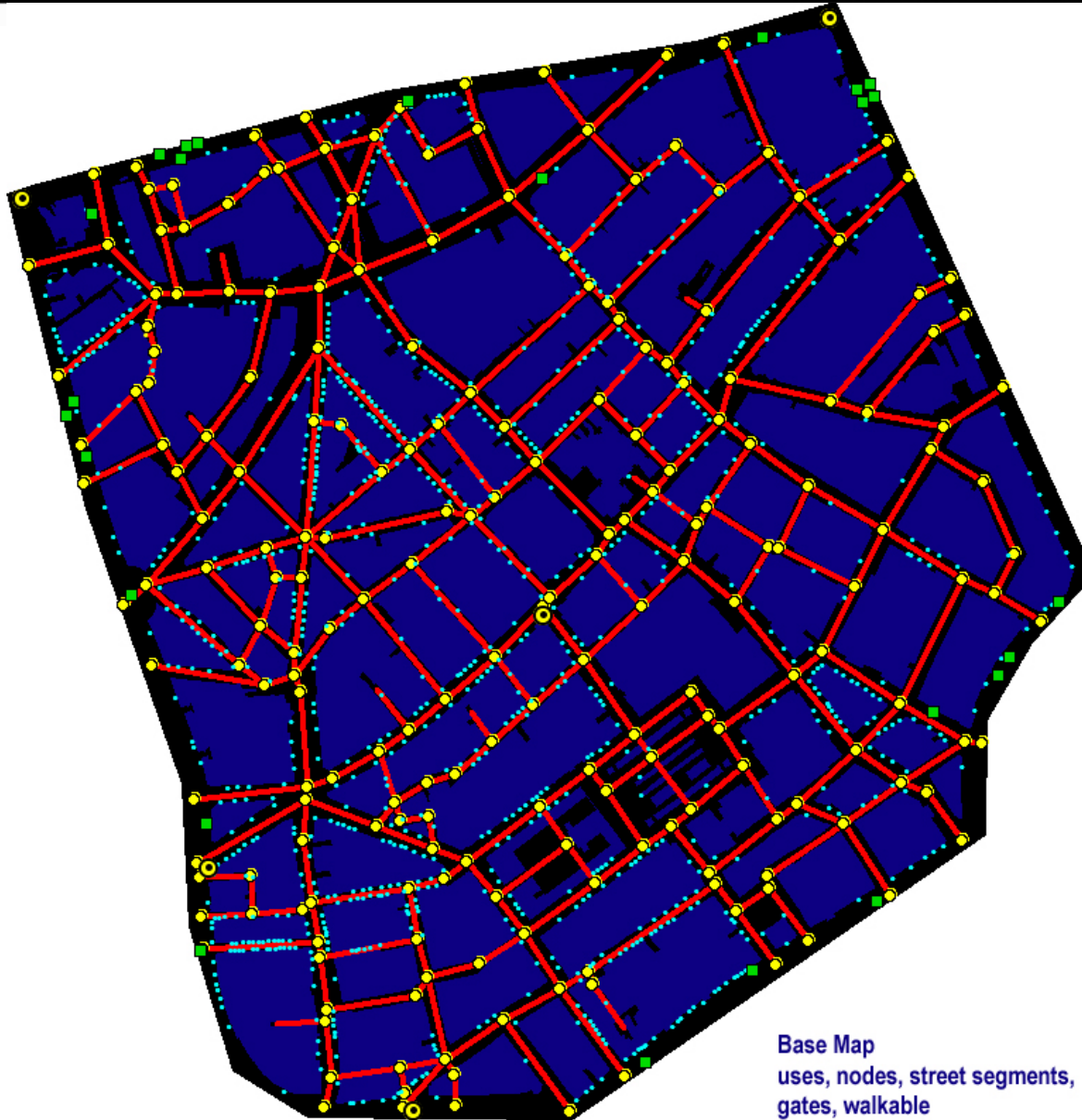
Agent Based Models at CASA

Thursday 19th January 2006



UCL

study area



Base Map
uses, nodes, street segments,
gates, walkable



STUDY AREA
Covent Garden

B. Topology and Representation

- i. Topologic awareness and vector representation (No ESRI options - JAVA)
 - a. OpenMap Java library used both for topologic calculations and for representations – lat/lon structure is a nightmare
 - b. OpenMap used for representation and JTS 1.4 topology suite used for calculations – still conversion to lat/lon will be needed for representation
 - c. Code a representation tool by yourself or find one...
- ii. Topologic awareness and vector representation (ESRI options JAVA)

Use JTS 1.4 for calculations and Agent Analyst for representation. Agent Analyst is a tool developed by the Repast team, to refresh the ArcGIS 9.x screen after every Repast step.

B. Topology and Representation

iii. Topologic awareness and vector representation (ESRI option VisualStudio)

JTS1.4 topologic suite - Vivid Solutions is now also available as GeoTools.NET for Visual Studio (C# and Visual-Basic) including Geotools functionality

However Repast 3.0 .NET not yet supporting GIS (no anl.repast libraries) and no C# interface for Agent Analyst.

You can write your own classes for GIS functionality in Repast.NET (very easy) choosing any ESRI (MapObjects, ArcEngine etc) or other available .NET vector representation tool.

For topologic calculations use either Geotools.NET or any ESRI product.

C. Data Handling

i. GIS for data handling

GeoTools 2.0 – Java suite compatible to ESRI and MapInfo software

OpenMap libraries – Java

In case of Visual studio .NET choice, use Geotools.Net or any ESRI product

ii. Using Matrices and text files

This solution may be faster but in complex systems can be hard to follow



D. Vision

- i. Network and Uses awareness – Useful for mid-scale way-finding

Implementation of direct vision algorithms to agents extremely power consuming.

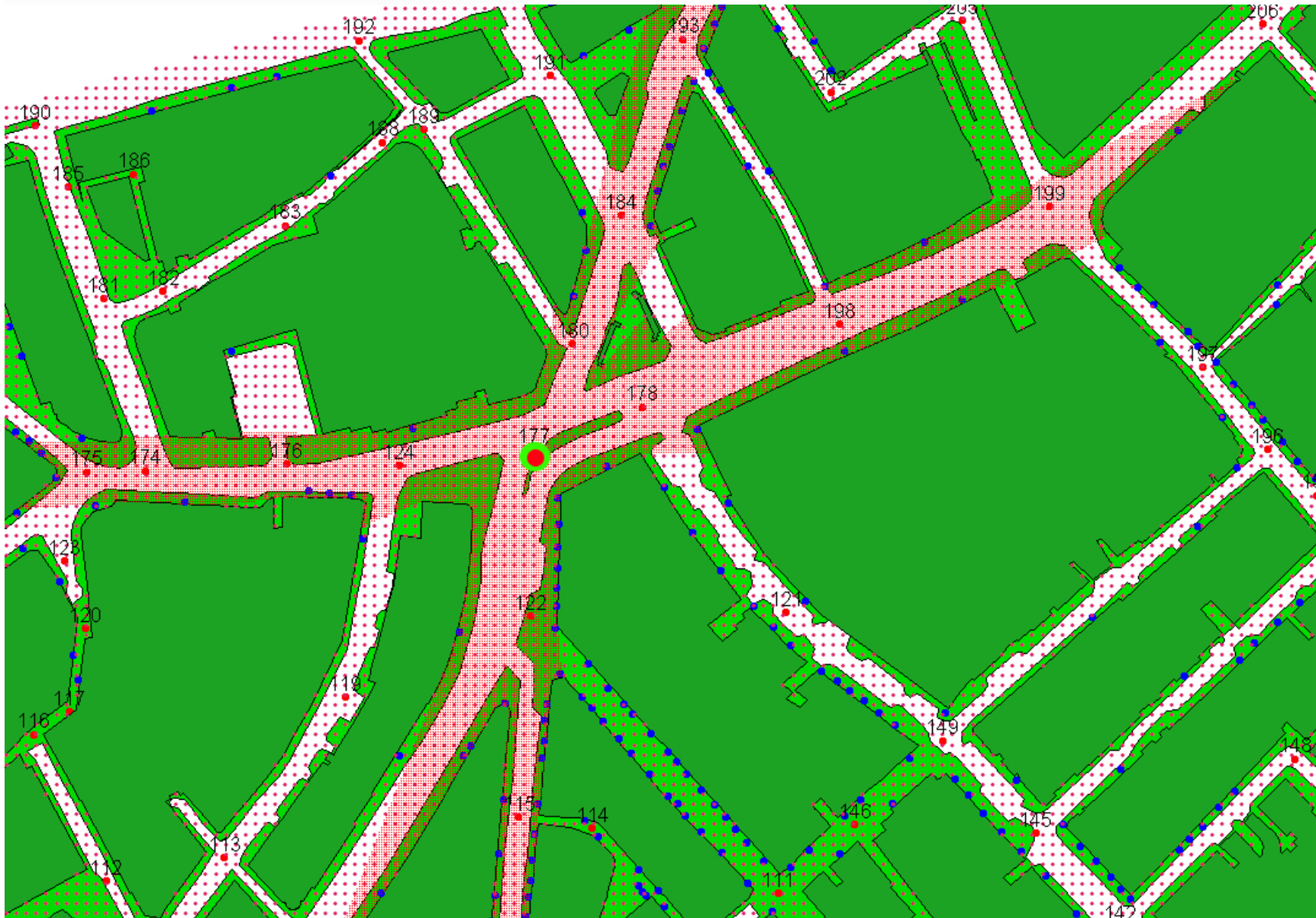
INDIRECT VISION: calculation of isovists for every node and use.

Exosomatic visual architecture...

“that is, a paradigm of vision that holds outside the body, in the environment” (Alasdair Turner).

Agent Based Models at CASA

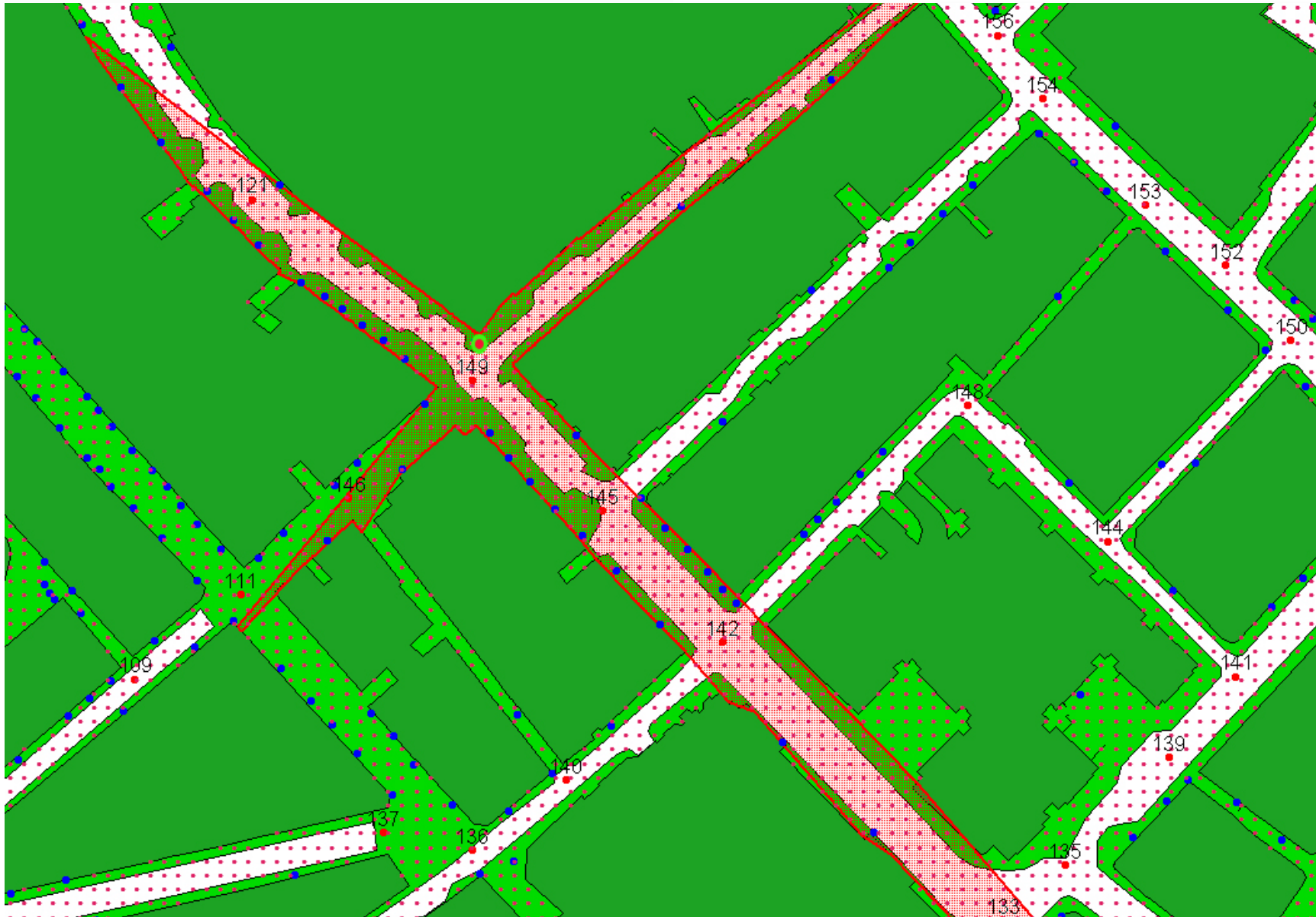
Thursday 19th January 2006



Vassilis Zachariadis - An Agent-Based Approach to the Simulation of Pedestrian Movement

Agent Based Models at CASA

Thursday 19th January 2006



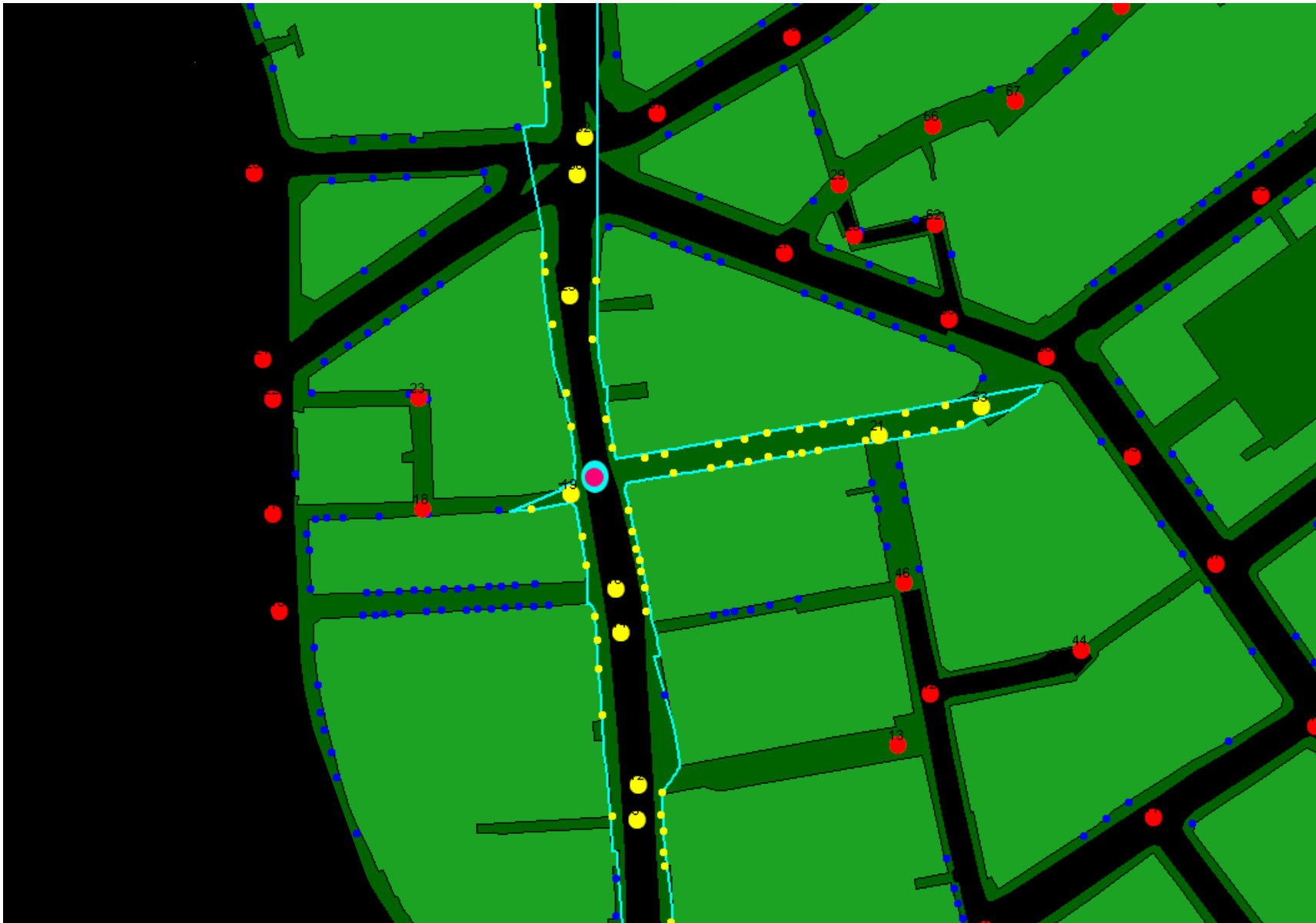
Vassilis Zachariadis - An Agent-Based Approach to the Simulation of Pedestrian Movement

Agent Based Models at CASA

Thursday 19th January 2006



UCL



Vassilis Zachariadis - An Agent-Based Approach to the Simulation of Pedestrian Movement

E. Scheduler – target destinations

- i. Random set of goals/destinations – predefined
- ii. Predefined – Based on set of criteria

distance (real distance) from entry point or previous position, floor-space, type of activity, design etc.

In model : floor-space / distance

- iii. Not Predefined goals ➔ visual stimulation (floor-space/type/design)

Thresholds and Memory

- a. accuracy / pool of choices
- b. dropability of targets during predefined schedules

If attractiveness > threshold ➔ Engage ➔ Reform schedule

F. Way Finding

Algorithm

step 1 : check position

step 2 : find closer node

step 3 : find node closer to target

step 4 : load appropriate *route finder* (e.g. *shortest path algorithms* (Dijkstra's), *utility maximising paths* etc.)

Route Finder



$\forall i \in \text{nodes} \ \& \ i_value1 = \text{value1}$

find $k \in \text{nodes}, k \in \text{isovist_i}, k \neq i \ \& \ i_value1 = -1$

give $i_value1 = \text{value1} + 1$

$\text{value1} = \text{value1} + 1$

Simpler route, minimum distance (used in this version of the model),

composite cost : (distance, visibility, retailing units/active frontage, traffic (exp.), walkability etc.)

Choice depends on profile/familiarity with network

```
//viewpoint developer
```

```
pw = new PrintWriter(new FileOutputStream(tempFile12), true);
pw1 = new PrintWriter(new FileOutputStream(tempFile22), true);

for( ip=0; ip<4182; ip++) {

    isov = fisov[ip].getDefaultGeometry();

    for (kkp=0; kkp<4182; kkp++){

        if ((nodes[kkp].within(isov) == true)){

            distancedir = (int) (nodes[ip].distance(nodes[kkp]));
            pw.print(distancedir + " ");
            pw1.print(1 + " ");
        }

        else{
            pw.print(-1 + " ");
            pw1.print(0 + " ");
        }

    }

    pw.println();
    pw1.println();
}
```

This algorithm constructs two matrices. One representing i/o links between nodes and the other strength of links (in this case distance)

```
//Path developer
```

```
//compute path complexity for every node (1 to 1782) in the study area
for (kki=0; kki<1782; kki++){
```

```
//build new attributes "route" and "distance" for nodes through nodesadapter and add complexity val = 1 and valdist = 0 for kki
nodesadapter[kki].addAttribute("route", val); // val = 1
nodesadapter[kki].addAttribute("distance", valdist); // valdist = 0
```

```
while (m<4){ // "4" should be a complexity depth greater than that of the given network
```

```
    //see if node kk has the appropriate value to start computation
    for (kk=0; kk<1782; kk++){
        nint=-1;
        o = nodesadapter[kk].getAttribute("route");
        odist = nodesadapter[kk].getAttribute("distance"); // odist informs whether node has been given a value
```

```
        if (odist != null){
            testdist = Double.parseDouble(odist.toString()); //if it has been given a value - get it!
```

```
            if ((int)testdist != -1){ //check it is not the null value
                Coordinate cdist = new Coordinate(nodesadapter[kk].getX(), nodesadapter[kk].getY());
                GeometryFactory factorydist = new GeometryFactory();
                pointdist = factorydist.createPoint(cdist); // create a point for the given node (kk) coordinates
```

```
                for (ik=0; ik<1782; ik++){ // for any other node ik
                    if ((nodes[ik].within(isov[kk]) == true)){ //that is within sight
```

```
                        odist2 = nodesadapter[ik].getAttribute("distance"); //check whether
                        findist = testdist + nodes[ik].distance(pointdist);
```

```
                        if (odist2 != null){ // it has been given a value yet, and if it has
                            findist2 = Double.parseDouble(odist2.toString());
                            if (findist2 > findist || (int)findist2 == -1){ // check if it is greater than the distance of kk to ik +kk's value
                                findist2 = findist;
                                valdist = new Double(findist2); // if yes, set its new value to kk's + their distance
                                nodesadapter[ik].addAttribute("distance", valdist);
                            }
                        }
```

```
                    }
                    else{ // if node ik has not value yet
                        valdist = new Double(findist);
                        nodesadapter[ik].addAttribute("distance", valdist); // set its val to kk's + their distance
```

This algorithm constructs two matrices. One containing shortest paths between any nodes based on route complexity between them and the other containing shortest paths based on route distance

G. On the fly route changes (1)

Important!!! It enables emergence through interaction and explorative behaviours for agents with no set schedule like tourists, by-passers and leisure shoppers

A check is needed every time an agent is at a node or every n steps

1. Exploration

Agent calculates the attractiveness of each street segment (possible choice)

a. Shop entrances / length or floor-space / length

b. Visibility

c. Traffic flows – Pedestrian flows

too low safety - too high discomfort and speed

but also indication of importance  mimic behaviour

d. Already visited etc.

3. Model strategy

G. On the fly route changes (2)

2. Agent interaction – Agents with set targets

At each node calculate cost of travel to target through each choice (street seg.)

Cost = cost of seg. + cost from there on (from *route finder*)

Cost of seg. → Traffic flows

In cases (depends on profile):

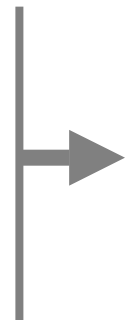
visibility, shops, already visited etc.



4. Scenario

Opera

Flows scaled on data from Intelligent Design Partnership

1. Run with 500 actors – no event
 2. Run with 500 actors – just event
 3. Run with 500 + 500 actors – event
- for 30 minutes each
- 
- A vertical line is positioned to the right of the three numbered scenarios. A horizontal arrow points from the middle of this vertical line to the text 'for 30 minutes each'.

Agents for event entering area following reverse Poisson distribution.

→ Check changes at : a. density, b. flows

Check difference between cases [(1) + (2)] and (3)

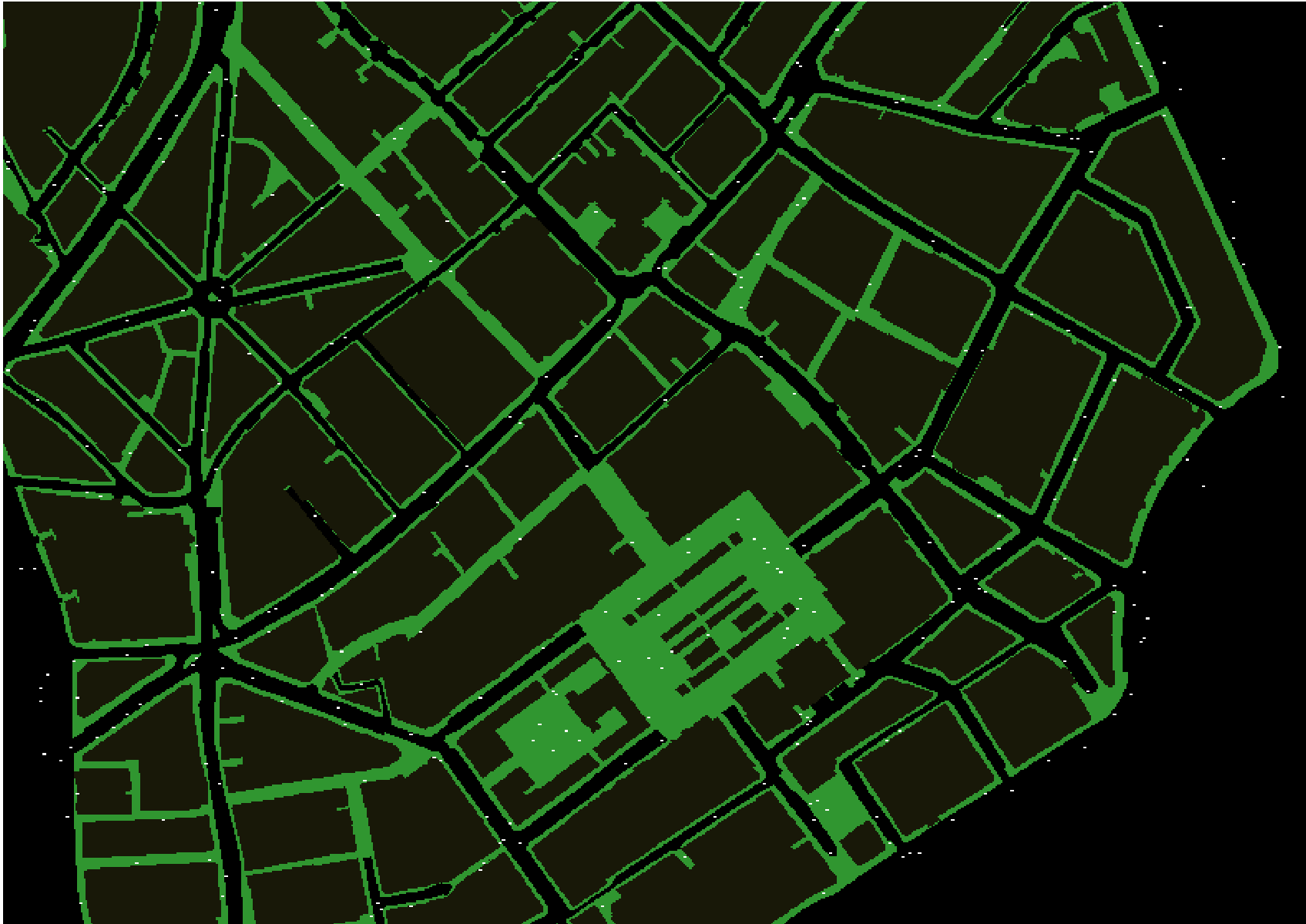
Agent Based Models at CASA

Thursday 19th January 2006



UCL

Case (1)



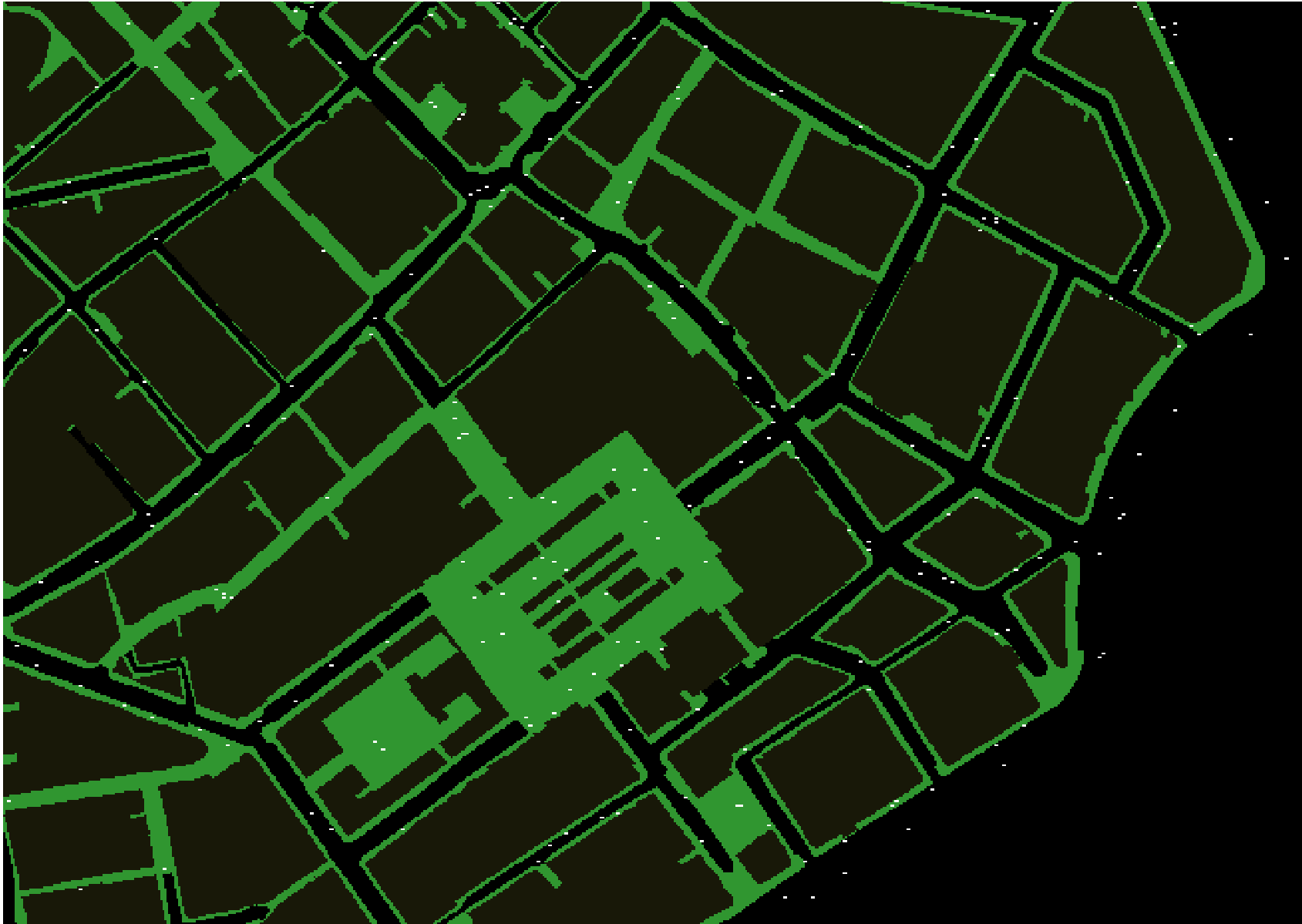
Vassilis Zachariadis - An Agent-Based Approach to the Simulation of Pedestrian Movement

Agent Based Models at CASA

Thursday 19th January 2006



UCL



Case (2)

Vassilis Zachariadis - An Agent-Based Approach to the Simulation of Pedestrian Movement

Agent Based Models at CASA

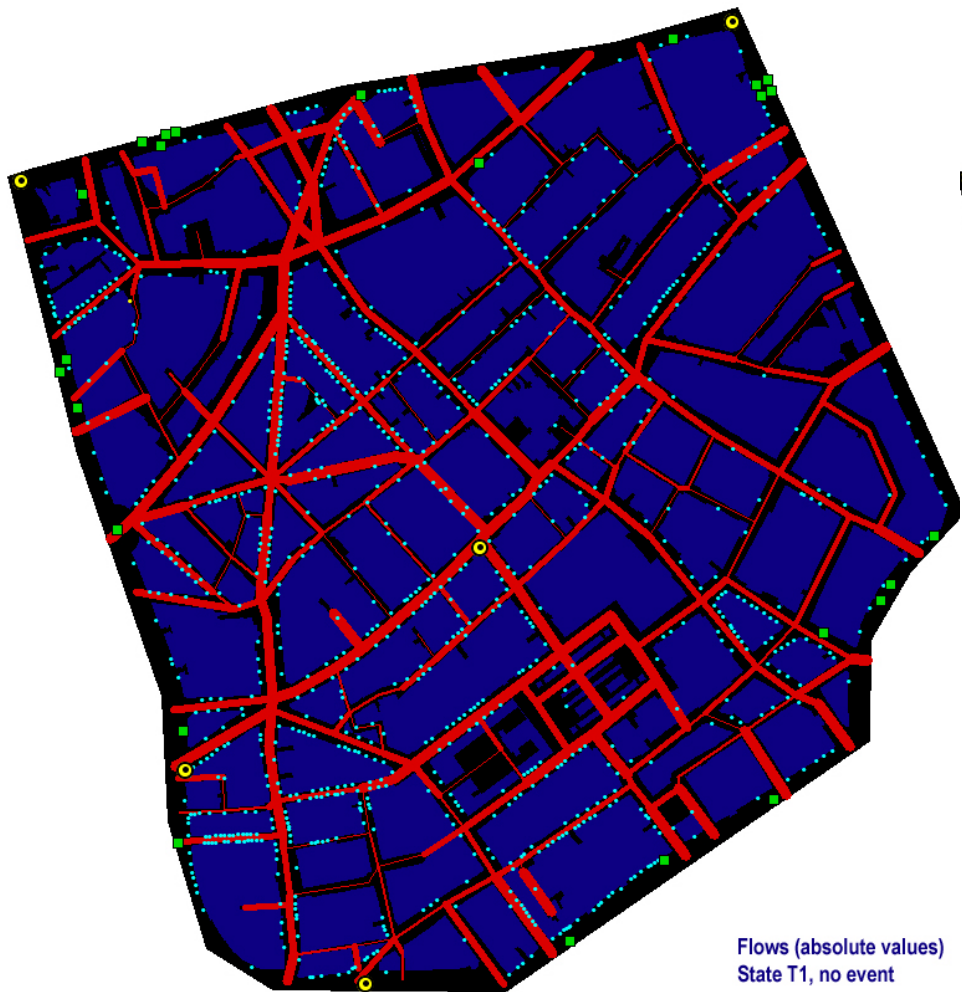
Thursday 19th January 2006



UCL

4. Scenario





Flows (absolute values)
State T1, no event



Flows (absolute values)
State T2, with event

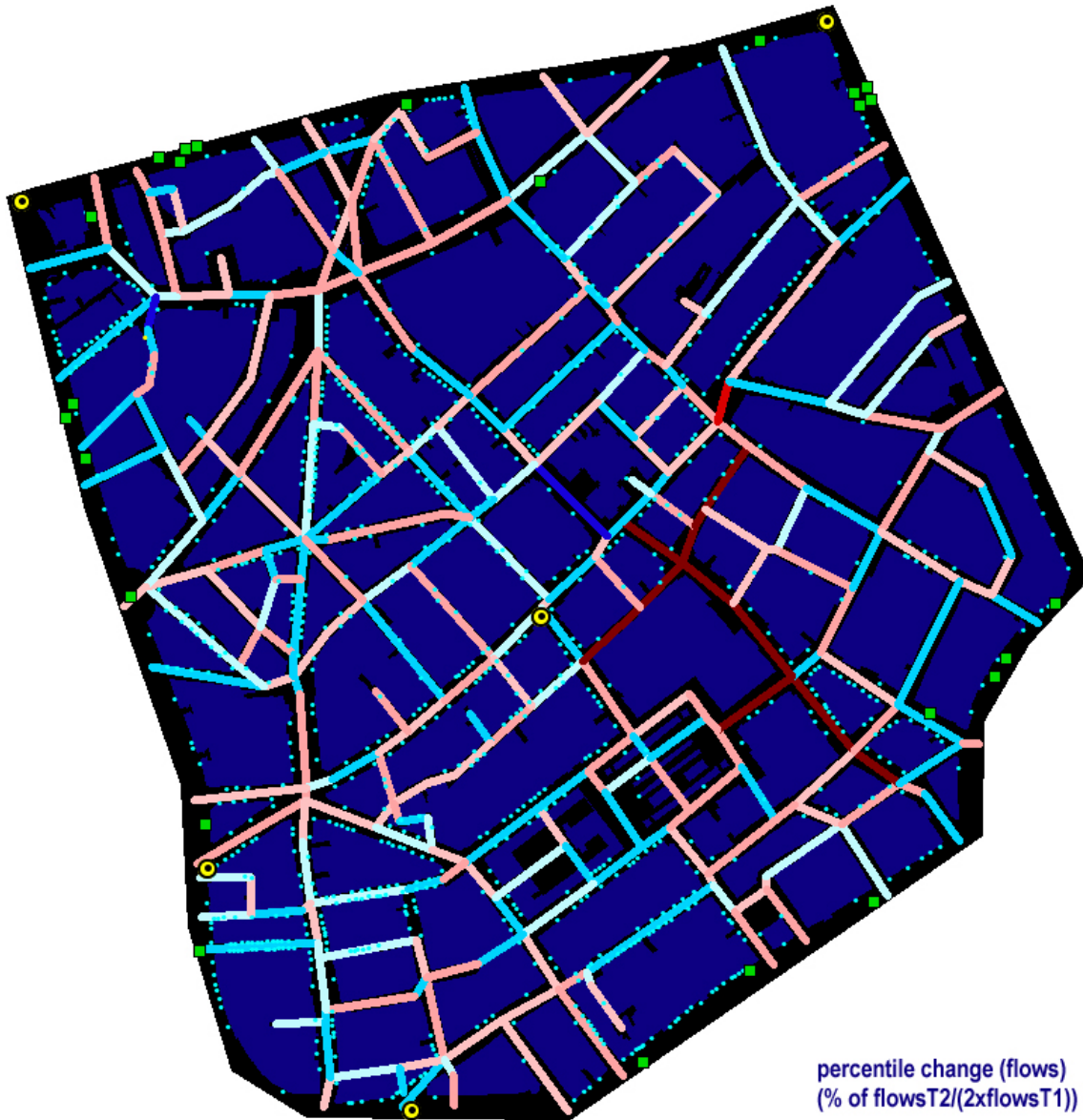
Agent Based Models at CASA

Thursday 19th January 2006



UCL

4. Scenario



percentile change (flows)
(% of flows $T_2 / (2 \times \text{flows}T_1)$)





1. Calibration – Validation
2. More detailed classification of uses
3. Graphic User Interface (GUI)
4. Local movement / presentation
5. Development of remaining modules

Micro-scale simulation of pedestrian movement using space metrics, agent strategies and 3D visualisation

Combined methods for pedestrian movement simulation using XSI Behavior®

First stage

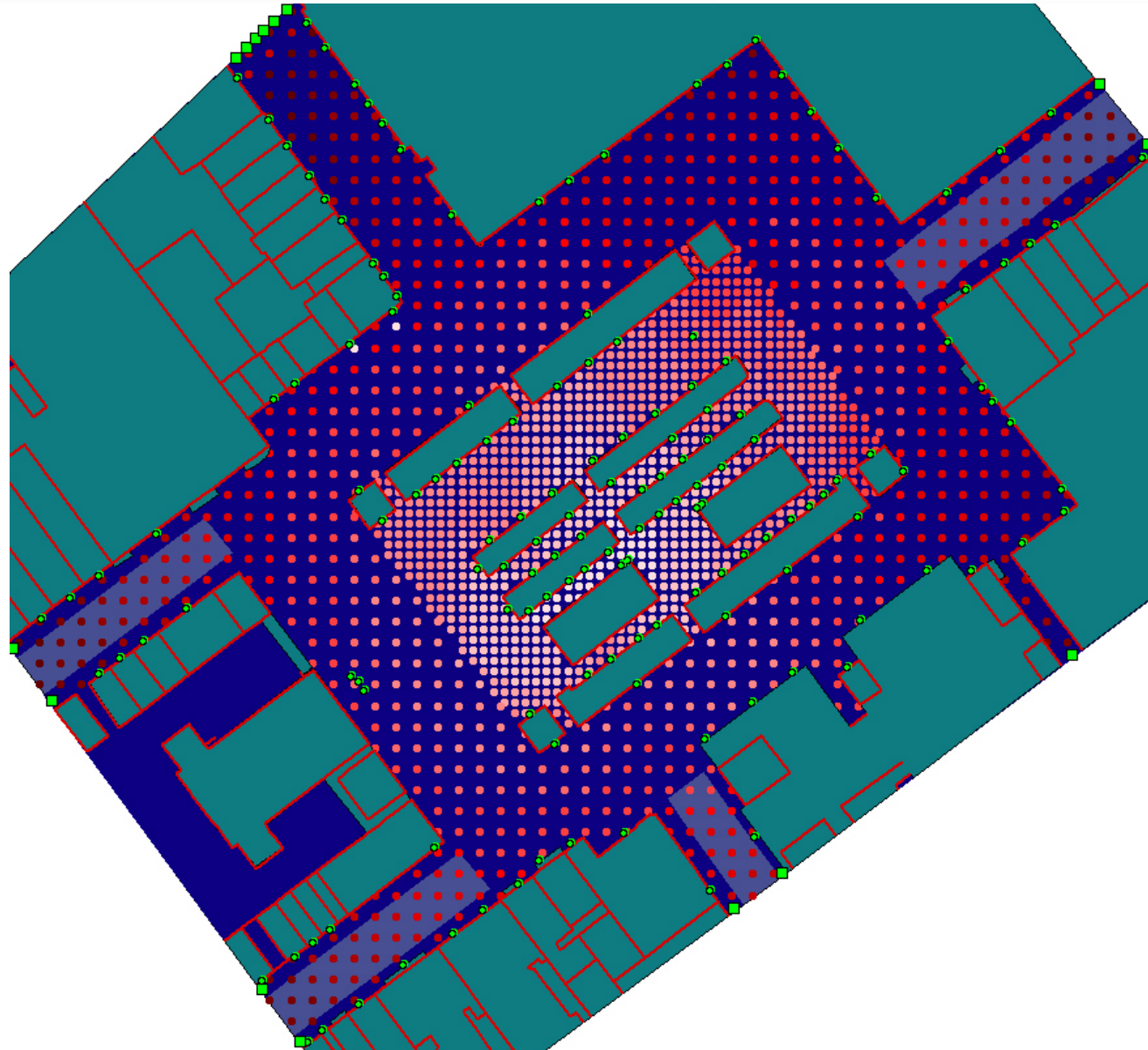
1. A set of algorithms is used to calculate route complexity and other space metrics of the study area's space.
2. A set of points is used
It can be laid out in a grid format or any other arrangement that gives an optimal coverage of the space based on its structure – for instance, points' density correlated to corridor width or distance from walking restrictions (buildings or other non-walkable areas).
3. For each point, the route complexity is calculated (number of changes of direction / turns), minimum network distance and minimum angular change to any other point.

Agent Based Models at CASA

Thursday 19th January 2006



UCL



Distance of every node from a specific origin/destination

Second stage

1. Pedestrian agents profiled to utilise various strategies/tactics (minimisation of distance, angular change, number of turns or composite strategies) are used in order to identify the trajectories that they are likely to follow in order to reach from any given origin to every destination.
2. Origins and destinations are identified as the points closer to a retail unit's or any building's entrance and points closer to the area's entry gates (study area's boundaries).
3. Any origin is at the same time a destination. So in the case of a set of n origins/destinations, $n \times (n-1)$ trajectories are produced for each strategy (depending on the agent's profile/strategy the route from i to j may differ from the one from j to i) for each profile.

Third stage

Each trajectory is loaded with a weight representing the expected importance (flow load) between the origin destination pair that it links. Trajectory weights are analogous to expected visits to the set of entrances.

Once the trajectories and their weights are identified, each point is informed about the number of trajectories that fall within its space and their weights' sum.

The sum of the weights allocated to each point represents the expected pedestrian density in the point's cell.

To refine the simulation's detail the second stage of the process is repeated taking pedestrian densities into consideration during the agents' movement.

- Agents now give preference to movement in less congested space... *Repeat to converge*

Fourth stage

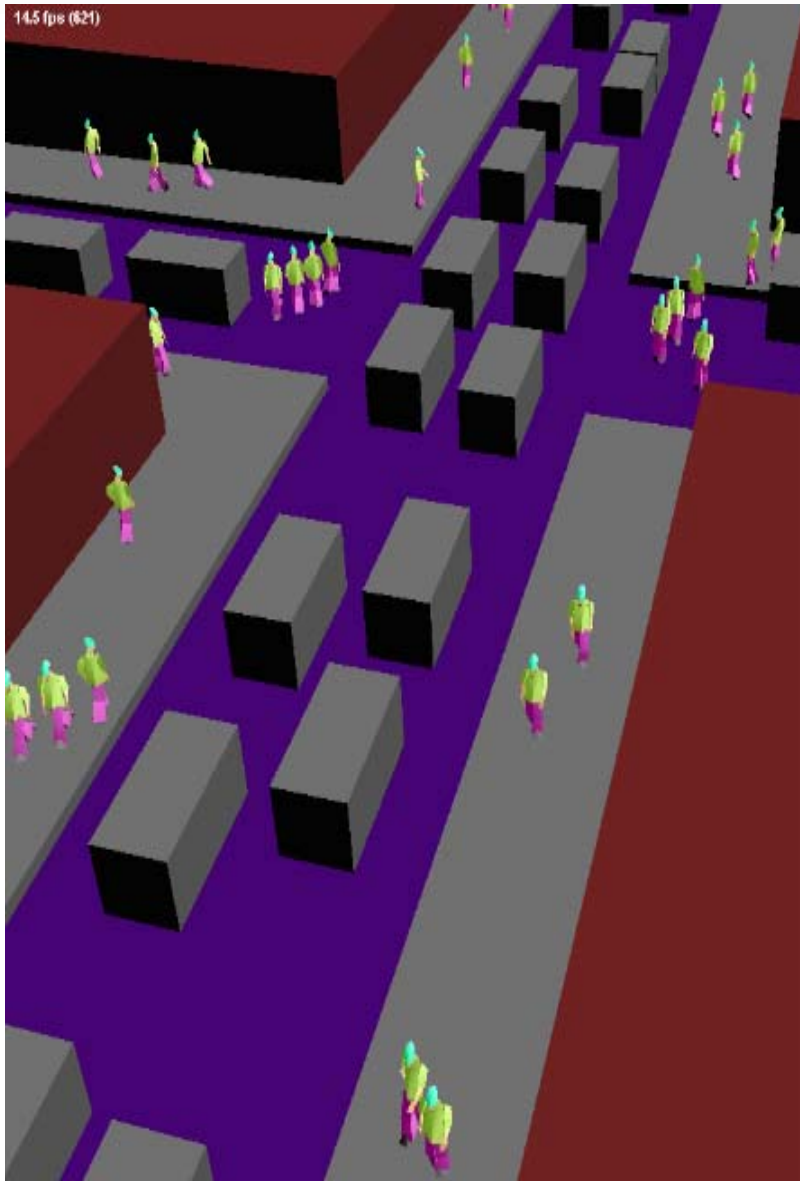
Final trajectories produced after the iteration of the previous stage are appropriately formatted and entered into a 3D crowd simulation suite (XSI Behavior®).

Each trajectory is loaded with agents based on its weight and the combined outcome is presented through a real time 3D visualisation.

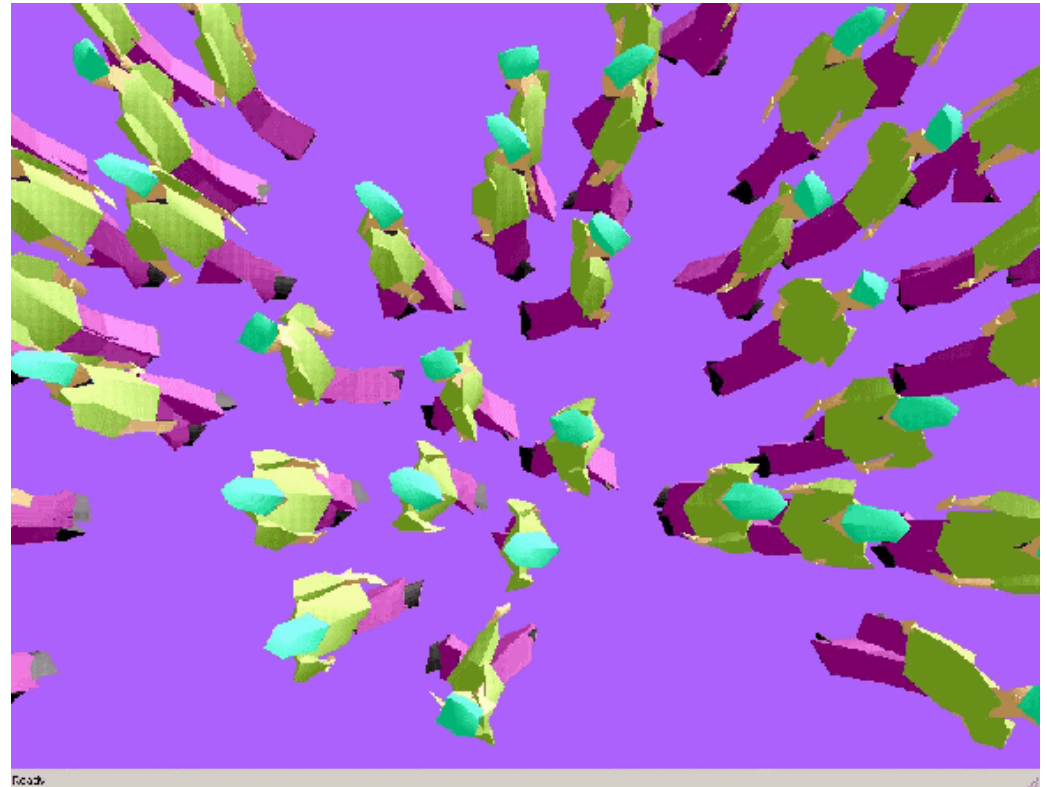
Observation of the visualised system (buildings pedestrians and movement) allows for identification of local stresses, location where safety issues may arise and related analysis.

Agent Based Models at CASA

Thursday 19th January 2006



Demo of XSI Behavior®
visualisation screen



Vassilis Zachariadis - An Agent-Based Approach to the Simulation of Pedestrian Movement

Thank You