Procedural Generation of Road Systems in Computer Games

Robert Giacinto

robert.giacinto@gmail.com

Abstract—The availability of powerful yet affordable computers in the consumer market has led to an increase in the demand for high-quality and diverse content in modern computer games. These requirements cannot be met solely by the artists who produce them but the development and use of procedural techniques has become worthwhile in supporting the assets creation process. This paper presents a pheromone-based multiagent system for the procedural generation of artificial road networks as one building block for the creation of artificial urban structures.

Index Terms—Procedural Content Creation, Multiagent System

I. Introduction and Related Works

The expectations regarding the graphical content and its quality in modern computer games cannot be satisfied solely by an increasing production budget. For this reason there is a growing interest in ways to procedurally generate assets to meet those expectations and needs from the different stakeholders without an increase in expenses.

Examples for the use of procedurally generated content can be found in a growing number of games and various scopes. The most common technique that has been around for more than a decade is the generation of artificial height maps in strategy games like The Settlers (Bluebyte), StarCraft (Blizzard) or Tribal Trouble (Oddlabs) which extended the height-map creation by a model of artificial erosion to increase the degree of visual realism of the game [1]. It not only increased the replay value of those games but also helped to reduce the size of the product.

A more recent and ambitious example for the use of procedural techniques is the creation of artificial trees and forests as in The Elder Scrolls IV: Oblivion [2] which uses the SpeedTree engine [3].

The CityEngine by Parish et al. [4] is capable of creating believable urban structures by processing different inputs provided by the user (population distribution in the city, landscape features and other statistical data). This creation process can be further influenced by the user by the definition of new building rules using an extended L-System which are applied on different parts of the city. That way the user can directly influence the outcome of the CityEngine.

Another approach to define and create urban structures in a top-down manner is described by Greuter et al. [5] which uses a grid of lots of a fixed size and assigns a specific type of use to it. The result is a city that looks more generic than the urban structures from the CityEngine.

The CityBuilder by Lechner et al. [6] procedurally generates patterns of urban land use by simulating the interaction of different types of agents. Their system includes three types of land use: commercial use, industrial use and residential use. The simulation runs indefinitely but can be stopped by the user at any time as the user sees fit. The result of the simulation can be influenced interactively by the user at runtime by using incentives called honey which is manually added positive or negative feedback to the system that adjusts the agent’s behaviour.

This paper presents a bottom-up system design to create and adjust urban structures or whole cities at runtime in computer games. Structures can emerge, change or be discarded over time based on the interaction of individual social agents. Following the example of eusocial insects like ants it is using a pheromone system as the base for the interaction and communication between individual agents. The origin of this development is an emergent process where local information and actions lead to a global development.

II. System Design

The system was designed with a high modularity and extensibility in mind. The underlying conceptual idea is presented in fig. 1.

There are two sub-components (two different types of agents) within the system, persons and buildings. The latter are used for the person’s need satisfaction. The world component is a singleton object that stores the state of the world and makes local information available to the individual persons.

The developed prototype was implemented with JADE a multi agent framework for Java that implements an ACL compliant interface which enables agents from different simulations to communicate with agents of a running instance.

III. Agents and need satisfaction

All agents own an inherent set of different needs that need to be satisfied to survive in the simulated world. To decide which need has to be satisfied next and what actions have to be taken to achieve this goal the agent is controlled by a decision function which is implemented as a decision component. This component manages the all agent’s needs and triggers actions based on the priority (or importance) of a need in question.
A. Needs

Agents have different needs that are differently relevant to the maintenance of the agent’s organism. In this simulation, needs are formalised by the tuple \((v_{\text{current}}, \lambda, p)\), with \(v_{\text{current}}\) being the current energy value of the need, the trigger probability function \(\lambda\) and the priority (or relevance) \(p\).

These needs are the essential driving force for an agent to interact with other agents or the world. The relevance of a need is taken from Maslow’s hierarchy of needs [7] from which the most important ones were implemented:

- **Physiological and Safety needs**
  - Hunger
  - Sleep
  - Security
  - Financial Security

- **Love and Belonging**
  - Social Contacts

Each need can, depending on its relevance, its current energy level and its trigger probability, be an agent’s motivation for future actions. They are updated and handled by a need manager component at discrete time steps.

B. Need System and Need Management

The need system is implemented by a Need Manager component that updates all needs of an agent. The order in which the needs can trigger a certain behaviour when there is more than one need that has to be satisfied depends on their priority according to Maslow’s hierarchy of needs.

The trigger probability function \(\lambda\) includes different input parameters from the agent and the world. As a subcomponent of a person agent the Need Manager tries to maximise the level of satisfaction where satisfaction is defined as the absence of unsatisfied needs.

The need system evaluates the current state of the agent including the trigger probability of the needs at discrete time steps. When the Need Manager selects a need for satisfaction, the component triggers a need satisfaction behaviour that takes control over the agent. Is the agent not at a location where it is able to satisfy a need, the agent can facilitate a steering component that navigates it through the world. To find a resource that is needed by the agent, the steering behaviour takes the local pheromone information around the current position of the agent and moves along the gradient that leads it to a location where it is able to satisfy the need.

In all but one case the pheromones are emitted by buildings that are service providers for a specific need that can be satisfied in their vicinity. The only special case is the need for social interaction and its related pheromone; this is emitted by agents that are looking for a social contact.

Fig. 2 depicts the activities of the need satisfaction as it is implemented in the Need Manager subcomponent. The need satisfaction consists of two main stages. The update of all needs where the most important of the unsatisfied needs is added to a job queue and a job manager stage in which one need is selected from the queue and a new need satisfaction behaviour is triggered. In most cases this behaviour depends heavily on the steering behaviour of the agent which moves it to a place in the world where the need can be satisfied.

C. Pheromones

Agents neither have an internal knowledge representation of the world (e.g. an internal map that can be used for navigational tasks) nor do they have access to global knowledge that is maintained by the world itself.

Decisions are made upon the evaluation of the pheromones in the Moore vicinity of the agent. Thus communication between agents takes place indirectly on the basis of different pheromones. The way how pheromones as a communication medium work for
multiagent system is described in [8] and can be summed up as follows:

- Pheromones of the same type add their strength
- The strength of pheromones decreases over time
- Pheromones diffuse in space
- Pheromones can be perceived by agents and build the foundation of the decision-making process

The two components that rely on this kind of communication are the steering behaviour and the need satisfaction behaviour of the agent. This leads to two main types of pheromones in this simulation.

1) Road pheromones
   Road pheromones are emitted by an agent while it is controlled by an active steering behaviour. It leaves a trace of road pheromones that can be used by all agents to orient itself.

2) Interaction pheromones
   a) Social Pheromone
      Social Pheromones can be used by agents to find other agents in the vicinity to interact with each other. It corresponds with the need for love and belonging according to Maslow [7].
   b) Building pheromones
      Each type of building that exists in the simulation is able to emit a distinct pheromone. In some cases can this type of pheromone be bound to a specific owner (i.e. the home of an agent).
      i) Home pheromone
         This pheromone is an example for a pheromone that can only be processed by exactly one agent. It is used to find the home of an agent that provides shelter and a place to sleep.
      ii) Service pheromones
         This type of pheromones is emitted by buildings that provide services that are needed for the satisfaction of needs (e.g. selling food). Agents can use the information to find a way to a service provider. For a building being able to provide a service, there has to be an employed agent that is producing this good. In that way agents benefit from service buildings in two ways: 1. agents can satisfy the need for financial security and 2. they are the source of different services that are utilised by other agents.
      iii) Work pheromones
         As mentioned above buildings need employees to be able to offer a service. By emitting a work pheromone a building propagates the information of a vacancy in the system. Every unemployed agent can react to this pheromone by following the gradient of this pheromone. The vacancy is assigned to an agent in a FCFS manner.

D. Behaviours

Actions triggered by an agent’s Need Manager are implemented as instances of a behaviour class and can be loaded dynamically on demand and executed by the corresponding agent. There are two types of behaviour that will be described in more detail in the following sections:

- Need satisfaction behaviour
  This behaviour is triggered when a need is selected by the Need Manager.
- Steering Behaviour
  The steering behaviour is utilised when an agent has to navigate through the world to accomplish a task triggered by the Need Manager.

1) Steering Behaviour: The steering behaviour moves the agent it is controlling to a point in the world by following as many existing roads as possible.

Road networks emerge over time while agents interact with their environment by leaving pheromone trails and using existing trails of other agents to find a way to their destination and so enforce existing patterns. The current implementation of the steering behaviour gets the type of service provider the agent is looking for as input from the need manager and uses it as the destination of the steering behaviour. This behaviour takes control over the agent while being active and steers the agent to the destination by using the pheromone informations in the vicinity.

Fig.3 shows how an agent can find a free house to live in. The active steering behaviour follows the pheromone of a free house until the agent is near enough to move in.

The general flow of the steering behaviour is as follows:

```python
def moveTo(service type):
    while agent is not at target location:
        get current vicinity of agent
        if pheromone of service type exists in vicinity
           and strength is larger than threshold:
               agent has reached target location
```
Abbildung 4. Simulation result of four agents after 70 minutes.

```java
if pheromone of service type exists in vicinity[
    set heading of agent to direction of the maximum
    if road is in vicinity of agent and heading matches[
        set heading of agent to the weighted mean of road and pheromone direction
    ]
]
else if road is in vicinity of agent[
    set heading of agent to the direction of the road
]
else[
    set heading of agent to random direction
]
move agent one step in the new direction
```

The behaviour ends when the agents reaches an area in which a pheromone exceeds an arbitrary threshold. There the needs satisfaction behaviour can start satisfying the need which is associated with the service provider in the vicinity by decreasing the need value by a random number.

Less frequently used pheromone trails dissolve over time and agents are less likely to use them when evaluating the pheromones strength. That way the system optimises existing routes and forgets wrong or less optimal paths through the road network.

IV. Results

The prototype is capable of creating basic road patterns that can be used to navigate the agents from one place to another. The structure of the emerging city evolves over time so that less important roads dissolve and more important pheromone trails are enforced. This corresponds to the foraging behaviour of ants as it has been described in various studies (eg: [9], [10], [11]).

The results of the interaction between four agents and their need satisfaction activities are shown in fig. 4. The emergence of main roads that are used by agents more frequently can easily be identified. However, the performance of the prototype is very low and needs a thorough evaluation and profiling before extending it.

V. Evaluation and Future Work

In this paper I could show that it is possible to describe and simulate the emergence of road networks in cities on the basis of just a few needs and a decision-making process which only takes the information in the vicinity of an agent into account. The resulting prototype should be seen as a proof of concept work which leads to more sophisticated questions concerning the use of multiagent systems in the simulation and creation of urban structures for computer games:

- What effect does the topology of a landscape have on the development of cities, their road networks and the emergence of urban structures?
- How can knowledge on segregation and population distributions be integrated into the simulation to improve the realism of the simulation results?
- How and with what data structure could the simulation or its results be integrated into existing game engines?

Future works should extend the world that surrounds the agents in terms of additional topological informations about the world since features like mountains, plains or access to water or food are essential for a decision-making process and directly influence the shape of a city. At a later point the simulation will need to be evaluated and compared to real world data of existing cities that and additional evaluation criteria specific to the needs of the computer game industry.

References

