The design and simulation of traffic networks in virtual environments

Christopher Steven Applegate

A thesis submitted for the degree of
Doctor of Philosophy
at the University of East Anglia
School of Computing Sciences
April 2013

The design and simulation of traffic networks in virtual environments

Christopher Steven Applegate

Publications

Several parts of this thesis have been published as follows:

- Applegate, C.S., Laycock, S.D., and Day, A.M., *"Real-Time Traffic Simulation Using Cellular Automata"*, In EG UK Theory and Practice of Computer Graphics, Sheffield, UK, pp 91-98, September 2010.
- Applegate, C.S., Laycock, S.D., and Day, A.M., *"A Sketch-Based System for Highway Design"*, Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling, Vancouver, British Columbia, Canada, pp 55-62, August 2011.
- Applegate, C.S., Laycock, S.D., and Day, A.M., *"A Sketch-Based Algorithm for Generating Roads"*, Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling, Vancouver, British Columbia, Canada, August 2011.
- Applegate, C.S., Laycock, S.D., and Day, A.M., *"A sketch-based system for highway design with user-specified regions of influence"*, Computers & Graphics, Volume 36, Issue 6, October 2012, Pages 685-695, ISSN 0097-8493.

Acknowledgements

I would like to thank the members of my supervisory team, Prof. Andy Day, Dr. Stephen Laycock, and Dr. Robert Laycock, for their support and guidance throughout the course of this project. Their collective advice and experience has been invaluable to me throughout this process. In addition, I would like to thank my viva examiners, Prof. Alan Chalmers and Dr. Barry Theobald.

A special thanks goes to my family and friends for all their help. In particular, I would like to thank my parents, Steven and Tracy Applegate, my girlfriend, Sarah Boram, and also Christopher Asque, for their collective support and advice, and for taking the time to proofread this thesis.

Abstract

For over half a century, researchers from a diverse set of disciplines have been studying the behaviour of traffic flow to better understand the causes of traffic congestion, accidents, and related phenomena. As the global population continues to rise, there is an increasing demand for more efficient and effective transportation infrastructures that are able to accommodate a greater number of civilians without compromising travel times, journey quality, cost, or accessibility. With recent advances in computing technology, transportation infrastructures are now typically developed using design and simulation packages that enable engineers to accurately model large-scale road networks and evaluate their designs through visual simulation. However, as these projects increase in scale and complexity, methodologies to intuitively design more complex and realistic simulations are highly desirable. The need of such technology translates across to the entertainment industry, where traffic simulations are integrated into computer games, television, film, and virtual tourism applications to enhance the realism and believability of the simulated scenario.

In this thesis two significant challenges related to the design and simulation of traffic networks for use in virtual environments are presented. The first challenge is the development of intuitive techniques to assist the design and construction of high-fidelity three-dimensional road networks for use in both urban and rural virtual environments. The second challenge considers the implementation of computational models to accurately simulate the behaviour of drivers and pedestrians in transportation networks, in real time. An overview of the literature in the field is presented in this work with novel contributions relating to the challenges defined above.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and aims

For over half a century, researchers from a diverse set of disciplines have been studying the behaviour of traffic flow to better understand the causes of traffic congestion, accidents, and related phenomena. As the global population continues to rise, there is an increasing demand for more efficient and effective transportation infrastructures that are able to accommodate a greater number of civilians without compromising travel times, journey quality, cost, or accessibility. Furthermore, with increasing concerns of environmental pollution, the energy crisis, and the current financial situation of the world economy, it has never been more important to design suitable and effective methods of transportation.

The necessity of effective design is most apparent when studying transportation infrastructure in developing countries, such as India and China, where emerging markets for automobile trade are seeing phenomenal growth. The China Association of Automobile Manufacturers reported that 13.6 million vehicles were sold in China in 2008, and for the first time in history, more cars were sold in China than in the United States. However, despite recent surges in automobile sales, non-motorised vehicles are still frequently used for short-distance trips, due to low income levels and general convenience. In developing areas, both motorised and non-motorised traffic

Product	Simulation		Price(£)
	Vehicle	Pedestrian	
Quadstone Paramics UAF Pedestrians	✓	✓	24,223
Quadstone Paramics Standard	✓	×	10,099
TTS Aimsun Standard	✓	✓*	7,004
Caliper Transmodeler	✓	×	6,725
Trafficware Synchro Studio 8	✓	×	2,757

Table 1.1: Examples of traffic simulation products with details of their key features and prices. Prices correct at time of publication. Fields marked with an asterisk indicate optional features.

are permitted to travel in unison. However, without appropriate control structures, the flow of high-volume and mixed traffic is unsustainable, and typically results in the formation of start-and-stop waves, or even complete jams.

With recent advances in computing technology, traffic solutions are now typically designed and developed using simulation packages which allow transportation engineers to accurately model road networks and evaluate their designs through visual simulation (Table 1.1). These products provide three-dimensional representations of different traffic scenarios prior to their actual construction, allowing professionals, local authorities, and the general public to conceptually understand the expected impact on the surrounding environment. However, as these projects increase in both scale and complexity, methodologies to model larger-scale and more realistic simulations are desired. The need of such technology translates across to the entertainment industry, where traffic simulations are integrated into computer games, television, film, and virtual tourism applications to enhance the realism and believability of the simulated scenario. Recent state-of-the-art commercial computer games, such as *Grand Theft Auto IV*, *Just Cause 2*, and *Crysis 2*, have explorable environments that include extensive and complex road networks across both urban and rural terrains.

In this thesis, two significant challenges related to the design and simulation of traffic networks for use in virtual environments are identified:

1. The development of intuitive techniques to assist the design and construction of high-fidelity three-dimensional road networks for use in both urban and rural virtual environments.
2. The implementation of computational models to accurately simulate the behaviour of drivers and pedestrians in transportation networks, in real time.

1.2 Novel contributions

In the remainder of this thesis, several significant novel contributions relating to the challenges defined in the previous section are presented:

- A technique to automatically generate road networks from digital map data. The algorithm requires a connected set of vertices and edges as input, and determines road topology, such as the location of junctions and the layout of individual roads, using a set of rules that considers the number of connections at each vertex and the angle between connecting edges.
- A sketch-based tool to semi-automate the design, creation, and visualisation of road networks for virtual environments that is guided by input sketches and a combination of prioritised constraints, such as the curvature of roads, their inclination, and the volume of ground that would be displaced during construction. This technique extends previous work in the field through the development of an algorithm that constrains the geometric properties of roads in three-dimensions to generate road networks across both flat and undulating terrain.
- 'Influence Regions' are introduced, which are user-specified areas of a virtual environment that can be used as constraints in the sketch-based tool to influence the path of generated roads. These regions provide guidance throughout the road design process and can be tailored to either attract or repel roads

to/from certain obstacles or designated areas, such as forestation, listed buildings, marshland, etc.

- A basic framework for modelling the interaction between vehicles and pedestrians in urban environments in real time. The model provides a basic level of interaction, derived from well-established and recognised behaviour models for simulating vehicles [Gip81] and pedestrians [Rey99], independently.

1.3 Thesis structure

This section describes the content and arrangement of the remaining chapters in this thesis.

Chapter 2 provides an overview of the literature in the field, focusing on significant contributions and current state-of-the-art. Topics discussed include techniques to generate large-scale and complex road networks for use in virtual environments, and computational models used to simulate the behaviour of drivers and pedestrians. The limitations of current systems are also assessed to highlight some of the most significant challenges that remain in the field. These challenges are further investigated throughout the remainder of this thesis.

Chapter 3 presents a novel technique to automatically generate road networks from digital map data. The algorithm requires a connected set of vertices and edges of an existing road network, and determines the location of junctions and the layout of individual roads based on the number of connections at each vertex and the angle between connecting edges. A road graph is then constructed by segmenting each road into a series of cells that can be used as waypoints for a traffic behaviour model. The performance of the algorithm is evaluated by comparing the topologies of generated road networks with their real-world counterparts.

Chapter 4 presents a novel sketch-based tool to semi-automate the design, creation, and visualisation of road networks for virtual environments. The tool is guided by input sketches and a combination of prioritised constraints, such as the curvature of roads, their inclination, and the volume of ground that would be displaced during construction. The technique extends previous work in the field by developing an algorithm that constrains the geometric properties of roads in three-dimensions to generate road networks across both flat and undulating terrain. Constraints can be tailored to generate roads that exhibit particular characteristics, such as roads with minimal inclination that cut through the environment and roads with high curvature that meander across the terrain. Furthermore, 'Influence Regions' are introduced, which are user-specified areas of the environment that influence the path of generated roads. These regions provide guidance throughout the road design process and can be tailored to either attract or repel roads to or from certain obstacles or designated areas, such as forestation, listed buildings, marshland, etc. A user study is conducted to evaluate the usability of the system and to evaluate the quality of roads generated in a diverse range of scenarios.

Chapter 5 presents a model for simulating high-detail traffic networks in real time. The model presented is agent-based and operates on rules influenced by the pioneering work of Gipps [Gip81]. Furthermore, the work of Nagel and Schreckenberg [NS92] is extended to produce a cellular automata model for low-detail simulations, using interpolation techniques to overcome the animation continuity issues that are typically associated with discrete behaviour models.

Chapter 6 introduces a novel basic framework for modelling the interaction between vehicles and pedestrians in urban environments. The model presented is not a comprehensive solution to vehicle-pedestrian simulation, but provides a basic level of interaction, derived from well-established and recognised behaviour models for simulating vehicles [Gip81] and pedestrians [Rey99], independently. The work presented in this chapter aims to provide motivation for continued research in this area.

Chapter 7 concludes this thesis, providing an overview of the work presented in previous chapters, as well as, possible future directions.

Chapter 2

Background research

In this chapter an overview of the literature in the field is presented, with a focus on significant contributions and current state-of-the-art. Topics discussed include techniques to generate large-scale and complex road networks for use in virtual environments, and computational models used to simulate the behaviour of vehicles and pedestrians. The limitations of current systems are also assessed to highlight some of the most significant challenges that remain in the field. These challenges are investigated throughout the remainder of this thesis.

2.1 Road generation

To simulate traffic within a virtual environment, an efficient infrastructure that accurately represents the topology of the entire road network with details of individual roads and their connections is required. This infrastructure is known as a road graph. With recent advances in computing technology, there is an elevated demand for larger, more complex, and more realistic virtual environments that contain road networks. This presents a significant challenge for developers of these systems, as additional resources are often required to enhance and create further content, increasing both development times and associated costs. Automated techniques to improve the efficiencies and associated costs of generating content within virtual environments have been studied extensively in the literature, and their effectiveness is investigated in the remainder of this section.

2.1.1 Procedural modelling

Procedural modelling is a technique that is used ubiquitously in computer graphics to automatically generate three-dimensional models and textures using a set of rules. Procedural algorithms are used to generate both natural and artificial objects including plants [PL90], buildings [MWH$^+$06], and terrains [Ols04]. Furthermore, procedurally generated textures are used extensively within virtual environments to add both variety and detail to geometric models, using noise at different fluctuation levels [Per85]. They are also used to create realistic representations of natural materials such as wood, marble, granite, metal, and stone [EMP$^+$02], as well as natural elements including clouds, fire, and smoke [EMP$^+$02]. Procedural algorithms are often employed when developing large-scale virtual environments, as they can significantly reduce the laborious and time-consuming process of content creation, reducing production times and associated costs without compromising diversity and detail. For applications that require specialised or more artistic content, procedural techniques can provide an initial framework, which can be iteratively modified to meet the design requirements.

A variety of techniques to procedurally generate roads have been studied in the literature. In 2001 Parish and Müller [PM01] presented a system to procedurally generate large cities from a series of maps. These maps detail geographical and socio-statistical information, including land elevation, building heights, population density, zone classification (residential, commercial, industrial, etc.), and terrain classification (land, water, vegetation, etc.). Using this information, their system generates a road graph using an extension to traditional L-systems [PL90], where proposed parameters are influenced and modified by ‘global goals’ and ‘local constraints’ to produce realistic road layouts across a variety of environments. Global goals are used to promote the growth of highways towards peaks in the population density, and encourage local streets to adhere to common road patterns (checker, radial, branching, etc.). Local constraints are applied to ensure that roads avoid ‘illegal’ areas, such as parks and lakes, and are used to connect neighbouring road segments to form junctions, intersections, and crossings.

In 2007 Kelly and McCabe presented CityGen [KM07], an interactive system to procedurally generate cities. In their work, the authors divide the task of creating a virtual city into three distinct processes:

- Primary road generation
- Secondary road generation
- Building generation

Initially, the user defines the topological structure of the primary road network by creating ‘control points’ throughout the environment. These points form a high-level graph where the nodes correspond to road intersections and the edges represent primary connections between these nodes. The high-level graph is then sampled, plotted and interpolated to form a low-level graph that represents the path of primary roads in greater detail. To provide more control of the road generation process, different sample selection strategies are employed:

- Minimum elevation - This strategy selects samples with the lowest elevation, producing paths similar to river streams.

- Minimum elevation difference - This strategy attempts to avoid drops and rises in elevation. However, as the sampling process occurs simultaneously from both the source and destination, a significant discrepancy in elevation can occur at the point where the two sampling processes meet, producing unrealistic results.
- Even elevation difference - To improve the above technique, this strategy attempts to distribute the elevation evenly across the samples by comparing the difference in elevation between each sample and the destination.

Regions bounded by primary roads are then extracted to create ‘city cells’, which are populated with secondary roads using L-systems. These roads can be customised to exhibit certain characteristics with parameters relating to the size of road segments, the number of branches that emanate from nodes, and connectivity probabilities. A deviance parameter is also used to introduce noise, which is used to distort raster patterns to better resemble industrial and organic street layouts.

Chen et al. [CEW$^+$08] propose a method to interactively and intuitively generate large-scale road networks, based on tensor fields that guide the structural layout of roads. In their work, the authors present a variety of techniques to manipulate these tensor fields to obtain typical urban street networks. The techniques presented focus on the layout of roads, rather than geometric detail, and are limited to environments with relatively flat terrain, as they are unable to displace the height of the landscape to produce a smooth road inclination.

An alternative agent-based approach for generating complex road networks within urban environments was developed by Lerchner et al. [LWW03]. In their system, multiple agents traverse and interact with a user-defined environment to generate and extend the road network. There are two types of agent, extenders and connectors. Whenever a road segment is generated, surrounding patches on the terrain are stamped with proximities to the road network. The role of the extender agent is to expand the city by traversing the environment and locating isolated patches. Once identified, the agent uses the proximity values to find a suitable route from the isolated patch to a local road. To produce roads that are suitable for the environment, the selected route is influenced by environmental factors, such as land elevation and design constraints that include road density and proximity to local intersections. The connector agent constructs routes between existing roads to create a complex lattice of paths throughout the environment. These agents are only permitted to traverse the road network and are instructed to take periodic and random samples of road patches within a specified range of their location. If the route between an agent and its sample is too complex, a more direct route is created.

The authors describe how agent behaviour could be tailored to produce road networks that exhibit particular characteristics. For example, constraints could be set-up to only permit the placement of roads over a pre-established pattern, with a limitation on the level of acceptable deviation. In subsequent work [LWR^{+}04], the authors introduce a primary road agent to build highways that connect densely populated areas to the city centre. These agents traverse the environment searching for peaks in the population density that are isolated from the primary road network. Once a population peak has been identified, an agent will generate a road from the peak to the city centre. To prevent a surplus of primary roads accumulating at the city centre, the agent will search the surrounding area for an alternative primary road to connect to.

Galin et al. [GPMG10] proposed an automated technique for generating roads in complex environments that contain rivers, lakes, mountains, and forests. A weighted shortest-path algorithm is performed between an initial and a final location on the landscape, which is discretised into a regular grid. Weightings can be adjusted interactively to create roads which exhibit certain characteristics. However, there is no facility to explicitly describe the desired path of the road, and the output may require significant editing to meet user expectations. More recently, Talton et al. [TLL$^+$11] have proposed an algorithm to generate grammar-based procedural models from a high-level specification of the desired output and an associated grammar. The input specification can be image-based, volume-based, or as an analytical objective. The algorithm has been demonstrated on trees, cities, buildings, and Mondrian paintings.

Techniques to enhance the visual realism of urban environments have been studied by Reynolds [Rey10], who presents an algorithm to procedurally generate accurate and highly-detailed road markings throughout a city scene by tagging the terrain with semantic information determined from its layout. Furthermore, methods of road degradation are employed, and street furniture including bus shelters, post boxes, traffic signals, and street lamps, are automatically integrated into the environment.

2.1.2 Sketch-based modelling

Recently, there has been an abundance of research on sketch-based modelling techniques that allow novice and advanced users to intuitively design and develop geometric structures. Jeon et al. [JJLC10] presents a sketch-based interface to create three-dimensional animations, using a multiple-pass system that allows users to draw the trajectory of motion in different planes. The authors approximate the input sketch using a uniform cubic B-spline due to its affine invariance and local control. To provide increased functionality, additional parameters can be specified using gestures. Sketch-based systems for road design have been explored by Blessing et al. [BSAR09], who generate the shape and location of a road by segmenting user sketches into a series of straight lines and Bézier curves. Consecutive tangent lines are then connected using cubic curves to construct a continuous centre line for the road. The resulting roads are generated in real time, but the system is restricted to the generation of roads in

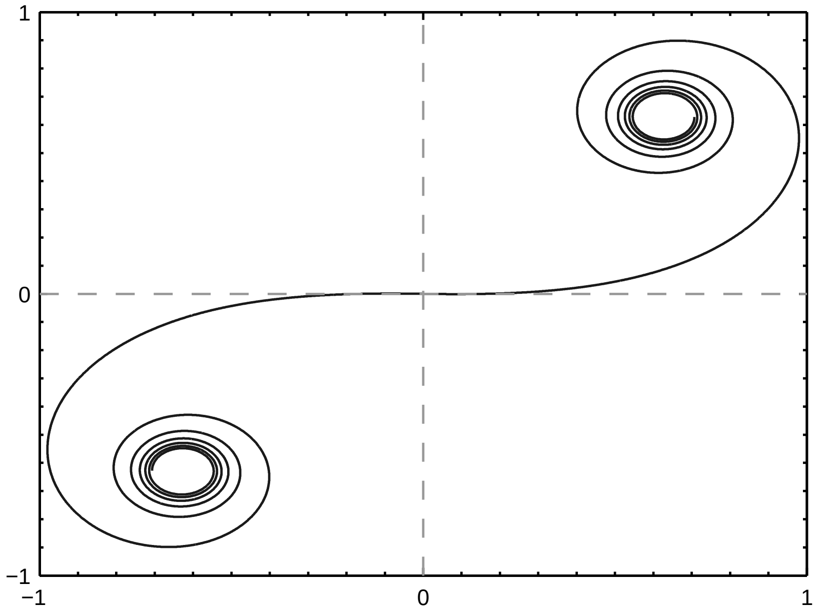


Figure 2.1: Clothoid: A curve whose curvature varies linearly along its arc length.

two-dimensions only.

McCrae and Singh investigated sketch-based techniques for road generation using piecewise clothoid curves [MS09b, MS09a]. Clothoid curves (also known as Euler Spirals or Cornu Spirals) (Figure 2.1) are used in road design, as their curvature varies linearly along the length of its arc. In their work, the authors present a technique for creating roads from sketches by transforming input strokes into a discrete curvature-space representation. In this space, the sketch is simplified using a dynamic programming algorithm to reduce its complexity, and the resulting linear pieces are mapped to clothoid segments in Euclidean-space using Fresnel integrals:

$$v = \pi B \begin{pmatrix} C(t) \\ S(t) \end{pmatrix}$$

where,

$$C(t) = \int_0^t \cos \frac{\pi}{2} u^2 du$$

$$S(t) = \int_0^t \sin \frac{\pi}{2} u^2 du$$

where v is the two-dimensional Euclidean-space vertex at length t along the arc and πB is a scaling parameter. Evaluating Fresnel integrals can be computationally expensive, therefore rational approximations are used to obtain similar results when performance of the algorithm is more important than accuracy [Hea85]:

$$C(t) \approx \frac{1}{2} - R(t) \sin \left(\frac{1}{2} \pi (A(t) - t^2) \right) \tag{2.1.1}$$

$$S(t) \approx \frac{1}{2} - R(t) \cos \left(\frac{1}{2} \pi (A(t) - t^2) \right) \tag{2.1.2}$$

where,

$$R(t) = \frac{0.506t + 1}{1.79t^2 + 2.054t + \sqrt{2}} \tag{2.1.3}$$

$$A(t) = \frac{1}{0.803t^3 + 1.886t^2 + 2.524t + 2} \tag{2.1.4}$$

The resulting roads are constrained in the plane perpendicular to the world up-axis (y-axis) to produce realistic results on relatively flat terrain. However, their algorithm does not constrain road inclination and could produce roads with gradients that are impossible to navigate, especially if the user were to sketch over mountainous or undulating terrains.

2.1.3 Summary

There are a variety of techniques for generating road networks in virtual environments. Procedural methods dominate the literature, and are able to produce extensive and complex road networks with minimal input. However, resulting networks appear to follow conventional city centre design patterns and require significant fine-tuning to obtain non-standard or customised layouts. Sketch-based techniques for road design provide greater control over the resulting road layout, but current techniques are limited to extremely flat environments, as they do not support mountainous or undulating terrains.

After extensive research, a solution that enables novice and advanced users to create roads intuitively across a variety of landscapes, whilst providing high-level control over the resulting paths of generated roads, could not be identified.

2.2 Behaviour models

Traffic and pedestrians are required to enhance the realism and believability of the simulated environment. Behavioural models are used to automatically control and simulate groups of entities, such as crowds of pedestrians, flocks of birds, and vehicles within a city. Interactions between these entities are usually not pre-defined, but are typically determined in real time and are based on internal and external parameters that change throughout a dynamic simulation. Behavioural models are often used to reduce the workload of animators, and are typically used in projects where it is impractical to animate each entity individually. Behavioural animation techniques have been used extensively in the film industry. Example simulations include the swarm of bats in *Batman Returns*, the wildebeest stampede in *Disney's The Lion King*, and crowds of civilians in *Disney's The Hunchback of Notre-Dame.* The computer game industry has also made use of behavioural animation to control autonomous characters in *Creatures* and *The Sims*, and vehicles in *Grand Theft Auto IV* and *Just Cause 2*.

The most sophisticated commercial package for developing offline autonomous crowds for film, television, and computer game cinematics, is Massive software[1]. The

[1]http://www.massivesoftware.com/

system features rigid body dynamics to realistically simulate the physical properties and forces exerted on characters. Each simulated character uses sensory inputs and a network of fuzzy-logic nodes to determine its behaviour. Massive software was used to animate autonomous crowds in *The Lord of the Rings*, *The Dark Knight*, and *Avatar*, producing photo-realistic visuals, but at the expense of lengthy rendering times. For example, during production of the second instalment of *The Lord of the Rings* trilogy: *The Two Towers*, approximately 10 months of computations were required to render all digital characters on more than 1,000 SGI and IBM Linux workstations [Doy03]. More recently, Massive software has introduced support for interactive applications, such as evacuation simulation and traffic simulation.

Real-time simulations aim to produce the realism of offline solutions, but must achieve this at real time frame rates ($\geq$ 30 FPS is targeted in the gaming industry [RFD05, Rey06, Lie12, Par12]) to provide an acceptable level of interactivity. However, to meet the performance requirements, simulations are often compromised in visual and behavioural realism. Many techniques have been developed to simulate the behaviour of large autonomous groups and crowds in real time. In the remainder of this chapter techniques for simulating both traffic and pedestrians, and the interactions between these entities, in urban and rural virtual environments are discussed.

2.2.1 Traffic simulation

Macroscopic models

It has been observed that the aggregate motion of groups and crowds is analogous to the flow of fluid or gasses under certain conditions. At a macroscopic level, crowds can be simulated as a density continuum using theory from fluid dynamics. Typically, these models do not differentiate between individual agents, and it is therefore difficult to accurately simulate complex crowds with a diverse range of agent types and associated behaviour profiles. However, due to the lack of individual detail, macroscopic models operate at high computational speeds, and are therefore used for large-scale simulations that contain a considerable number of agents.

The pioneering work of Lighthill and Whitham [LW55], and independent work by Richards [Ric56], investigated the behaviour of traffic flow as a density continuum, modelling the macroscopic (or continuous) behaviour of traffic, comparable to fluid dynamics. Their simulation model, more commonly known as the Lighthill-Whitham-Richards Model (LWR Model), uses aggregate variables representing the average velocity of vehicles (v), traffic density (p), and traffic flow (q), to model the collective behaviour of vehicles from the defined relationship between these attributes. The fundamental theory behind the LWR model is based on the assumption that the number of vehicles between any two points along a road will be conserved, providing there are no entrances (sources) or exits (sinks):

$$\frac{\partial p}{\partial t} + \frac{\partial q}{\partial x} = 0$$

where x is a position along the road and t represents time.

More recently, Sewall et al. [SWML10] extended a macroscopic traffic model to produce detailed three-dimensional animations of traffic flow for large-scale networks. Their system uses continuum dynamics to model traffic, but maintains individual vehicle information to display each vehicle. Their method is scalable with multi-core/multi-processor architectures and runs at real-time frame rates.

Microscopic simulations

Microscopic (or discrete) behaviour models are commonly used in traffic simulation software. These models simulate individual agents that exhibit detailed behaviour using 'car-following' and 'lane-changing' rules, which model individual vehicle attributes such as speed, acceleration, braking forces, reaction times, etc. In these rules the movement of each vehicle is calculated based on the position and velocity of the vehicle in front. However, due to their complex nature, microscopic simulations are more computationally demanding than macroscopic models, but will typically produce more realistic visualisations.

In 1981 Gipps [Gip81] described the general form of car-following rules, as detailed below:

$$a_n(t+\tau) = l_n \frac{[v_{n-1}(t) - v_n(t)]^k}{[x_{n-1}(t) - x_n(t)]^m}$$

where:

- $a_n(t)$ is the acceleration of vehicle n at time t
- $v_n(t)$ is the speed of vehicle n at time t
- $x_n(t)$ is the position of vehicle n at time t
- τ is a constant reaction time for all vehicles

Parameters l_n, k, and m are estimated values used to fine-tune the model and have no direct connection to identifiable characteristics of a driver or a vehicle. Gipps evaluated the model and demonstrated that when reasonable values are assigned to these parameters, the model is able to mimic the behaviour of real traffic.

In subsequent work, Gipps described a lane-changing decision model that integrates with his car-following rules [Gip86]. The model handles a variety of urban scenarios and includes support for traffic signals, obstructions, heavy vehicles, and transit lanes (lanes reserved for a subset of vehicles or vehicles containing a minimum number of passengers). The framework presented details three primary factors that are considered when a driver wishes to change lane:

- The possibility to change lanes
- The necessity to change lanes
- The desirability to change lanes

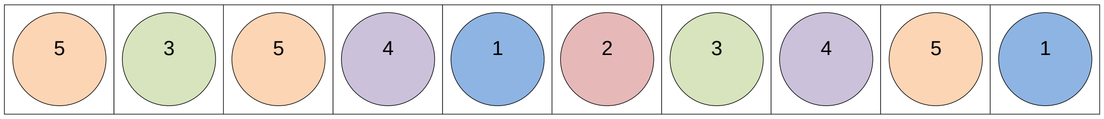
The influence of each of these factors varies with the distance to a vehicle's next turn.

Other significant contributions in this area can be found in the works of Gerlough [Ger55] and Newell [New61]. Popular microscopic traffic simulators include Quadstone Paramics and PTV AG's VISSIM.

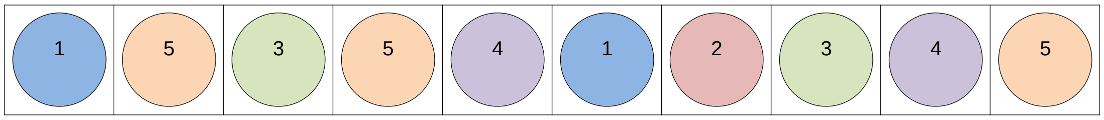
Mesoscopic simulations

Mesoscopic models combine macroscopic and microscopic systems, simulating individual agents with aggregate behaviours. As these systems simulate individual agents, mesoscopic simulations are typically more realistic than macroscopic models. Furthermore, the use of aggregate parameters reduces computation, enabling these systems to simulate a large number of agents in real time.

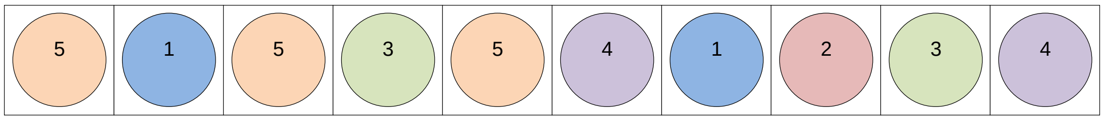
Many mesoscopic models are based on Cellular Automata (CA), which are dynamic systems that operate on a regular lattice (commonly a grid of cells), where each lattice site (cell) has one of a finite number of states. States are defined by local interaction rules, and are updated at discrete time intervals, usually in parallel. The discrete nature of cellular automata is highly suitable for computer models and simulations, as digital systems are also discrete in the spacio-temporal domain. For these reasons cellular automata are used in a diverse range of fields including, computer graphics, mathematics, physics, and biology.



(a) $t = 0$



(b) $t = 1$



(c) $t = 2$

Figure 2.2: A one-dimensional cellular automata where states are represented by numbered circles. (a) Shows the initial configuration ($t = 0$). (b) Shows the configuration at $t = 1$. (c) Shows the configuration at $t = 2$.

For clarity, a simple one-dimensional example of a cellular automata model is provided (Figure 2.2), with the following rule:

$$s(n, t) = s(n - 1, t - 1) \tag{2.2.1}$$

where $s(n, t)$ is the state of cell n at time interval t.

The rule defines that the state of a cell is equal to the state of the previous cell, during the previous time step. Therefore, given the initial configuration in Figure 2.2(a), the cellular automata will progress as demonstrated through Figures 2.2(b)-(c).

When cellular automata are used to simulate the interactions in traffic, it is typical for the traversable environment to be divided into a regular grid of cells, where each cell represents a discrete position within the environment. In this situation the state of the cell will often represent an individual entity (e.g. car, truck, bicycle, etc.), and the cell provides a mapping to the entity's relative position within the environment. Therefore, the rules of the cellular automata define how the entity traverses the environment.

Many cellular automata systems specify a maximum velocity (v_{max}), where velocity is represented as an integer, v, such that $v \in [0, v_{max}] \in \mathbb{Z}$. In these systems the maximum velocity typically represents the maximum number of cells that an entity can traverse between consecutive time steps. For example, if $v_{max} = 1$, each entity can move to one of its adjacent cells during the update process. Other CA systems incorporate greater maximum velocities ($v_{max} > 1$) to simulate entities travelling at different speeds. Furthermore, a CA is often defined to have periodic or open boundaries. A one-dimensional CA with periodic boundaries can be represented as a circular array, as the last cell is considered to be adjacent to the first cell. This principal also applies to CA of higher dimensions. It is important to note that a system with periodic boundaries will have a fixed density, as the number of entities will remain constant throughout the simulation. Models with open boundaries do not have a fixed density, as the first and last cells are disjoint.

In 1992 Nagel and Schreckenberg published pioneering work [NS92], presenting a cellular automata model to simulate the flow of freeway traffic in both open and periodic boundary conditions. Their model, now commonly known as the Nagel and Schreckenberg (NaSch) model, uses a one-dimensional array of size n to represent a single lane carriageway of n cells. In their model each cell can occupy a maximum of one vehicle. Cells are updated at discrete time intervals using the rules described below:

1. Acceleration: the vehicle will increase its velocity, v, by 1, providing that it does not exceed v_{max} and the number of cells to the next vehicle is greater than

$v + 1$

2. Deceleration: the vehicle will decrease its velocity, v, by 1, if the distance to the next vehicle is less than or equal to v cells away

3. Randomisation: a probability factor is used to randomly decrease vehicle velocities by 1, simulating natural alterations in velocity due to human behaviour or external parameters. Without this step the model would be deterministic and a pattern of motion would emerge. It is important to note that randomisation is only considered when the vehicle is in motion, i.e. $v > 0$

4. Update: each vehicle is advanced v cells along the array

Updates are performed per cell and in parallel, thus the performance of this model is scalable with multi-core/multi-processor architectures. Nagel and Schreckenberg quantitatively compared their model against real traffic, measuring the average velocity of vehicles, traffic densities, and the velocity of backward-travelling start-stop waves in congestion. The simulated results were analogous to their real-life counterparts and a qualitative comparison highlighted phenomena visualised in real life, such as the transition from laminar traffic flow (streamline) to repeating start-stop waves, when the density of traffic increased.

Rickert et al. [RNSL96] extended Nagel and Schreckenberg's model to incorporate two lanes of traffic where vehicles are permitted to change lanes based on their distance to neighbouring vehicles. Nishinari and Takahashi [NT99] use Burgers Cellular Automaton (BCA), an ultra discrete version of the Burgers equation to model traffic flow. In their model each cell can contain a number of vehicles, and movement between cells is defined by the evolution of the BCA. In Nishinari and Takahashi's model individual lane-changing rules are neglected, but the macroscopic effects of lane changes are preserved.

Although cellular approaches combine the advantages of macroscopic and microscopic models, their discrete nature is not directly suitable for animating the continuous movement of traffic, as vehicles appear to ‘pop’ between disjoint cells. Animation quality could be improved using finer discretisations, but at the cost of reducing performance.

Interactions between mixed traffic

Currently there is relatively little literature concerning the interactions between motorised vehicles (i.e. cars, buses, trucks, etc.) and non-motorised vehicles (i.e. bicycles). This is surprising considering that many governments are now encouraging civilians to use environmentally-friendly alternatives to motorised transport, particularly for short-distance journeys.

Zhao et al. [ZJG07] introduced a Combined Cellular Automata (CCA) to simulate mixed traffic flow containing both motorised vehicles (m-vehicles) and non-motorised vehicles (nm-vehicles). In this model non-motorised vehicles are defined as vehicles that are unsubstantial, more flexible, and have a lower maximum velocity, when compared to m-vehicles. To account for these dissimilarities, Zhao et al. combined two cellular automata models; the NaSch model was used to define the behaviour of m-vehicles, and the Burgers Cellular Automata model (BCA) was adopted for nm-vehicles, as it permits M nm-vehicles to occupy one cell, reflecting their smaller size. In regions where both types of vehicles are permitted, a cell can be empty, occupied by one m-vehicle, or occupied by $[1, M] \in \mathbb{Z}$ nm-vehicles.

It is important to note that a single lane in BCA is actually representative of a carriageway with M parallel lanes, as M nm-vehicles can travel simultaneously between consecutive cells. In this system microscopic lane-changing rules are neglected, but the macroscopic effects of lane changes are preserved.

The CCA model was used to investigate the interactions between mixed traffic at a bus stop, where stopping buses were permitted to enter lanes designated for nm-vehicles (nm-lanes), and nm-vehicles were permitted to temporarily enter lanes designed for m-vehicles (m-lanes) in order to pass the stopping buses. This situation is a common occurrence in developing countries like China, where there are high levels of mixed traffic. The results from the investigation show that there exists a phase transition from laminar flow to saturated flow at critical values for the insertion rate of both m-vehicles and nm-vehicles. However, a comparison with real-life data is required to validate these results.

Xie et al. [XGZL09] modelled mixed traffic flow between m-vehicles and nm-vehicles at an unsignalised intersection. In their model traffic is partitioned into m-lanes and nm-lanes, and interaction occurs at the intersection when left-turning nm-vehicles (travelling from south to west) cross the path of m-vehicles.

The authors introduced a two-dimensional car following model based on the Optimal Velocity model developed by Nakayama et al. [NHS05], and incorporate velocity difference terms to model the behaviour of both m-vehicles and nm-vehicles. The resulting simulations show that the nm-vehicles turning left have a greater effect on m-vehicles travelling north compared to m-vehicles travelling south. This is expected, as nm-vehicles will block the south-north lane, as they wait for an opportunity to cross the north-south lane. The simulation also exhibits a well-known phenomenon where groups of m-vehicles and nm-vehicles pass the intersection alternatively. However, a comparison with real traffic data is necessary to validate the accuracy of the model.

Li et al. [LGJZ09] modelled the interactions between right-turning m-vehicles crossing the path of forward moving bicycles. The two types of vehicle travel in their designated lanes, but interact at the intersection. The system combines two cellular automata models: the VDR (Velocity Dependant Randomisation) model, which is used to define the behaviour of m-vehicles, and the BCA model, which is used to define cyclist behaviour [ZJG07]. To reflect the difference between vehicle sizes, m-vehicles occupy two lattice sites and bicycles occupy one lattice site.

Based on a survey undertaken in Beijing, Li et al. [LGJZ09] found that some motorists would yield to cyclists at intersections when conflicts occurred, whereas other motorists would simply pass the intersection with complete disregard to the safety of cyclists. The results also demonstrated that the size of the gap in traffic that vehicles required before crossing at the intersection was inconsistent between vehicles. However, it was determined that if the intersection point is occupied by a m-vehicle, then there is a high probability that a m-vehicle will occupy the intersection point during the next time step. Furthermore, as the number of bicycles waiting to cross the intersection increases, the probability that the intersection point will be occupied by a bicycle during the next time step also increases.

Li et al. incorporated these characteristics of mixed traffic into their model by introducing two probability parameters, p_a and p_b, where p_a represents the probability that the intersection point is occupied by a m-vehicle during two consecutive time steps, and p_b represents the probability that the site is occupied by a bicycle during two consecutive time steps. To replicate the behaviour found in the survey, the values of p_a and p_b were defined using conflict rules, which are dependant on the state of the intersection point, s, during the previous time step $(t-1)$. The state of the intersection point at time t, $s(t)$, can take one of the following values:

- $s(t) = 0$: no occupant
- $s(t) = 1$: occupied by a m-vehicle
- $s(t) = 2$: occupied by a bicycle

For a definition of the conflict rules, the reader is referred to [LGJZ09]. The authors intend to calibrate the values of p_a and p_b in future work to ensure that the simulation is representative of mixed traffic in real-life scenarios.

An alternative technique for modelling mixed traffic was proposed by Meng et al. [MDDZ07]. In their work, the authors use a single-lane cellular automata to simulate driver behaviour for both cars and motorcycles. To reflect the smaller size of motorcycles, the single-lane automata is divided into three virtual sub lanes, permitting motorcycles to travel side-by-side in parallel. The length of each cell is set to 3.75 m (approximately the length of a motorcycle including safety-distances to neighbouring vehicles). Meng et al. suggest that cars are twice this length and therefore occupy two cells throughout the simulation. Furthermore, motorcycles are not permitted to enter the virtual left sub lane (VLSL), as the model is representative of traffic systems where motorists drive on the right-hand side of the road (e.g. China). The NaSch model is used to update the behaviours for both cars and motorcycles, as motorcycles are considered to be small cars with a lower maximum velocity. To model motorcycle lane-changing, the lane-changing rules of Nagel et al. [NWWS98] are adopted.

Meng et al. also simulated the interaction between cars and motorcycles, and analysed traffic density against flow. The results indicate that it is necessary to set a physical barrier or incorporate specific lanes to isolate the flow of cars and motorcycles, in order to improve both efficiency and safety on roads (except in some special density regime).

Summary

Macroscopic traffic models are highly-suited to large-scale traffic simulations where aggregate approximations of vehicle speed, traffic flow, and traffic density can be estimated with minimal error. In contrast, microscopic traffic models are more accurate than macroscopic solutions when modelling traffic in greater detail, for example, when simulating traffic along minor roads, intersections, crossings, and roundabouts.

However, due to their complexity, the resulting simulations are more computationally demanding. Mesoscopic traffic models can simulate traffic in high-detail and are more computationally efficient compared to microscopic alternatives. However, their discrete nature can make these models unsuitable for high-quality visualisation. After extensive research, a traffic behaviour model that is able to simulate traffic in great detail, is highly-scalable, and produces high-fidelity visualisations, could not be identified.

2.2.2 Pedestrian simulation

Rule-based simulation

Rule-based simulations use a set of rules to define the low-level interactions between individual agents within a group or crowd. These low-level actions often emerge into more complex and interesting behaviour when considering the group collectively. Higher-level behaviours such as path-planning, hunting for food, aiding an injured individual, or catching a train at a train station, can be modelled using a decision-making model with a 'divide-and-conquer' strategy, where a set of rules simplify the complexity of high-level goals into a series of lower-level behaviours.

The seminal work of Reynolds [Rey87] introduced a distributed behavioural model to simulate flocks using an evolved particle system, where each particle is represented by a 'boid' (bird-like object). Flocks are defined as a group of entities that exhibit a collective aggregate motion where members of the flock have aligned trajectories, and avoid collisions with other flock members. This definition generalises a flock to encompass many types of natural formations, including flocks of birds, schools of fish, and herds of land animals. Reynolds recognised that natural flocks are not constrained in size, as small schools of fish can consist of just a few individuals, whereas larger schools can contain millions of members. This suggests that natural flocks operate based on a perception of a localised environment. Reynolds simulated localised behaviour by adding three behavioural rules that are evaluated in a neighbouring region around each boid:

1. Collision avoidance (boids separate from their neighbours to avoid inter-boid collisions)
2. Velocity matching (boids match the velocity of their neighbours)
3. Flock centring (boids converge with their neighbours to maintain group structure)

For each boid in the flock the three behavioural rules are evaluated. Each rule requests acceleration in a particular direction in order to satisfy its requirements. To prevent conflicting requests, which may ultimately lead to a collision, the requests are prioritised such that acceleration is accumulated in this order, until the maximum acceleration value for each boid is met. To avoid collisions with static environmental objects, two techniques were explored. The first technique uses a field of repulsion forces (aligned with object surface normals) that act against the acceleration forces of approaching boids. Problems occur when boids travel towards objects, as parallel repulsion forces will not provide any side thrust to steer the boid to avoid collision. Furthermore, the technique does not allow boids to travel alongside environmental objects, such as buildings, as the object's repulsion forces will repel the boid. The second technique, 'steer-to-avoid', better simulates natural collision avoidance, as it only avoids collisions with objects in the boid's direction of travel. The technique casts a ray from the boid's local forward-vector and where it intersects an environmental object, the object's silhouette edge is extracted. A radial vector is then computed that steers the boid so that it passes the obstacle boundary. Subsequent work by Reynolds defined other steering behaviours for autonomous characters [Rey99], including, seek, flee, pursuit, evasion, arrival, obstacle avoidance, wander, path following, wall following, containment, flow field following, unaligned collision avoidance, and leader following. This work founded the popular open source toolkit, 'OpenSteer'[2], which allows fast prototyping and integration of steering behaviours into applications such as game engines.

Brogan and Hodgins [HB94, BH97] also modelled group behaviour, simulating herds of creatures with significant dynamics (dynamics that significantly affect individual motion). For example, the authors simulated a herd of one-legged robots, which were significantly dynamic, as they could only adjust their velocity when in contact with the ground. Physical properties including balance and momentum were considered when modelling these complex dynamics. In their model, each agent perceives its nearest n neighbours within a defined radius, similar to the perception model used by Reynolds in his work on crowds [Rey06]. The global desired position of each agent is computed as a weighted average of the set of desired positions, relative to each perceived neighbour. Its desired velocity is then calculated from the global desired position using a spring and damper model. Throughout the simulation the control module computes the necessary forces of each of the agent's joints to produce the desired motion.

Crowds of pedestrians have also been simulated using flocking rules [Rey06]. Due to their underlying simplicity, large crowds containing thousands of individuals can be simulated in real time. Furthermore, as group behaviour is modelled on a localised perception of the environment, this technique can benefit from the advantages of parallel processing. Reynolds simulated a crowd of 15,000 members (constrained

[2]http://opensteer.sourceforge.net/

to a two-dimensional plane), at 60 FPS on a PlayStation 3 (PS3) development system (DEH-R1015 with SDK 0.8.4), using the PSCrowd library[3]; this included three-dimensional visualisation in high definition (720p). To obtain such a high frame rate a technique called 'SkipThink' was employed that updated the behaviours of $1/10^{th}$ of the population per frame. The system was parallelised by partitioning the environment into a regular lattice of buckets, where each bucket stored a dynamic list of references to the individuals within its bounds. Behaviour was updated per bucket in parallel utilising the Synergistic Processor Units (SPUs) on the PS3.

Unlike the boids model presented by Reynolds in 1987 [Rey87], this implementation improves perceptual realism by considering only the k-nearest neighbours within a defined radius, rather than considering all agents within that region. To ensure that agents within neighbouring buckets are also considered when updating an agent's position, a condensed copy of each bucket is cached at the start of each frame, and a copy of the surrounding condensed buckets (eight in two-dimensions and 26 in three-dimensions) are sent to the SPU, along with the bucket to be updated. One limitation of the current implementation is that each bucket has a fixed size, and therefore, can only contain a certain number of individuals. This could be resolved by incorporating variable-sized buckets through the use of a more efficient space partitioning algorithm, such as Binary Space Partitioning (BSP) or kd-trees. Employing such techniques will also reduce memory constrains. Further improvements could reduce the memory limitations of the SPUs by streaming neighbouring buckets, rather than sending all neighbouring buckets simultaneously.

Musse and Thalmann [MT01] present a hierarchical rule-based model to simulate crowds in real time, with a particular focus on groups of pedestrians. The hierarchy represents crowds, groups, and individual agents, from the highest level to the lowest

[3]http://www.research.scea.com/pscrowd/

level respectively, where each level of the structure stores parameters inherited by subsequent lower levels. Groups are the most complex entity within the structure, containing information that is shared amongst the individuals within its group, including intelligence, memory, intention, and perception. Members of the same group follow a set of Interest Points (IPs) and Action Points (APs) through the environment, with each agent following a different Bézier curve between two points to introduce diversity within their trajectories. Sociological effects are also modelled, which allow agents to switch groups or to become a group leader. Three levels of autonomy are defined to control the simulation, and are presented below in a descending order of computational complexity:

1. Guided behaviour (user-interaction/external control)

2. Scripted behaviour (programmed)

3. Reactive behaviour (autonomous)

These levels of autonomy enable the system to adapt to a variety of applications, as a balance between behavioural realism, system performance, and interactivity can be established based on the application requirements and constraints. The resulting simulations exhibit diverse and complex behaviours; with a crowd of 100 agents performing scripted behaviours at 20 FPS, and with a combination of scripted, guided, and reactive behaviours at 16 FPS. Both simulations were executed on an Onyx 2 (SGI).

Ulicny and Thalmann [UT01, UT02b, UT02a] modelled high-level crowd behaviour in real time using a layered technique with hierarchical Finite State Machines (FSMs). At the highest layer complex behaviours are defined using a rule set, which considers the agent's current state and its surrounding environment. Rules consist of a selection part (who the rule affects), a conditional part (when the rule applies), and a consequential part (what is affected by the rule). In their later work [UT02b, UT02a], an intermediate layer incorporated hierarchical FSMs, which divide complex tasks into lower-level actions (which could also be hierarchies of FSMs), and a lower layer, which operates path finding and collision avoidance. This system has been used for a Virtual Reality (VR) training system that simulates up to 1,000 agents using a crude cube representation for each agent, and typically 20 agents with more-realistic deformable bodies. Both simulations were executed on an Onyx 2 (SGI).

Artificial Life (ALife) is concerned with the study and artificial reconstruction of natural living systems. Contrary to Artificial Intelligence (AI), Artificial Life uses a 'bottom-up' approach to simulate biological systems, where the low-level mechanics of living organisms are modelled. The simulation of Artificial Life on computer systems will often model the physical dynamics of organisms, their behaviours, and components of a natural life cycle, including birth, growth, reproduction, and death. Tu and Terzopoulos [TT94] defined complex behavioural rules to simulate the artificial life of autonomous fishes [4], by modelling muscle movements and hydrodynamic locomotion to animate deformable bodies based on a spring-mass system. This complex framework incorporates agent perception and intention to determine an acceptable behaviour for each fish, such as wandering, schooling, feeding, mating, and evading predators. This unscripted animation performs in real time with 10 fish, 15 food particles, and 5 static objects on a Silicon Graphics R4400 Indigo2 at 4 FPS (wire-frame mode), and exhibits schooling behaviours that have similar characteristics to behaviours exhibited in Reynolds' flocking model [Rey87]. Although the resulting behaviours are impressive, this system is heavily coupled with the animation and behaviour of fishes, and is not directly applicable to the simulation of human crowds.

Shao and Terzopoulos [ST07] used rule-based simulation to create a model of

[4]'fishes' relate to a group of fish consisting of two or more species.

artificial life, using a virtual reconstruction of New York's original Pennsylvania Station, before its demolition in 1953. Their work extends Tu and Terzopoulos' work on Artificial Fishes [TT94] and incorporates perception, motor control, behaviour, and cognitive components for each agent. To introduce realistic and high-level behaviour into the simulation, the environment was modelled using a hierarchy of maps, where each tier of the structure provides a level of abstraction. The highest level of the hierarchy stores a topological map of the environment to establish the relationship between large regions within the station, and is used for global path planning. The following tier of the hierarchy stores and maintains stationary and mobile maps, which are used for agent perception and localised path planning. The lowest tier of the structure stores specialist objects within the environment, such as ground areas, seating, ticket booths, etc. This structural representation of the environment allows agents to access the level of information required in an effective and efficient manner.

The perception of static objects is modelled using stationary maps, which are divided into a regular grids. Each grid contains a number of cells that store information about objects within its area. Agents cast a fan of rays about their forward direction (Figure 2.3), which are rasterised onto the grid map. Objects associated with the corresponding grid cells are then queried for their information.

Mobile objects (primarily other pedestrians) are perceived using the aforementioned mobile map. This map is also divided into a regular grid, but with a lower resolution (approximately $1/20^{th}$ of the resolution of stationary maps). A fan is defined about the agent's forward direction and the underlying grid cells are segmented into tiers, based on their distance to the current agent. A depth-first search is initiated through the tiers of the fan to identify other agents, which continues until either all the tiers have been searched or until n agents have been perceived. Shao and Terzopoulos used a value of 16 for n, modelling the upper-limitation of the number of

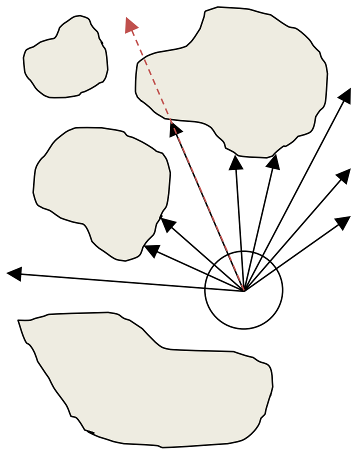


Figure 2.3: A pedestrian perceives a stationary object along its current path (red arrow) and explores alternative directions (black arrows).

individuals a human can realistically track in close proximity, based upon the limited capacity of the human nervous system [Bro65]. However, there is a discrepancy in value chosen to represent this limitation, as research by Miller [Mil56] suggests that the amount of information a human can recall from stimuli in absolute judgement is approximately seven distinct entities. Miller's research has been used by Sakuma et al. [SMK05] in a rule-based psychological crowd simulation in which agents perceive the position and speed of a maximum of seven other agents within close proximity. In Shao and Terzopoulos's model, six basic reactive behaviours are defined using local perception and are then used to operate the agent's motor controls. These behaviours include:

1. Avoid stationary objects
2. Safety in turning
3. Temporary crowding
4. Cross collision and head-on collision
5. Front safe area

6. Verify direction

The above behaviours are performed sequentially in the following permutation: 3-1-2-6-5-4, and were selected empirically through evaluation. Realistic and more-complex behaviours, including meeting and conversing with friends, sitting on a bench, watching a performance, buying a drink, and purchasing a train ticket, are realised by combining cognitive rules with an agent's internal states to determine high-level behaviours. A 'divide-and-conquer' strategy is employed to simplify complex behaviours into simple manageable instructions, which can be performed using path planning techniques and the basic reactive behaviours defined above. Shao and Terzopoulos have simulated 1,400 autonomous agents at 30 FPS (without rendering), on a 2.8 GHz Intel Xeon system with 1 GB of RAM. However, the real-time performance significantly decreases with rendering enabled; 500 pedestrians were simulated at 3.8 FPS with the same system using a NVIDIA GeForce 6800 GT AGP8X 256 MB. The resulting behaviours are highly realistic, as pedestrians appear motivated and perform high-level tasks with reasonable diversity. However, there is a lack of group activity in which individual members of the same group share common goals. Group behaviour is present in every-day life, particularly within places of public transport, where people often travel with family, friends, and co-workers. Therefore to increase realism, group behaviour would need to be incorporated. Support for groups could increase the system's real-time performance, as attributes can be shared within groups rather than stored and accessed on an individual basis, reducing both memory and processing costs. Furthermore, parallelisation techniques could be explored to combat low frame rates.

Cellular automata simulation

Burstedde et al. [BKSZ01] used a two-dimensional cellular automata to simulate pedestrian dynamics, where each agent occupies one cell and $v_{max} = 1$. In their work, the authors propose that $v_{max} = 1$ is suitable for crowd simulation, as pedestrian acceleration and deceleration times are negligible, and there is little variation in velocity for the majority of pedestrians. The authors also propose that it is valid to allow a maximum of one agent to occupy a cell, as pedestrians typically attempt to maintain a minimum distance between other pedestrians in order to avoid collisions. However, further studies into pedestrian dynamics have found that these conditions are not satisfactory for all circumstances [KKN+04]. Throughout the simulation agents are assigned a preference direction from which a 3x3 matrix, M, detailing transition preferences is constructed. The matrix stores the probability that the agent will move from its current cell to each of its eight neighbouring cells, with the centre position of the matrix storing the probability that the agent will remain stationary during the next time step. Preference directions could be pre-defined or continually assigned using the result of a path-planning algorithm.

All agents are updated in parallel and each agent uses their preference matrix to determine their desired movement. If an agent targets an unoccupied cell and no other agent has also targeted this cell, the agent will perform its desired movement. In the event that multiple agents target the same cell, one agent is selected to move based on their relative probabilities and the unselected agents remain stationary. Using these local rules, no two agents will occupy the same site to ensure collision avoidance is guaranteed. In densely crowded environments, individuals will often follow streams of pedestrians to maintain flow and reduce collisions. Burstedde et al. [BKSZ01] incorporated these long-range interactions by introducing a dynamic floor field, which is an additional grid structure that essentially overlays the cells and stores recent activity at each site. This information is used to bias transition probabilities to encourage agents to follow the recent trajectories of other pedestrians in a similar manner to chemotaxis, where organisms are attracted to certain chemicals in the environment. These trails diffuse and decay over time to reflect the dynamic nature of the simulation. A static floor field is also incorporated to provide attraction points, for example shop windows, street performances, or emergency exits (in evacuation simulations). This information remains constant throughout the simulation and can provide collision avoidance for static objects by directing agents around obstacles. Burstedde et al.'s floor fields are very efficient, as long-range pedestrian interaction is simulated using localised rules. Furthermore, the number of interactions only increases linearly with the number of agents. The resulting simulations also exhibit realistic phenomena, including the occurrence of lane formations along corridors, and alternating groups of individuals with opposing directions passing through doorways.

This model was extended by Kirchner et al. [KNS03] by incorporating the effects of friction and clogging in high-density panic scenarios. In these situations high-pressures often build-up when multiple individuals compete to occupy the same space, forming arch-like structures around exits that restrict movement and increase evacuation times. A parameter μ, where $\mu \in [0, 1]$, is introduced to model friction. Conflicts in agent desired positions are then resolved with probability $1 - \mu$, revealing the original model with no friction when $\mu = 0$.

Further investigations into cellular automata have analysed the effects of increasing the maximum velocity of pedestrians ($v_{max} > 1$), and the use of a finer lattice discretisation [KKN$^+$04]. Contrary to previous work [BKSZ01, KNS03], the authors propose that an increased maximum velocity is actually desirable, because their $v_{max} = 1$ model does not accurately simulate results found in empirical studies. Furthermore, pedestrians do walk at different speeds and agent interactions are not isotropic, i.e., an individual is influenced more by pedestrians in its direction of travel than pedestrians that are behind or beside the individual. Finer discretisations are also desirable, as they can better approximate the shape and size of geometry, which is a crucial consideration if accurate simulations are required. This is typical of evacuation simulations, where the position, shape, and spacing between complex geometry will significantly influence escape times. Kirchner et al. found that an increased maximum velocity ($v_{max} = 3$) produced results that were more symmetric with empirical studies; in particular the comparison between pedestrian flow and density. However, this condition held for simulations of pedestrian flow through narrow channels, but did not significantly influence simulations concerning the evacuation of a large room. The authors believe that an increased velocity is shown to affect simulations in a narrow channel, as the one-dimensional nature of agent trajectories form independent lanes of traffic that increase pedestrian flow.

Jiang and Wu [JW07] also simulated pedestrian behaviour in a narrow channel using a two-dimensional cellular automaton with $v_{max} > 1$. Their investigations compared pedestrian flow against density, demonstrating that with an increased maximum velocity ($v_{max} = 5$), critical densities better corresponded with empirical studies.

Simulations using a finer lattice discretisation [KKN$^+$04] showed an increase in pedestrian flow, as agents were able to move when there was space smaller than their own size. However, a finer discretisation can increase the number of local conflicts, which may ultimately lead to non-local conflicts, where groups of agents mutually block each others' movements. It is important to realise that if the maximum velocity is increased, a finer discretisation is required to keep the time scale constant. In summary, Kirchner et al.'s investigations demonstrated that the choice of lattice discretisation and maximum velocity should be carefully considered, as these parameters can significantly influence the resulting simulations.

Data-driven simulation

Realistic pedestrian behaviour can be inferred from real-life data by recording the actions of pedestrians in crowded environments and using this data to control the simulation. Lerner et al. [LCL07] tracked the trajectories and velocities of real pedestrians over time from video segments that were captured from crowded areas within an urban environment. For every frame of the video sequence, the position of each pedestrian and the positions of relevant static geometry were marked by hand to obtain pedestrian trajectories over time. A weighted configuration of the factors that may have influenced a pedestrian's path were also stored. Throughout the simulation the database of results is queried for the closest configuration to the current scenario. The returned result is then used to control the simulation in order to produce realistic behaviour. In the event that no example in the database will result in a collision-free trajectory for a pedestrian, a rule-based approach is adopted, which selects a collision-free path that best maintains the pedestrian's preceding trajectory. Lerner et al. also modelled influence functions that incorporated attenuation using a product of a Gaussian fall off. This enabled the influence of neighbouring agents to diminish with distance. In addition, they recognised that this fall off should be asymmetric in the direction perpendicular to the direction of movement, as agents behind an individual will typically have less impact on an individual's movement. Furthermore, agents moving towards an individual are assigned greater influence than agents which are travelling in a similar direction. The results produced by this technique correspond highly to behaviours exhibited in real-life examples and are generally considered to be well suited to applications that require accurate models of human behaviour. However, there are several factors limiting the real-time performance of this technique, especially those associated with database memory usage and queries. To increase the level of realism, more examples covering a broader range of scenarios would be required, but at the cost of further increasing the aforementioned constraints. A secondary consideration is the time that would be required to manually annotate video sequences. An automated approach to track pedestrian trajectories within a crowd would increase productivity, but currently remains a challenge in the field of Computer Vision [SBFF11, ELSVG08, SM06]. The model was extended by Lerner et al. [LFCCO09] to fit secondary actions such as pointing, looking around, and talking on a cell phone, by annotating agent trajectories with 'action-tags'. A stimuli-map and a validity-map are created for each action over the video sequences. The stimuli-map stores the density and relative location of stimuli that influence these actions. The validity-map stores regions where stimuli must be present prior to performing actions. These maps are stored in an action graph with directed links, where nodes are actions and edges are transitions. Throughout the simulation, each agent traverses the action graph and is assigned its next action based on a probabilistic function.

Learning the behaviour of crowds from real-life examples can reduce the memory and performance constraints of a referenced behaviour system, as a simplified model can be deduced from the input data and used to control the simulation. Lee et al [LCHL07] used a data-driven technique to learn and simulate crowd behaviour from video footage captured from an aerial view in a controlled environment. A semi-automated kernel-based tracking system was implemented to record individual pedestrian trajectories, from each pedestrian's initial position to their final position, and vice-versa. A linear blend is performed between the two trajectories to produce a more reliable result, and a Gaussian filter is applied to remove high-frequency oscillations. Trajectories that have drifted from the target can be improved by manually specifying the location of the target at user-defined key frames. Individual behaviour is modelled on an agent's perception of its local environment through state-action samples that describe an agent's action for a given state. States are modelled on an empirically weighted combination of the agent's speed, neighbourhood formation, position and orientation relative to pivots (objects that alter agent behaviour), and the agent's intended direction of travel. Each component's weight is determined by its influence on the current behaviour. State-action samples are extracted from the video, and principal component analysis (PCA) is performed on the states to represent them in a lower dimension for improved storage efficiency. Throughout the simulation, agents query the behaviour model to obtain an appropriate action for their current state. However, due to the discrete mapping of input states to output actions, large variations in output can occur when similar inputs are used. To reduce this undesirable effect, Lee et al. use a locally weighted linear regression technique with the k-nearest state-action samples to obtain an output which better represents the aggregate action of a set of similar inputs. To reduce the computational expense associated with finding the k-nearest neighbours, the state-action samples are stored in a kd-tree. This technique, like other data-driven techniques, simulates realistic human behaviour, but also has the ability to exhibit a diverse range of behaviours from video footage implicitly. Lee et al. also conclude that the system can imitate other crowd simulation techniques, such as Reynolds' rule-based flocking algorithm [Rey87]. However, a disadvantage of this system is its applicability to real-time simulation, as memory usage increases linearly with the size of the training data. Furthermore, the computational cost associated with the neighbourhood search limits performance.

Metoyer and Hodgins [MH04] describe agent behaviour that is learned from user-specified situation-specific preferences that are provided during an initial training stage. During the simulation, agents are instructed to follow a user-defined path, in which they are allowed to deviate to avoid collisions using a reactive behaviour model that extends from Helbing's work on social forces [HM95]. A data-driven model is incorporated to improve steering behaviours. During the training phase, a two-dimensional simulation is executed that highlights potential collisions, allowing the user to stop the simulation and select an appropriate navigation rule to maintain a collision-free path. Metoyer and Hodgins define five navigation primitives; walking around-left, walking around-right, yielding, cutting-in-front, and no-action. Upon selection of one of these primitives, the system records the local information that could have influenced the decision and the primitive that was selected. If a potential collision is detected during the real simulation, the system considers the agent's surrounding features and consults the training data to find the most suitable navigation manoeuvre to perform. The resulting force is then combined with the reactive social force to produce more-natural steering behaviours. Through evaluation, Metoyer and Hodgins highlight the successes of their technique with 'reasonably' crowded environments. However, they concluded that as the scene becomes more congested, fewer navigational options exist and behaviour becomes more reactive and less strategic. One advantage of this system is that it provides an abstract level of crowd authoring to system designers; however, the level of authoring is limited in variety by the number of navigational rules defined, and is limited in quality by the variation and number of examples in the training data.

Lai et al. [LCF05] used data-driven animation to simulate crowds in their work on Group Motion Graphs (GMGs), which extends the concept of directed motion graphs typically used for character animation to create a data-structure that encodes acceptable transitions between a set of crowd configurations. From an input sequence, individual frames are sorted into n groups of similar configuration using k-means clustering with a Hausdorff distance metric. The Hausdorff metric is used to measure the difference between configurations, rather than an Euclidean distance metric, because it better represents the formation shape by penalising configurations with agents in one frame that are remote from agents in another frame. The nodes of the graph represent different crowd configurations, with each edge of the graph providing a transformation sequence between two configurations. To introduce variation, a family of arc edges are constructed between each pair of nodes, from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$ radians, at $\frac{\pi}{12}$ radian intervals. To enhance the realism of the simulation, transition statistics are collected from the input sequence and are used as probabilities for traversing between graph nodes. Lai et al. constrain GMGs to simulate groups that walk randomly, follow paths, and stay within a defined region, by extending Reynolds flocking algorithm [Rey87] with the addition of path-following, configuration matching, and target orientation rules. GMGs provide an efficient structure to control and simulate crowds in real time. However, the presented technique is limited in realism, as it cannot model the interactions between different groups. Furthermore, groups cannot interact with objects in the scene, unless a similar situation is present in the recorded data. Lai et al. propose switching from GMGs to a rule-based behavioural model to overcome the above difficulties.

Paris et al. [PPD07] define a rule-based behavioural model that predicts neighbouring agents' movements to determine future trajectories. This rule-based model is calibrated using data obtained from motion-captured examples of crowds to produce realistic behaviour. Motion-capture data was preferred over other inputs, as movements and trajectories extracted from motion capture are highly accurate and clearly convey the intentions of each individual, allowing a deeper analysis to be performed. However, realism is often compromised when using motion capture, as interactions are reconstructed in a controlled environment with 'actors', and therefore the resulting data may not truly represent movements exhibited in real life situations. To obtain the necessary data, two phenomenon were reconstructed; to configure the model, the interaction of two pedestrians with intersecting trajectories, and to test the configuration, the behaviour of a small crowd passing through a narrow passage. During the configuration stage, Paris et al. found that reactions were observable when the minimum predicted distance between agents was less than 0.5 m. Anticipation of intersections were realised several seconds before the potential collision, and reactions to potential collisions involved adaptations to both speed and orientation. When no collision was predicted, no reaction occurred in the captured data. Using their findings, Paris et al. tuned parameters in their rule-based system, so that their simulation output corresponded with the motion-captured data. The configured model was then compared with the recorded behaviour of a small crowd passing through a narrow passage. It was concluded that the rule-based model produced similar crowd characteristics to those found in the examples, including lane-formations in corridors, alternating groups of individuals with opposing directions at x-crossings, and large changes in speed when visibility was restricted by obstacles.

Peters et al. [PEM08] used a data-driven approach to evaluate the perceptual plausibility of pedestrian orientations in both ‘open’ and ‘constrictive’ environments. Pedestrian orientations were manually annotated from images, and were quantised to one of eight uniformly separated directions. The original photographs were virtually reconstructed by positioning avatars at the relevant locations in the scene, and each pedestrian was rotated to match its corresponding quantised orientation. A variety of images were then reconstructed, with half of the images modified using random or contextual rules to alter pedestrian orientation. From the data set, participants were asked to determine which of the images represented real-life data, and which images had modified pedestrian orientations. The results indicated that humans are typically able to determine which of the images had been fabricated. However, the results also show that this differentiation is more difficult to determine when the orientations were assigned using contextual rules, compared with random assignment.

This investigation was extended in subsequent work [EPO08] to evaluate the perceptual plausibility of pedestrian positions, and was further extended in [EP09] to incorporate the perception of groups of pedestrians. Perceptual investigations are useful when analysing simulations, as they can be used to validate the realism of behaviour models, and could even provide a platform for comparing simulation techniques, based on their ability to reconstruct real-life phenomenon as exhibited in the recorded data. However, future work in this area will need to evaluate pedestrian behaviour in the spatiotemporal-domain to obtain a more-accurate perception model.

Summary

Rule-based crowd simulations can produce a variety of behaviours at moderately interactive frame rates ($\geq$ 20 FPS), with more-complex behaviours demonstrated through the use of hierarchical structures [UT02b, UT02a] and the incorporation of sociological effects [MT01]. Furthermore, it has been shown that real-time performance is feasible ($\geq$ 30 FPS), through the adoption of parallel processing techniques [Rey06]. However, as with any rule-based system, the realism of the resulting behaviours are limited to the complexity and accuracy of the rules themselves. Additionally, the behaviour of the crowd has to be defined explicitly by the rules, which makes it difficult to author the overall characteristics of a crowd.

It is typical for simulations based on cellular automata for the traversable environment to be divided into a regular grid of cells, where each cell represents a discrete position within the environment that a pedestrian can occupy. Due to their discrete nature, these systems are usually highly parallelisable and are therefore suitable for real-time applications. However, as with rule-based systems, the realism of the resulting behaviours are limited to the complexity and accuracy of the rules themselves. Additionally, coarse cell discretisations reduce both realism and accuracy of the model.

A diverse range of data-driven techniques for crowd simulation have been reviewed, using inputs from both video and motion capture. Referenced behaviour systems can achieve high levels of realism, but have limited suitability for real-time applications, due to the performance costs associated with database memory usage and queries. Computational performance could be improved using a learned behaviour model [LCHL07], by extracting parameters from the data. However, performance enhancement techniques must be adopted to work with large populations in real time. Group Motion Graphs were also reviewed, but due to their interaction limitations, they are considered unsuitable for accurate pedestrian simulation.

2.2.3 Interactions between vehicles and pedestrians

Interactions between vehicles and pedestrians have been modelled by Jiang and Wu, who used a two-dimensional cellular automaton to simulate the interaction between pedestrians and a low-speed vehicle (e.g. bicycle, tricycle, or barrow) in a narrow channel [JW06b]. At each time step of the simulation, an agent can either remain stationary or move to an available adjacent site in its forward direction (east or west), or laterally (north or south). This direction is selected using transition probabilities that are assigned according to the availability of the corresponding sites. The system also incorporates a drift factor, favouring forward movement whenever possible. Both pedestrians and vehicles have a maximum velocity of 1 ($v_{max} = 1$). The results suggest that when the vehicle travels in an opposite direction to the pedestrian flow, its average velocity will increase when it moves along the centre of the channel. However, when travelling in the same direction as the pedestrian flow, the vehicle's average velocity will increase whilst travelling along-side the walls of the channel. Unfortunately, a comparison with real-life data was absent in this work and it is therefore difficult to validate the accuracy of these findings.

The authors refined the model in later work [JW06a], permitting the vehicle to travel at variable speeds ($v_{max} > 1$). Regions of influence were defined around the vehicle that alter the transition probabilities of the underlying automata to bias agent movement away from the path of the vehicle. The dimensions of the regions of influence increase with the velocity of the vehicle.

In recent years, there has been significant research on collisions between vehicles and pedestrians. In 2005, Lee and Abdel investigated collisions between vehicles and pedestrians at intersections in Florida [LAA05], studying factors that impacted collision frequency. Xu et al. [XLLZ09] developed a model to reconstruct the interactions between a pedestrian and a vehicle, in collisions where a pedestrian's head impacts a vehicle's windscreen. The model estimates the velocity of the vehicle from variables describing the dimensions of the windscreen, the location of the impact point, the mass of the vehicle and pedestrian, and the mass and radius of the pedestrian's head. The accuracy of the model was validated using data from real-world accidents, and the results show that the model is accurate for low-speed collisions, but accuracy of the model diminishes with vehicle speed. Furthermore, a system that automatically analyses the interactions between vehicles and pedestrians at crossings was developed by Ismail et al. [ISSL09]. In their system, feature-based tracking is used to monitor the trajectories of both pedestrians and vehicles. Conflict indicators are calculated and important interactions are reported for further examination by human observers.

Summary

The interactions between vehicles and pedestrians have been studied in the literature, with a significant focus on vehicle-pedestrian collisions [LAA05, XLLZ09, XLC^{+}11, ISSL09]. The majority of reviewed work is primarily concerned with the analysis of recorded incidents, where reconstruction or statistical evaluation of these events can be used to improve road safety. There is little work that focuses on the development of behavioural models that simulate the interactions between vehicles and pedestrians, especially for scenarios where interaction does not result in an incident. Furthermore, a significant volume of work focuses on interactions at intersections or crossings where pedestrians typically adhere to crossing rules [LAA05, ISSL09]. Furthermore, the few models that do simulate a greater level of pedestrian freedom, only demonstrate very basic behaviours [JW06b, JW06a]. After an extensive review of the literature, there appears to be an absence of research on the development of behaviour models that simulate general interactions between vehicles and pedestrians in regions where pedestrian behaviour is less defined (high-streets, marketplaces, minor roads, etc.).

2.3 Summary

In this chapter an overview of the literature in the field has been presented, with a focus on significant contributions and current state-of-the-art.

A variety of techniques for generating road networks in virtual environments have been explored. Procedural methods dominate the literature, and are able to produce extensive and complex road networks with minimal input. However, resulting networks appear to follow conventional city centre design patterns and require significant fine-tuning to obtain non-standard or customised layouts. Sketch-based systems provide an alternative to procedural methods and offer a greater level of control in the design process, but current techniques are limited to extremely flat environments.

Computational models used to simulate the behaviour of drivers and pedestrians were also studied. Macroscopic traffic models are highly-suited to large-scale traffic simulations where aggregate approximations of vehicle speed, traffic flow, and traffic density can be estimated with minimal error. In contrast, microscopic traffic models provide greater accuracy, but are typically more computationally demanding. Mesoscopic models are able simulate traffic in high-detail and are more computationally efficient compared to microscopic alternatives. However, their discrete nature can make these models unsuitable for high-quality visualisation.

For pedestrian simulation, rule-based crowd simulations can produce a variety of behaviours at moderately interactive frame rates, with more-complex behaviours demonstrated through the use of hierarchical structures and the incorporation of sociological effects. However, as with any rule-based system, the realism of the resulting behaviours are limited to the complexity and accuracy of the rules themselves. Additionally, the behaviour of the crowd has to be defined explicitly by the rules, which makes it difficult to author the overall characteristics of a crowd. Simulations based on cellular automata divide the traversable environment into a regular grid of cells, where each cell represents a discrete position within the environment that a pedestrian can occupy. Due to their discrete nature, these systems are usually highly parallelisable and therefore suitable for real-time applications. However, as with rule-based systems, the realism of the resulting behaviours are limited to the complexity and accuracy of the rules themselves. Additionally, coarse cell discretisations reduce both realism and accuracy of the model. Data-driven referenced behaviour systems can achieve high levels of realism, but have limited suitability for real-time applications, due to the costs associated with memory and query performance. Computational performance could be improved using a learned behaviour model by extracting parameters from the data. However, performance will need to be improved for the technique to work with large populations in real time.

The interactions between vehicles and pedestrians were also studied. The majority of work reviewed is primarily concerned with the analysis of recorded incidents, where reconstruction or statistical evaluation of these events can be used to improve road safety. There is little work that focuses on the development of behavioural models that simulate the interactions between vehicles and pedestrians, especially for scenarios where interaction does not result in an incident. Furthermore, a significant volume of work focuses on interactions at intersections or crossings where pedestrians typically adhere to crossing rules, but there appears to be an absence of research on developing behaviour models that simulate general interactions between vehicles and pedestrians in regions where pedestrian behaviour is less defined.

This research has highlighted several significant challenges in the field. Therefore, two significant challenges related to the design and simulation of traffic networks for use in virtual environments are identified:

1. The development of intuitive techniques to assist the design and construction of high-fidelity three-dimensional road networks for use in both urban and rural virtual environments.

2. The implementation of computational models to accurately simulate the behaviour of drivers and pedestrians in transportation networks, in real time.

These challenges are investigated throughout the remainder of this thesis.

Chapter 3

Generating road networks from digital map data

To create traffic simulations of high visual fidelity, roads and road networks need to be designed and modelled in great detail to conform with the governing rules and regulations of highway design. Roads could be created manually, but this can become a time-consuming and laborious process when modelling large-scale networks. Therefore, automated techniques for generating road networks efficiently and effectively, without any prior user knowledge of road design principles and practices are highly desirable in both urban-planning and entertainment industries.

In this chapter, a novel technique to automatically generate road networks from digital map data is presented. The input to the algorithm is a connected set of vertices and edges of an existing road network. Topology is determined using a set of rules that considers the number of connections at each vertex and angle between connecting edges. A road graph suitable for a mesoscopic traffic behaviour model is then constructed by segmenting each road into a series of cells that can be used as waypoints during a simulation. The performance of the algorithm is evaluated by comparing the topologies of generated road networks with their real-world counterparts. The results indicate that the algorithm is able to correctly determine the layout of over 86% of all junctions in each of the real-world test scenes. The steps of the algorithm are summarised below:

1. Identify junctions
2. Identify individual roads
3. Distribute cells

3.1 Generating the road graph

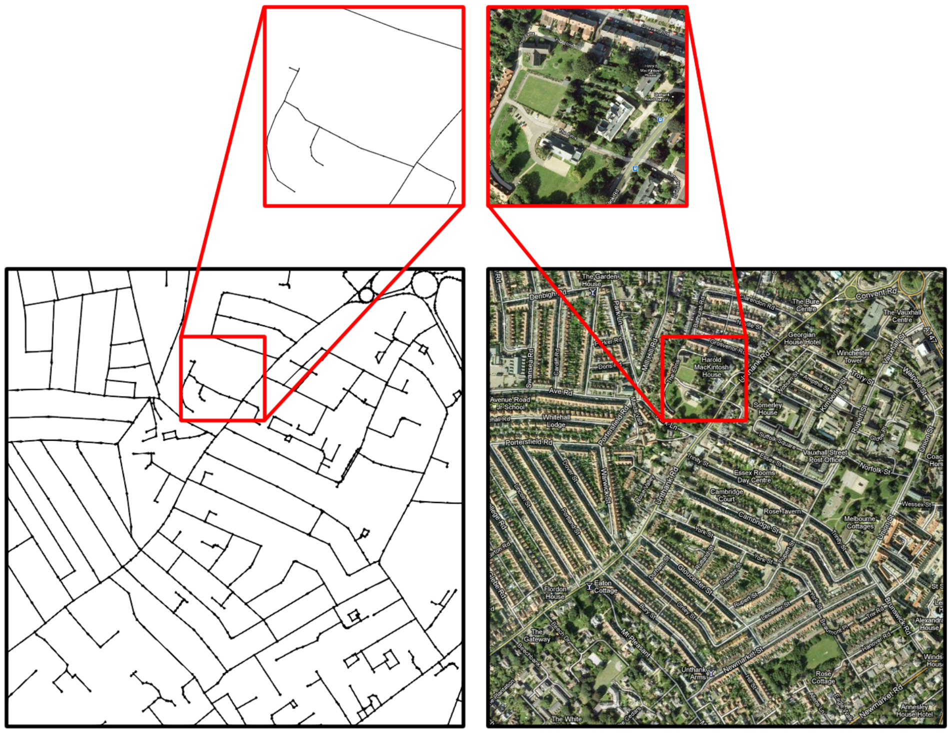
This section details the steps for generating road networks from digital map data.

3.1.1 Identify junctions

To generate a virtual representation of a real-world road network, road centre line information needs to be extracted from digital map data. To create a generic and flexible system, the input to the algorithm is a set of vertices and connecting edges that represents the topology of a road network (Figure 3.1). The algorithm identifies and classifies junctions, such that vertices with three or more connecting edges are stored as *road junctions*, and vertices with only one connecting edge are stored as *terminal junctions*, which are junctions that provide either entry onto or exit from the road network (Figure 3.2).

3.1.2 Identify individual roads

A road is defined as a set of connecting edges that start and end at a road or terminal junction. However, determining the extent of an individual road is a challenging problem, as there may be several junctions along any single road. The proposed algorithm compares the dot products between normalised edge vectors at junctions to determine the extent of individual roads. If the greatest dot product (λ) between a road's normalised edge vector ($\widehat{v_a}$) and the other normalised edge vectors at a junction



(a) (b)

Figure 3.1: (a) Road centre line data for a 1 km^2 tile area of Norwich, United Kingdom. The data is represented as a set of vertices and connecting edges. At the boundaries of the tile there are some isolated edges, created from the tile segmentation. (b) A hybrid satellite image of the same region. There are some discrepancies between the sets of data where mapping information is outdated, as highlighted above. Imagery ©2013 DigitalGlobe, Getmapping Plc, Infoterra Ltd & Bluesky, and The GeoInformation Group. Map data ©2013 Google.

($\widehat{v_1}$ to $\widehat{v_n}$), is greater than or equal to $\cos(\vartheta)$, then the edge that is associated with the greatest dot product is considered to be continuous with the road. Otherwise, the road is considered to terminate at the junction (Figure 3.3). In this system, the default value for $\vartheta = \frac{2\pi}{9}$ radians, a value that was derived through experimentation and analysis. This parameter can be adjusted by the user to obtain the required results. A formal definition of λ is provided:

$$\lambda = \max((\widehat{v_a} \cdot \widehat{v_1}), (\widehat{v_a} \cdot \widehat{v_2}), ..., (\widehat{v_a} \cdot \widehat{v_n}))$$

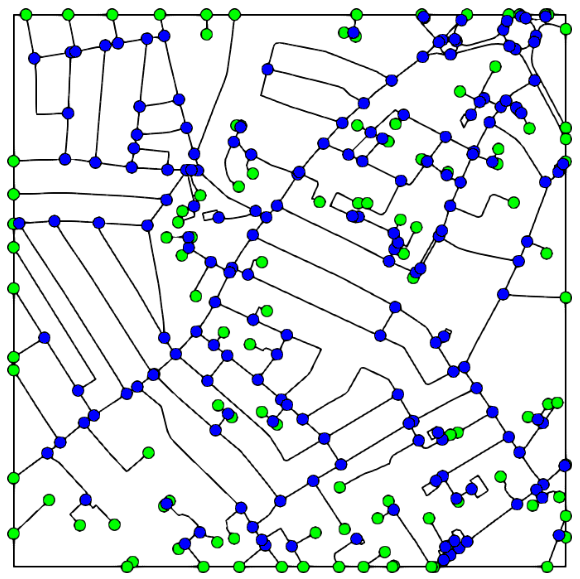


Figure 3.2: Identified junctions on the road graph. Blue points indicate junctions with three or more connecting edges, and green points indicate entry/exit junctions.

Using the continuation rules established above, individual roads are identified by the algorithm (Figure 3.4). This is an iterative process that first selects an unprocessed terminal junction, and traverses the road graph from the selected junction, following unvisited edges until the end of the road is identified. Once all terminal junctions have been processed, the algorithm is repeated, selecting one of the junctions with the lowest number of unvisited edges. This process continues until all edges have been traversed.

3.1.3 Distribute cells

The road graph is made suitable for a mesoscopic traffic behaviour model by distributing discrete cells throughout the road network where vehicles are permitted to occupy at discrete time intervals. Ideally the size of each cell would be constant throughout the road graph. However, as the road graph can be derived from real-world data, there is no guarantee that the length of a road will be a multiple of the desired cell length. To resolve this issue an approach used by Sewall et al. is adopted [SWML10],

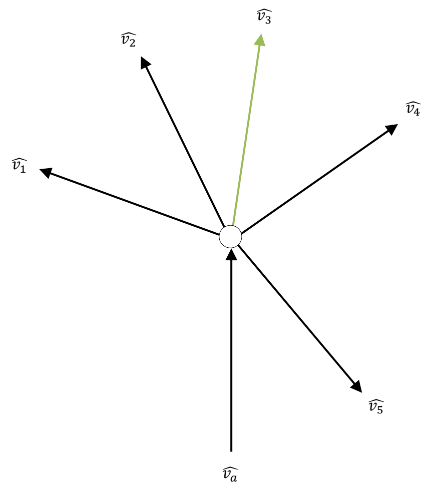


Figure 3.3: If the greatest dot product (λ) between a road's normalised edge vector ($\widehat{v_a}$) and the other normalised edge vectors at a junction ($\widehat{v_1}$ to $\widehat{v_n}$), is greater than or equal to $\cos(\vartheta)$, then the edge that is associated with the greatest dot product is considered to be continuous with the road (green edge). Otherwise, the road is considered to terminate at the junction.

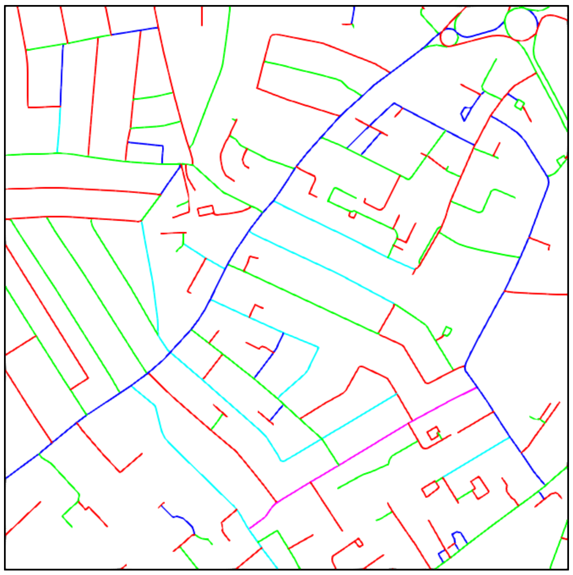
where a target cell length Δx is defined. Cells are then positioned uniformly along each road between junctions (road links), rather than over an entire road, as this ensures that a cell is placed at every junction. For a link i, of length L_i, the number of cells, N_i, and the cell length, Δx_i, is defined in Equations 3.1.1 and 3.1.2. In this system, $\Delta x = 6$ metres, which is approximately the length of an average car, including safe distances.

$$N_i = \left\lfloor \frac{L_i}{\Delta x} \right\rfloor, L_i > 0, \Delta x > 0 \tag{3.1.1}$$

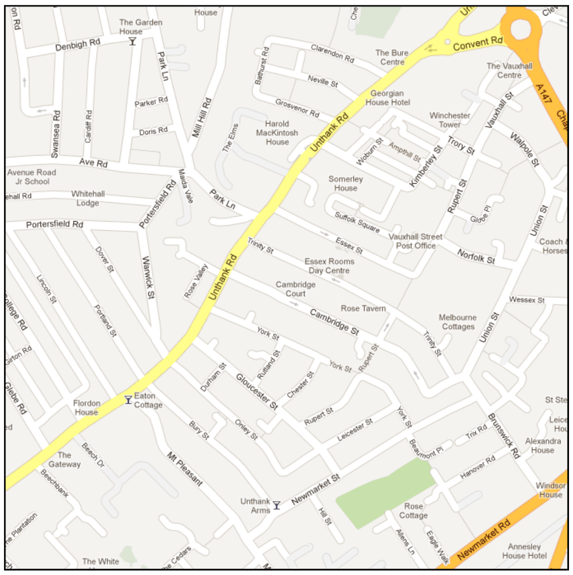
$$\Delta x_i = \frac{L_i}{N_i} \tag{3.1.2}$$

3.2 Performance

Performance of the algorithm was evaluated using a set of 10 digital maps representing 1 km^2 areas of the United Kingdom (Figure 3.5). These maps were exported directly



(a)



(b)

Figure 3.4: (a) Roads identified by the algorithm. The extent of individual roads are illustrated using different colours. (b) Digital map information of the same region. Map data ©2013 Google.

from OpenStreetMap in XML format, and were parsed to extract sets of vertices and connecting edges representing the topologies of their respective road networks. The algorithm was written in C++ and the time required to generate the topology of the each test scene was less than 11.16 ms. Performance was measured on a machine with an Intel Core i5 3.2 GHz processor with 4 GB RAM. To quantifiably evaluate the performance of the algorithm, the following goal was defined to help measure success:

> *Produce an algorithm that is able to replicate the topology of a real-world road network, given a set of vertices and connecting edges representing the network as input.*

To evaluate the algorithm, the topologies of generated road networks are compared with their real-world counterparts using Google Street View. As junctions are automatically inferred from the map data, it is assumed that the positioning of junctions are correct in the road graph. However, junction layout is not explicitly represented in the input data and therefore the layout of each junction in the road

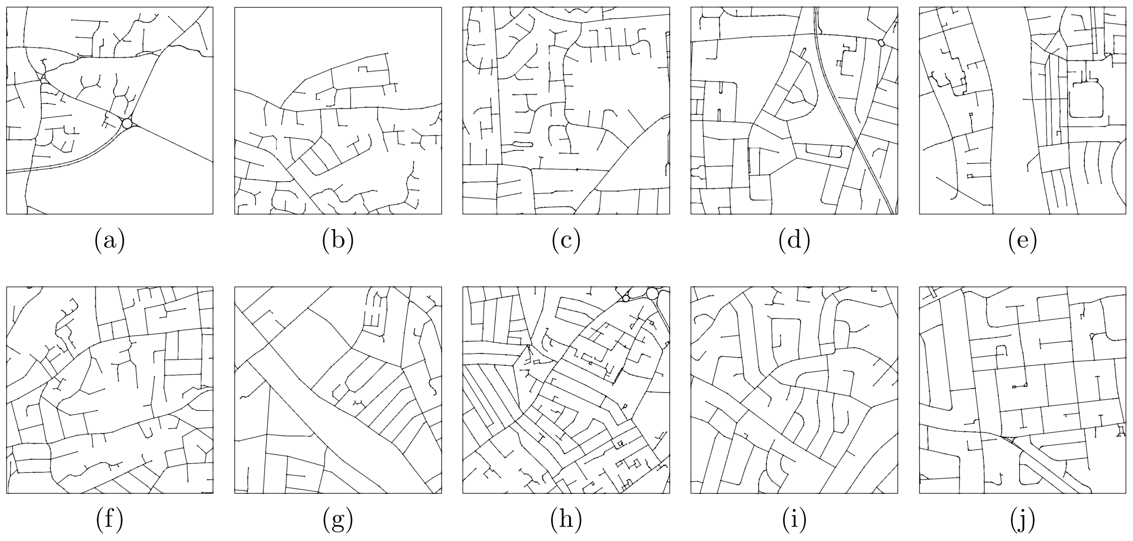


Figure 3.5: The set of 10 1 km^2 area maps used to evaluate the performance of the algorithm.

graph is compared with its real-world counterpart to determine system performance. The results of the evaluation are presented in Table 3.1.

The results demonstrate that the algorithm was able to correctly determine the layout of over 86% of all junctions in each of the 10 test scenes. By definition, there is only one possible topology for a terminal junction, as it does not connect to any other roads in the network. Therefore, we also evaluate the performance of the algorithm on non-terminal junctions to provide a better representation of accuracy. The results show that over 75% of all non-terminal junctions in each of the 10 test scenes were classified correctly. Figure 3.6 provides a visual representation of the performance of the algorithm on scene h, and shows that a large number of the incorrectly classified junctions were part of roundabouts or junctions where 4 edges intersected. To reduce the number of errors, these cases should be processed separately. Through further analysis, it was found that some of the incorrect classifications could be attributed to the inaccuracies of the input data. For example, Figure 3.7 highlights inconsistencies between the input data derived from OpenStreetMap and the same

ID	No. junctions		Classification (total no. junctions)			Accuracy %	
	Non-terminal	Total	Correct	Incorrect	Indeterminable	Non-terminal	Total
a	76	136	120	16	0	78.95	88.24
b	59	124	118	6	0	89.83	95.16
c	86	192	185	6	1	92.94	96.86
d	96	149	139	10	0	89.58	93.29
e	92	154	131	14	9	83.13	90.34
f	101	193	172	18	3	81.62	90.53
g	61	111	96	15	0	75.41	86.49
h	182	294	243	25	26	83.97	90.67
i	71	131	127	4	0	94.37	96.95
j	88	139	120	12	7	85.19	90.91

Table 3.1: The results detail the accuracy of the algorithm for junction layout classification over the 10 test scenes shown in Figure 3.5. The number of correct and incorrect classifications are shown, as well as the number of junctions where it was not feasible to determine whether the classification was accurate. This typically occurred where images were not available in Google Street View (private land, car parks, etc.).

region shown in Google Maps. In the vertex-edge data derived from OpenStreetMap, mini-roundabouts are represented as T-junctions, and were therefore processed accordingly.

A comparison of the topology of individual roads with their real-world counterparts, was also considered. However, initial observations of real-world data demonstrates that road names are not necessarily an accurate indication of topology. For example, Figure 3.8 highlights a scenario where two separate roads meet at an arbitrary point. However, through analysis at street level, it is obvious that these roads should be considered as one continuous road.

3.3 Summary

This chapter presented a novel technique to automatically generate road networks from digital map data, where the input to the algorithm is a connected set of vertices

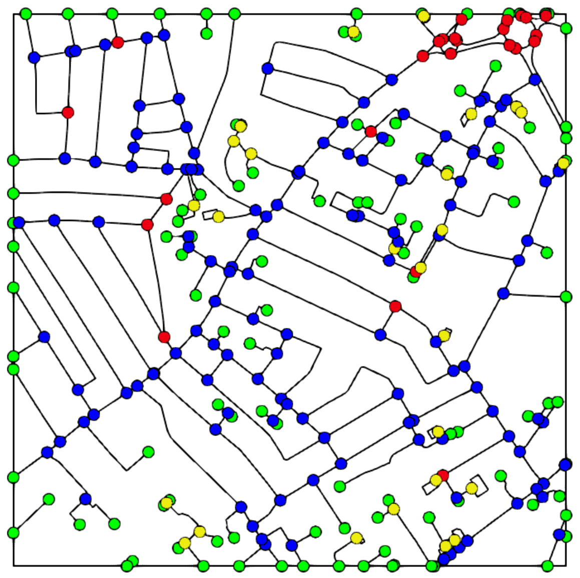
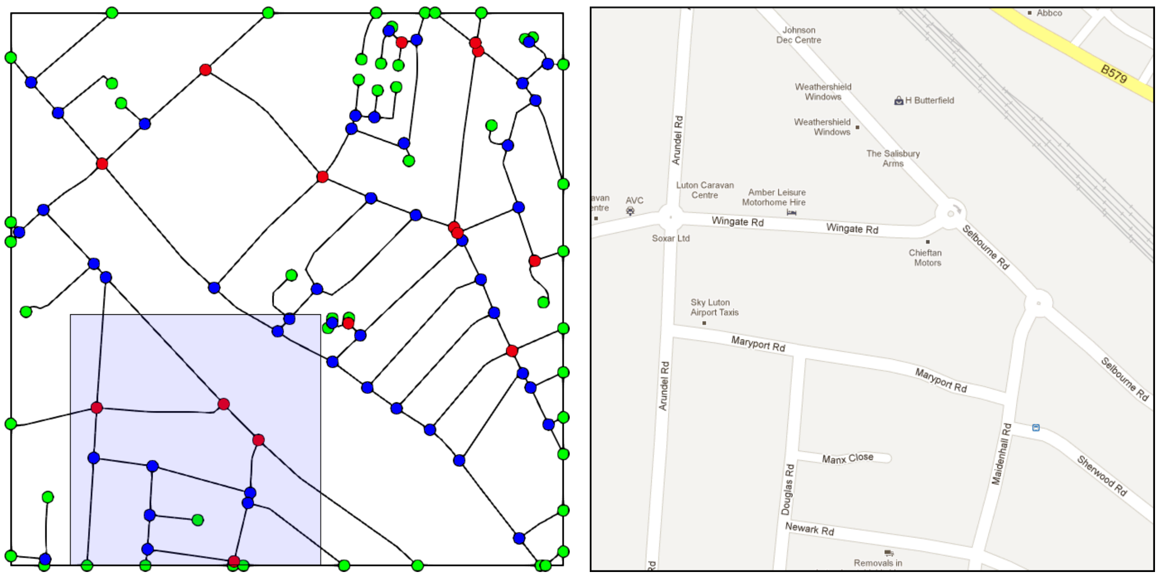


Figure 3.6: A visual representation of the performance of the algorithm on a 1 km^2 area of Norwich, United Kingdom. Terminal junctions are illustrated with green points, correctly classified non-terminal junctions are shown as blue points, incorrectly classified non-terminal junctions are shown as red points, and indeterminable classifications are shown as yellow points.

and edges of an existing road network. The algorithm identifies and classifies junctions based on the number of connecting edges at each vertex, and network topology is determined by comparing the angle between connecting edges. Individual roads are then segmented into a series of cells that can be used as waypoints throughout a traffic simulation. The performance of the algorithm was evaluated by comparing the topology of generated road networks with their real-world counterparts. The results demonstrated that the algorithm was able to correctly determine the layout of over 86% of all junctions in each of the test scenes. Therefore, the presented algorithm provides an accurate method to automatically generate virtual representations of real-world road networks for use in virtual environments, which could be used to reduce the development times and costs typically associated with virtual content creation. To improve the accuracy of the algorithm, it was considered that complex



(a) (b)

Figure 3.7: (a) A visual representation of the performance of the algorithm on a 1 km^2 area of Luton, United Kingdom. Terminal junctions are illustrated with green points, correctly classified non-terminal junctions are shown as blue points, and incorrectly classified non-terminal junctions are shown as red points. (b) A hybrid satellite image of the region highlighted in blue. A comparison between the images shows that there are inconsistencies between the two sets of data. In the vertex-edge data derived from OpenStreetMap (a), mini-roundabouts are represented as T-junctions. Map data ©2013 Google.

road structures, such as roundabouts, should be processed separately. These structures could be identified using image-processing techniques. For example, a Hough Transform could be used to extract the size and location of rounadbouts in the environment [DH72]. However, despite inaccuracies, it is considered that this technique would still be benefical for content creation, even with reduced performance, as the algorithm would produce an initial framework for further development.

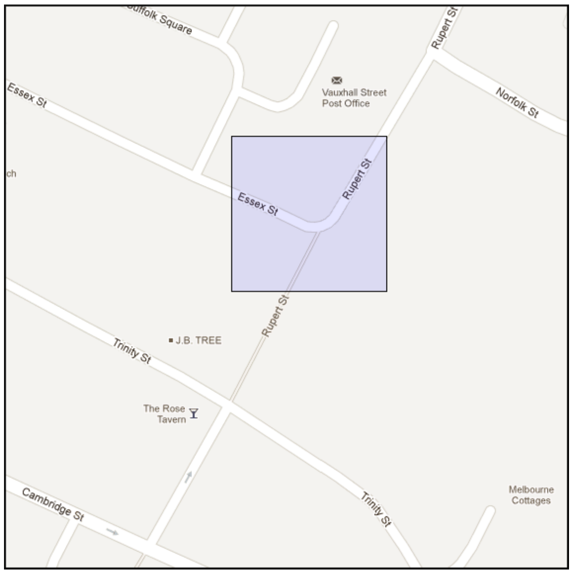


Figure 3.8: Road names are not necessarily an accurate indication of road topology. The area highlighted in blue shows what appears to be two separate roads (Essex Street and Rupert Street). However, through analysis at street level, it is obvious that these roads should be considered as one continuous road. Map data ©2013 Google.

Chapter 4

Generating road networks from sketches

To design and create road networks for virtual environments, each component part needs to be designed and modelled in great detail. In addition, a combination of artistic and modelling skills are often necessary to complete this time-demanding and costly process. Roads can be created manually, but this can become a time-consuming and laborious process when modelling large-scale networks. Therefore, automated techniques for generating road networks efficiently and effectively, is highly desirable in both urban-planning and entertainment industries. There are a multitude of commercial products available to aid the development process (Table 1.1). However, these packages typically use unintuitive and repetitive 'point-and-click' input methodologies to generate roads.

In recent years there has been an abundance of research on sketch-based modelling techniques with applications in a diverse range of disciplines. Systems that are able to utilise the intuitive feel of conventional drawing methodologies are becoming a popular alternative to those that use more traditional input techniques, primarily due to their increased usability and resulting productivity. Recent advances in this field have provided the motivation and need to develop techniques that allow novice and expert users, from both the entertainment and urban planning industries, to create road networks in urban and rural virtual environments with relative ease through assisted sketch-based tools.

In this chapter a novel sketch-based tool to semi-automate the design, creation, and visualisation of road networks for virtual environments is presented. The tool is guided by input sketches and a combination of prioritised constraints, such as the curvature of roads, their inclination, and the volume of ground that would be displaced during construction. The technique extends previous work in the field by constraining the geometric properties of roads in three-dimensions to generate road networks across both flat and undulating terrain. Constraints can be tailored to generate roads that exhibit particular characteristics, such as roads with little inclination that cut through the environment, and roads with high curvature that meander across the terrain. Furthermore, this work extends previous research on sketch-based systems for road design [ALD11], presenting a significant and novel contribution that both complements and advances previous work. The concept of 'Influence Regions' is introduced, which are user-specified areas of the environment that influence the path of generated roads. These regions provide guidance throughout the road design process and can be tailored to either attract or repel roads to/from certain obstacles or designated areas, such as forestation, listed buildings, marshland, etc. A user study is conducted to evaluate the usability of the system and to evaluate the quality of roads generated in a diverse range of scenarios. The results indicate that the system is both user-friendly and able to produce roads that are true to the user's intention. The steps of the algorithm are summarised below:

1. Sample the user's sketch
2. Identify connection constraints
3. Generate a clothoid curve

4. Constrain the curve in the xz-plane

5. Project the curve onto the terrain

6. Constrain the curve in the y-dimension

7. Construct the road geometry

4.1 Road generation

This section presents the steps for generating roads from user sketches.

4.1.1 Sample the user's sketch

To generate a road in the system, the user sketches the desired path of the road onto a three-dimensional terrain. Whilst sketching, each screen-space coordinate is projected onto the terrain to determine its position in world-space. For visual purposes a line is rendered through these points to provide instantaneous feedback to the user.

The application currently uses the mouse to provide sketch input. However, the system has been designed to be ported to sketch-based devices with relative ease. When capturing the sketch, the mouse is sampled at the highest frequency available. On completion, the algorithm iterates through the list of connected points and removes any point with a distance less than a pre-determined sample threshold, α, from the previous point. This significantly reduces the number of samples and ensures that they are approximately uniformly spaced along the original sketch. As a further optimisation, redundancy could be reduced by representing curved segments with a larger number of samples when compared with straight segments. The mouse was sampled at a frequency of 125 Hz, and a value of 5 metres was initially selected for α, based on the position of a fixed-height camera relative to the terrain. However, since the system evaluation presented in Section 4.3, it is recommended that α is defined in pixels to work at various levels of detail. A value of 18 pixels (at approximately 83 pixels per inch) produces similar results to the initial parameter value.

4.1.2 Identify connection constraints

Once the sketch has been sampled, its connectivity with the other roads in the network is determined. When a connection is identified, a 'Connection Constraint' is created and attached to both the sketch and the road(s) that the sketch connects to. These constraints describe the properties of the connection, such as the connection type and its location along each of the roads.

Initially, it is established whether the user has specified a 'ring' or 'loop' road where its start and end points coincide, by comparing the Euclidean distance between the endpoints of the sketch with a connection threshold $T = 25$ pixels, set as a default value based on a display device with approximately 83 pixels per inch. This parameter can be adjusted by the user in line with their targeting accuracy. Future work could involve developing a calibration tool to automatically set this parameter based on targeting exercises. If the distance between the endpoints is less than the threshold, a 'Loop Constraint' is created and attached to the sketch. Otherwise it is established whether the endpoints of the sketch should join with any other road endpoints. This is achieved by creating a matrix of distances between the start and end points of the sketch with every other endpoint in the road network. The minimum distance between the start point of the sketch, S_S, with the other endpoints is selected as the most suitable candidate for a connection with the start of the sketch. The minimum distance between the end point of the sketch, S_E, with the other endpoints is selected as the most suitable candidate for a connection to the end of the sketch. If these minimum distances are less than the connection threshold, T, 'Straight Constraints' are created and attached to the sketch and connecting roads (Figure 4.1).

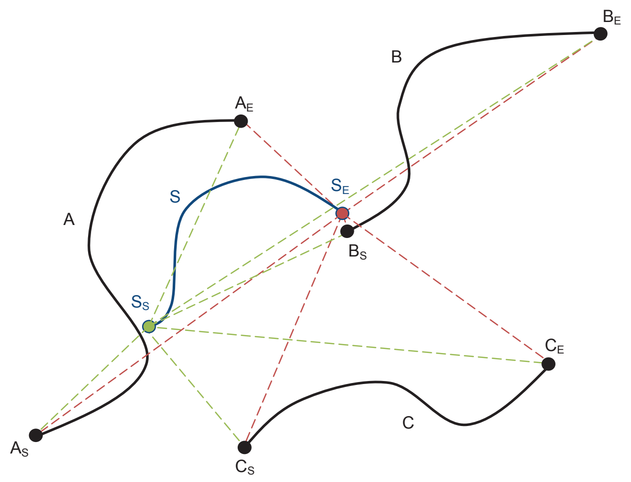


Figure 4.1: The user's sketch (blue) is tested for connection with the endpoints of existing roads A, B, and C (black). The distances between the start point of the sketch, S_S, and the endpoints of the other roads are stored in a matrix (green lines). The distances between the end point of the sketch, S_E, and the endpoints of the other roads are also stored in a matrix (red lines). The minimum distance in each set is then compared with the connection threshold, T, to determine whether the user's sketch should connect to any of the existing roads. In the example above, the closest endpoint to S_E is B_S, therefore these points are candidates for a connection.

Finally, if no other connections have been identified, it is established whether the endpoints of the sketch intersect any road(s) in the network to form a junction. For example, to identify a T-junction, a matrix is created that stores the Euclidean distances between the start and end points of the sketch with the closest point along each road. The minimum distance in each set is compared to the connection threshold, T, and if the distance is less than the threshold, a 'T-Constraint' is created and attached to the sketch and connecting roads (Figure 4.2).

4.1.3 Generate a clothoid curve

A road centre line is generated in the xz-plane using a series of piecewise clothoid curves constructed from the sampled input. To construct piecewise clothoid curves a technique proposed by McCrae and Singh [MS09b] is adopted, where a set of sketch

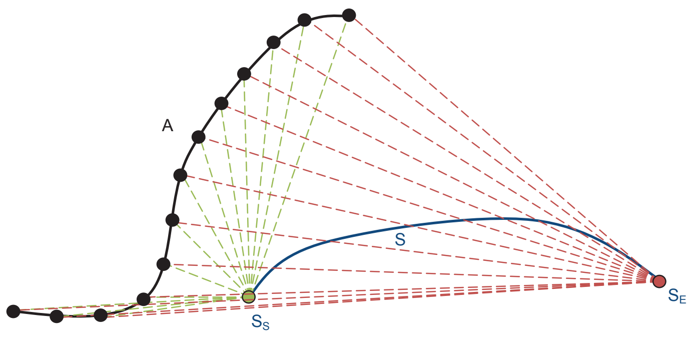
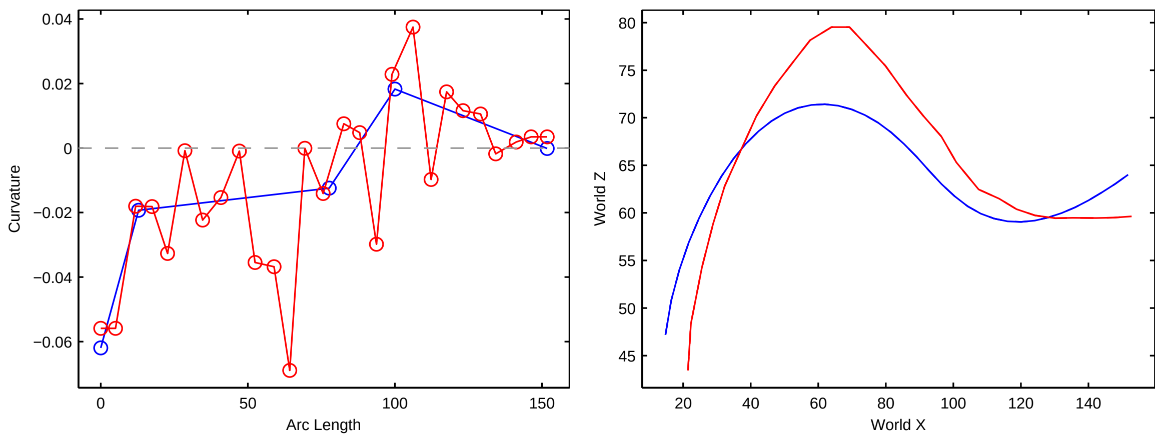


Figure 4.2: The user's sketch (blue) is tested for connection with the existing road A (black). The distance between the start point of the sketch, S_S, and each sample point along road A is stored in a matrix (green lines). The distance between the end point of the sketch, S_E, and each sample point along road A is also stored in a matrix (red lines). The minimum distance in each set is then compared with the connection threshold, T, to determine connectivity.

samples are transformed from Euclidean-space to curvature-space by making a discrete curvature estimation at each sample along the length of its arc, estimated from the circum-circle formed between neighbouring samples. The curve is then segmented into a minimal set of linear pieces in curvature-space using a dynamic programming algorithm that finds a suitable balance between the number of output line segments and the fitting error (Figure 4.3).

Each output segment in curvature-space is directly mapped to a clothoid in Euclidean-space using rational approximations, as described in Equations 2.1.1 to 2.1.4. The resulting clothoids are then translated and rotated so that their endpoints coincide with adjacent segments and their tangents align to form a continuous spline.

Finally, a two-dimensional rigid transformation is applied to the entire curve to fit over the original sketch. McCrae and Singh suggest weighting the endpoints of the curve when computing the required transformation so that these points better coincide with the endpoints of the original sketch [MS09b]. However, instead of just



(a) Curvature-space (b) Euclidean-space

Figure 4.3: (a) The original (red) and approximated (blue) sketch in curvature-space. (b) Euclidean-space representation of the original sketch (red) and the approximated clothoid (blue).

weighting the endpoints, weights are applied across the entire curve using an inverse Gaussian distribution to produce a mapping between the position and tangents of the curve's endpoints that are closer in proximity and direction to those of the original sketch.

4.1.4 Constrain the curve in the xz-plane

The output from the previous stage results in a curve that represents the original sketch. However, to ensure that the curve connects to the road network as specified, the shape of the curve is constrained in the xz-plane to meet the requirements of the 'Connection Constraints' that were identified in Section 4.1.2. These constraints are balanced with 'Geometric Constraints' to minimise road tangents, gradients, and the rate at which these values change, to ensure that the geometry appears realistic.

The algorithm presented converges to this balanced solution by incrementally adjusting the position of samples along the curve in a direction that best satisfies the applied constraints. Throughout this process the clothoid is adjusted in curvature-space to maintain the fundamental and desirable property where its curvature varies

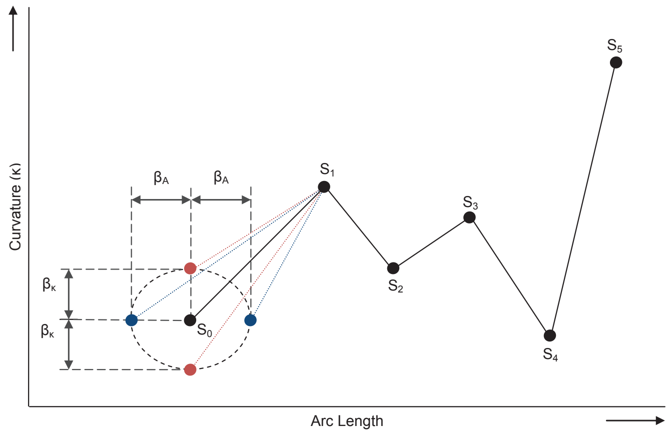


Figure 4.4: Each sample of the clothoid in curvature-space (S_0 to S_5) is incremented by $\pm\beta_\kappa$ along the curvature axis (red) and by $\pm\beta_A$ along the arc length axis (blue). For each increment, the accumulated error, ϵ, is measured and the result is stored in a matrix. The increment that produced the minimum error is then applied and the process recurses until no improvement is possible or the improvement is negligible.

linearly.

Each sample of the clothoid in curvature-space is incremented by $\pm\beta_\kappa$ along the curvature axis and by $\pm\beta_A$ along the arc length axis (Figure 4.4). For each increment, the accumulated error, ϵ, is measured and the result is stored in a matrix. This error is a numerical representation of how satisfied the constraints are, with smaller error values corresponding to greater satisfaction:

$$\epsilon = \sum_{i=1}^{n} w_i k_i$$

where k_i is the evaluated error for constraint i of the n applied constraints, and w_i is used to weight a constraint to bias the algorithm to produce roads that exhibit particular characteristics.

The increment that produced the minimum error is then applied and the process recurses until no improvement is possible or the improvement is negligible. The algorithm may not always reach a global minimum error, but will instead find a local minimum. This typically refines the path for the road whilst maintaining a high resemblance to the user's initial sketch. There may be instances where the system becomes over-constrained and produces undesirable results (Figure 4.10). However, the minimum error value could be made visible to the user and compared against a threshold value to highlight unsatisfactory roads. In this instance, constraint weights could be manipulated to adjust constraint priority.

As the curvature of a circle is defined as the reciprocal of its radius, the value of κ is bound between $[-2/R_w, 2/R_w]$, where the width of the road in metres $R_w > 0$. In this system, $R_w = 8$, $\beta_\kappa = 0.01$, and $\beta_A = 0.5$ (selected through experimentation and analysis). Furthermore, as the representation of the curve in curvature-space is a function (i.e. there exists a one-to-one mapping between the position along a curve and the curvature at its corresponding point), for any sample i, the value of its arc length, A_i, must lie between its neighbours such that $A_{i-1} < A_i < A_{i+1}$.

The algorithm is both powerful and flexible, as a constraint evaluation k_i can be customised to bias the geometry of the road to converge to specific characteristics. As an example, a built-in constraint evaluation is defined for when a sketch connects to a road to form a T-junction, where the distance between the connecting points, the gradient of the sketch's connecting tangent (relative to the perpendicular road tangent at the join point), and the change between consecutive sketch tangents are all minimal (Figure 4.5):

$$k_i = w_1\rho^2 + w_2\tau^2 + w_3\Delta\tau^2$$

where k_i is the evaluated error for constraint i, ρ is the proximity error at the join point, τ is the connecting tangential error, and $\Delta\tau$ represents the change of the gradient between consecutive tangents. Weights w_1 to w_3 are used to bias each of these properties. It is important to note that this is just one example of a constraint

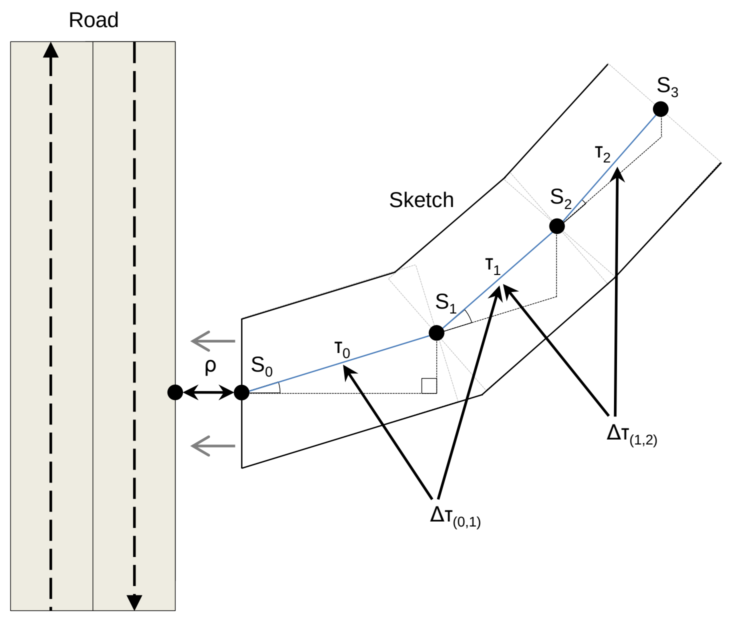


Figure 4.5: A built-in constraint evaluation is defined for when a sketch connects to a road to form a T-junction, where the distance between the connecting points (ρ), the gradient of the sketch's connecting tangent, τ_0 (relative to the perpendicular road tangent at the join point), and the change between consecutive sketch tangents ($\Delta\tau_{(0,1)}, \Delta\tau_{(1,2)}, ..., \Delta\tau_{(n-1,n)}$) are all minimal.

evaluation, and by creating different evaluation functions from expert domain knowledge, roads that conform more precisely to road design principles and practices could be created, but at the cost of increasing the processing time. Adversely, more approximate evaluation functions could be created to generate roads in a reduced time for prototyping applications.

Once the algorithm has converged to a solution the ends of the sketch are clamped to any connection points previously identified.

4.1.5 Project the curve onto the terrain

Until now the sketch has been configured in the xz-domain only. To incorporate height, each sample s along the clothoid curve is projected onto the terrain, which is represented as a heightmap whose axes are aligned with the world coordinate system. Aligning these axes enables us to identify the tile containing s, as described below:

$$t_x = \lfloor s_x/t_w \rfloor, \quad t_z = \lfloor s_z/t_h \rfloor$$

$$I = (t_x, t_z)$$

where I is the two-dimensional index of the heightmap tile containing s. The variables s_x and s_z represent the x and z components of the sample s, and t_w and t_h represent the width and height of tiles in the heightmap respectively (Figure 4.6).

The triangle that contains s is then identified by comparing u and v, which are defined as follows:

$$u = (s_x/t_w) - t_x, \quad v = (s_z/t_z) - t_z$$

If $1 - u > v$, then s is considered to be inside the top-left triangle of the tile, otherwise it lies within the bottom-right triangle. The height of the sample s can now be determined using the Barycentric coordinates of the identified triangle. For example, when s is contained within the top-left triangle of the tile ($\triangle ABC$), its projected three-dimensional coordinate, and thus its height, can be calculated as demonstrated in Equation 4.1.1 and Figure 4.6.

$$\overrightarrow{s} = \overrightarrow{A} + u\overrightarrow{AB} + v\overrightarrow{AC} \tag{4.1.1}$$

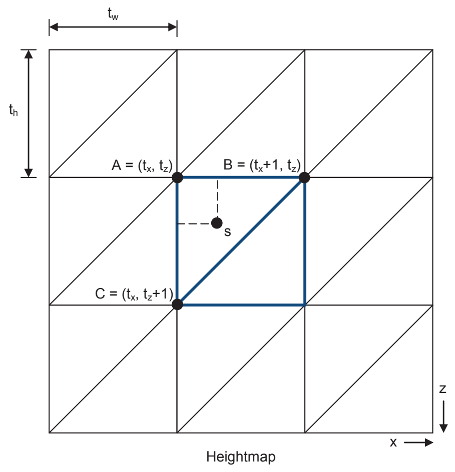


Figure 4.6: Each clothoid sample s is projected onto the terrain, which is represented as a heightmap whose axes are aligned with the world coordinate system. The tile containing s (blue) and triangle ($\triangle ABC$) are identified. The Barycentric coordinates of the identified triangle can then be used to determine the projected three-dimensional coordinate of s.

4.1.6 Constrain the curve in the y-dimension

Projecting the clothoid directly onto the terrain can produce an unrealistic centre line for the road, especially if the user were to sketch onto an undulating or mountainous terrain. To produce suitable results, the inclination of the road must be gradual and the gradient of the incline must be navigable by vehicles [Hig02]. Therefore, this projection is used as a starting guide for the shape and position of the road. The clothoid is then constrained in the y-dimension using similar techniques to those presented in Step 4.1.4 of the algorithm.

As before, the algorithm converges to a balanced solution that considers 'Connection' and 'Geometric' constraints, but adjusts the height of the curve directly in Euclidean-space. When constraining the height of the road 'Geometric Constraints' are implemented that evaluate the amount of ground displaced from the original terrain, the gradient between consecutive samples, and the change between consecutive gradients as described in the equation below:

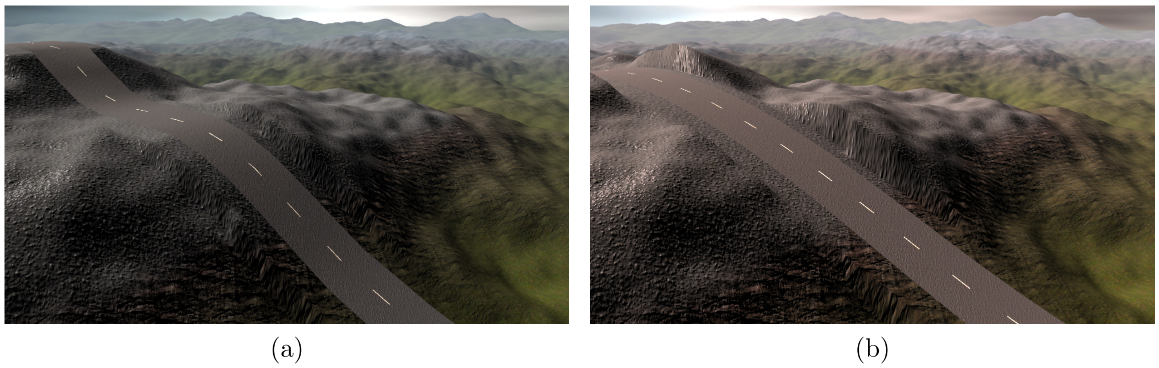


Figure 4.7: Weights can be adjusted to produce roads with certain characteristics. (a) Proximity is weighted highly. (b) Gradient properties are weighted highly.

$$k_i = w_1 \sum_{j=1}^{n} (H_j - h_j)^2 + w_2 \sum_{j=1}^{n-1} g(j, j+1)^2 + w_3 \sum_{j=1}^{n-2} \Delta\big(g(j, j+1)^2\big)^2 \qquad (4.1.2)$$

where k_i is the evaluated error for constraint i, H_j is the original height of the terrain at point j along the curve, h_j is the height of the road at this point, $g(j, j+1)$ is the gradient of the road between samples j and $j+1$, and $\Delta g(j, j+1)$ is the change in this gradient where $n > 2$. As before, weights w_1 to w_3 are used to bias the road to adhere to certain properties (Figure 4.7).

Each sample of the curve in Euclidean-space is incremented by $\pm\beta_Y$ along the y-axis. In this system $\beta_Y = 0.01$ metres. Once the algorithm has converged to a solution, the heights of the endpoints of the sketch are clamped to any of the connection points previously identified.

4.1.7 Construct the road geometry

To create the edges of the road, the algorithm iterates through each sample of the curve and stores two vertices that are perpendicular and equally spaced either side of the sample. The position of the vertices can be calculated using the vector between consecutive samples as a forward-vector, and by performing a cross product between this vector and the world up-vector to obtain a right-vector. The vertices are then placed $\pm$ a distance x from the clothoid sample along its right-vector.

The convergent algorithm is then applied to the edges of the road to produce a suitable camber. As in Section 4.1.6, the curve is manipulated in Euclidean-space by increments of $\pm\beta_Y$, where $\beta_Y = 0.01$ metres.

To displace the terrain the algorithm iterates through each face of the road model and identifies all of the heightmap vertices that have their xz-coordinate interior to the road face. These vertices are then clamped to the height of the road. Verges are created on each side of the road using the same technique.

4.2 Influence regions

'Influence Regions' are user-specified areas of the environment that influence the path of generated roads. These regions provide bias throughout the road design process and can be tailored to either attract or repel roads to/from certain obstacles or designated areas, such as forestation, listed buildings, marshland, etc. By creating these regions designers can influence the road generation process and create roads that take a minimum-cost route through an environment, which may be difficult or unintuitive to determine using more traditional design methodologies. A set of tools to create 'Influence Regions' are described below:

4.2.1 Influence points

Using the 'Point Tool' the user can select a point-radius 'Influence Region' on the terrain. The user first selects a centre-point for the area using the mouse, and then moves the cursor left-and-right to decrease and increase its radius respectively. A successive mouse click confirms the size of the region, permitting the user to then adjust its influence by moving the cursor up-and-down. The user-interface was designed

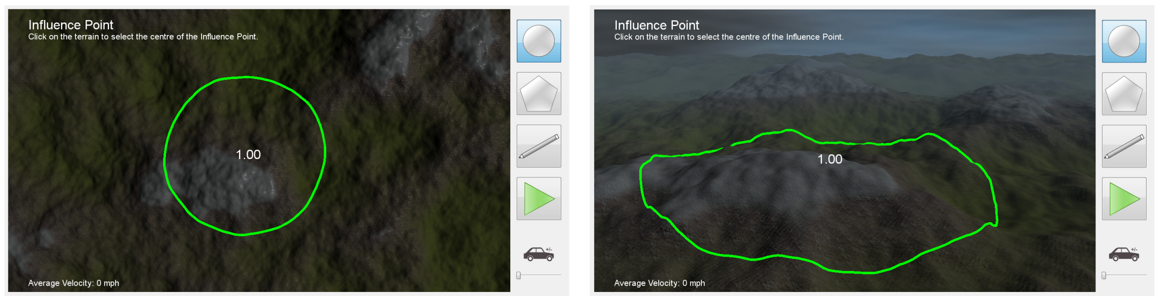

Influence Point
Click on the terrain to select the centre of the Influence Point.
1.00
Average Velocity: 0 mph
Influence Point
Click on the terrain to select the centre of the Influence Point.
1.00
Average Velocity: 0 mph

Figure 4.8: An example of an 'Influence Point' viewed from different perspectives. The value at its centre represents its weighting (1) and its boundaries are projected onto the terrain to better show its size and position within the scene. For visual enhancement, region boundaries are also coloured to reflect their value.

to allow the user to manipulate these regions using simple and intuitive gestures to increase user-productivity. It also establishes a design workflow that could be ported to sketch-based or touch devices with relative ease. For example, on a sketch-based device the user would select the centre-point of the 'Influence Region' using a stylus and then drag the pen across the different axes to change its properties. Influence is limited between [-1, 1], with a negative influence value denoting a repulsion region and a positive influence value denoting an attraction region. An area with zero influence will have no affect on the generated road. To increase user-productivity and to permit greater precision when designing 'Influence Regions', a region's weighting is displayed at its centre and its boundary is projected onto the terrain. For visual enhancement these boundaries are also coloured to reflect their value. For example, the boundary of a negative region is coloured red and the boundary of a positive region is coloured green, with saturation scaled by magnitude (Figure 4.8).

When generating a road, its path is constrained to conform to 'Influence Points' by including them as 'Geometric Constraints' in the algorithm. As roads are adjusted in the direction that best satisfies the accumulation of all the applied constraints, the magnitude of the constraint evaluation must be greatest when evaluated at its centre and fall-off with distance as detailed below:

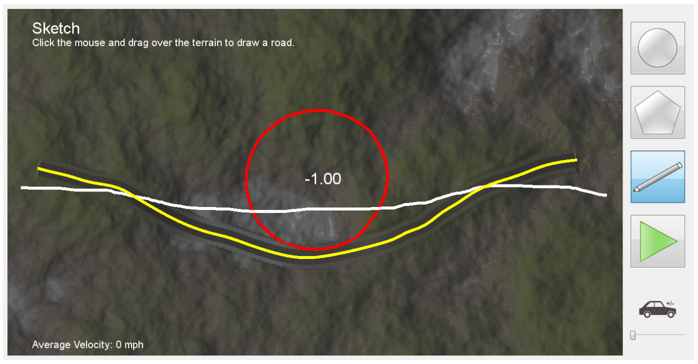


Figure 4.9: The original user-sketch (white) is manipulated in curvature-space so that it avoids the negative 'Influence Region' and produces the road in yellow.

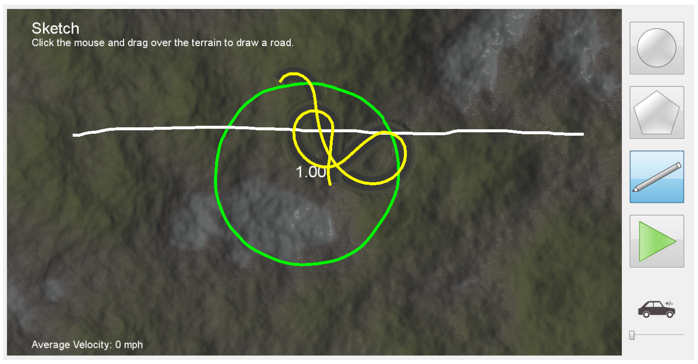


Figure 4.10: The original user-sketch (white) is manipulated in curvature-space so that it is attracted towards to the centre of the 'Influence Region'. Sample points along the resulting road (yellow) are 'pulled' towards the centre of the region. However, as the algorithm continues to iterate, these endpoints become close enough to the region that it has direct influence on their position, and are 'pulled' further towards its centre, beyond user-expectation.

$$k_i = \pm\big(1 - \frac{\min\left(|p_j - c|, r\right)}{r}\big)^2 \tag{4.2.1}$$

where p_j is the projection of the curve onto the xz-plane at point j along the road, c is the centre of the 'Influence Point', and r is the radius of the region. The evaluation error is squared to provide a greater bias towards the apex of the region. Figures 4.9 and 4.10 demonstrate how the path of the road is manipulated to avoid a negative 'Influence Point' and attracted to a positive 'Influence Point' respectively.

In these examples, the negative 'Influence Point' has adjusted the path of the

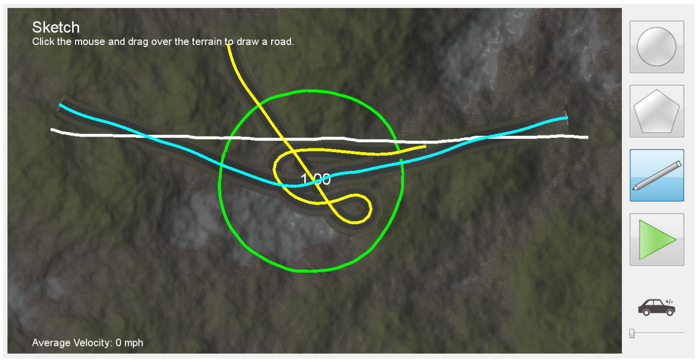


Figure 4.11: A deviation factor is included to prevent roads from ‘knotting’. The original user-sketch is shown in white, the road generated with a deviation factor $\gamma = 0$ is shown in yellow, and the road generated with a deviation factor $\gamma > 0$ is shown in blue.

road to avoid the marked area. However, in the example with the positive ‘Influence Region’, the resulting road becomes ‘knotted’ and is therefore undesirable. Through analysis it was discovered that the error evaluation algorithm would continue to ‘pull’ a road towards the centre of a positive region beyond user-expectation, causing it to wrap around itself in order to reduce its evaluation error. To prevent this from occurring an additional parameter γ is incorporated in Equation 4.2.1, that increases the evaluation error by a function of the number of iterations of the algorithm, b:

$$k_i = \max(\gamma, 0)b \pm \big(1 - \frac{\min(|p_j - c|, r)}{r}\big)^2$$

γ is defined as a deviation factor from the initial sketch, where a value of $\gamma = 0$ permits an unlimited level of deviation that decreases as the value of γ increases. The deviation factor has a default value of 0.1, but can be changed by the user in the system settings. Figure 4.11 demonstrates the effect of the deviation parameter when sketching a road through a positive ‘Influence Point’.

Various results can be produced from the same initial sketch using different deviation factors as shown in Figure 4.12. As the deviation parameter tends to 0, ‘Influence Regions’ have a greater influence, which can cause the endpoints of roads

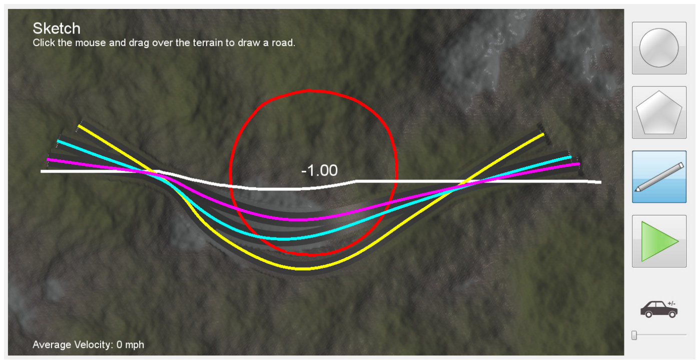
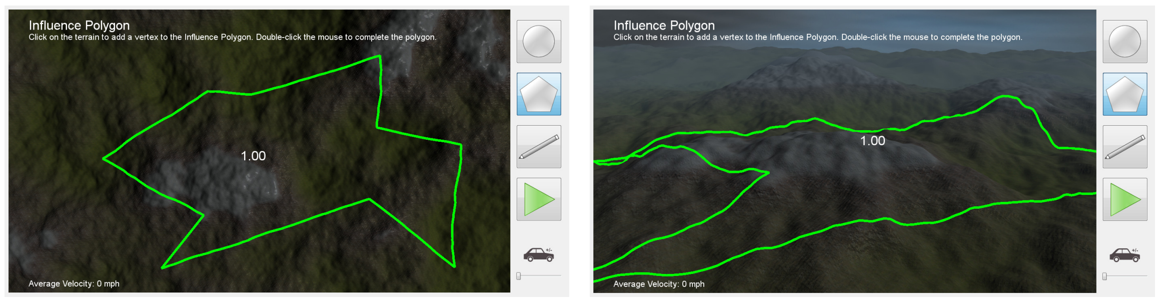


Figure 4.12: Roads with various characteristics can be produced from the same sketch when using different deviation factors. The original user-sketch (white) produces the yellow road when there is no deviation limitation, i.e. when $\gamma = 0$, the blue road is produced when $\gamma = 0.1$, and the pink road is produced when $\gamma = 100$.

to be displaced such that they no longer coincide with the original sketch. This effect diminishes as the distance between each endpoint of the sketch and the 'Influence Region' increases. To mitigate this issue, negative 'Influence Regions' could be positioned around the endpoints of the sketch to penalise deviation. To maintain continuity between connected road segments, an additional constraint is applied that penalises adjustments that cause its endpoints to deviate from their original location. Future work could extend this idea by incorporating a clamping tool that allows the user to specify fixed points along the sketch that the road should pass through.

4.2.2 Influence polygons

The toolset also includes a feature to specify more complex 'Influence Regions', such as concave and convex shapes using the 'Polygon Tool'. Upon enabling this feature, the user creates consecutive vertices of the polygon by clicking on the terrain and finalises the region with a double-click to close the polygon boundary. As with 'Influence Points' the user can then move the cursor up-and-down to increase and decrease the influence of the region (Figure 4.13). On a sketch-based or touch-device, the shape of an 'Influence Region' could be input as a sketch, which would then be approximated


Influence Polygon
Click on the terrain to add a vertex to the Influence Polygon. Double-click the mouse to complete the polygon.

Influence Polygon
Click on the terrain to add a vertex to the Influence Polygon. Double-click the mouse to complete the polygon.

Figure 4.13: An example of an 'Influence Polygon' viewed from different perspectives. The value at its centre represents its weighting (1) and its boundaries are projected onto the terrain to better show its size and position within the scene. For visual enhancement, region boundaries are also coloured to reflect their value.

using a polygon.

To evaluate the constraint of an 'Influence Polygon', an error function must be created that returns an error value with a magnitude that increases as the distance from an interior sample point to its nearest edge also increases. This problem is related to architectural design, where an architect must construct a roof for a building to protect its contents. To provide protection against rain the faces of a roof are often sloped so that water is directed away from the building and into designated drainage pipes.

When evaluating an error function, a virtual roof structure is generated and sampled where the height of the roof at a particular point is used to determine its error. In architecture there are a multitude of roof designs in use. In the system a 'Hip Roof' is generated, as it slopes upward from each edge of its footprint and has no vertical faces. To construct a 'roof', a technique for generating roof models from building footprints is used, as described by Laycock and Day in 2003 [LD03]. In their work, the authors first extract the Straight Skeleton of a building footprint [AAAG95]. This is a topological structure that represents a polygon's skeleton using only straight edges. It is formed by shrinking its boundary by moving all edges towards its interior

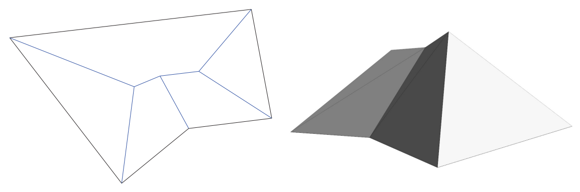


Figure 4.14: (a) An input polygon (black) and its straight skeleton (blue). (b) The skeleton is then lifted to produce a roof-like structure.

simultaneously and at the same speed. Each vertex of the polygon is moved along the angular bisector constructed by its adjacent edges. During the shrinking process two events can occur, an 'Edge' event, where an edge shrinks to a length of zero, or a 'Split' event, where an edge is split by a reflex vertex. The resulting polygon(s) are then shrunk recursively whilst they have a non-zero area. The implementation documented in [FO98] was used. Figure 4.14(a) illustrates the result of the algorithm applied to a simple polygon.

For each vertex of the Straight Skeleton, its height is defined as the distance from the vertex to its nearest edge. Height values are then normalised by dividing each value by the height of the highest vertex on the roof. A triangulation of the structure is performed so that the height of any point on the roof can be determined using Barycentric coordinates (Figure 4.14(b)).

The constraint evaluation for an 'Influence Polygon' is then defined as follows:

$$k_i = \max(\gamma, 0)b \pm {r_j}^2$$

where b is the current iteration of the algorithm and r_j is the height of the 'roof' at point j along the road, calculated using Barycentric coordinates. As with 'Influence Points', the error is squared to increase bias away from the region boundary. Figure 4.15 demonstrates how the path of the road is manipulated to avoid a negative

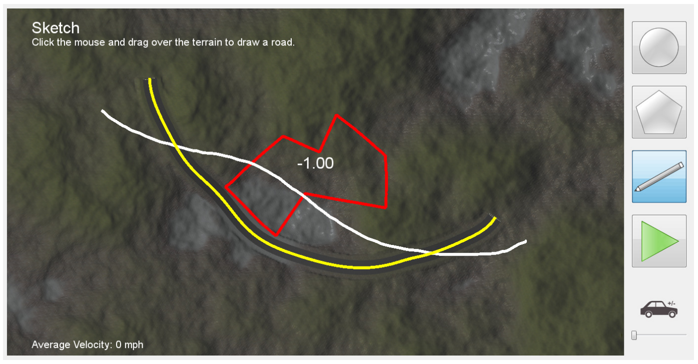


Figure 4.15: The original user-sketch (white) is manipulated in curvature-space so that it avoids the negative 'Influence Region', producing the road in yellow.

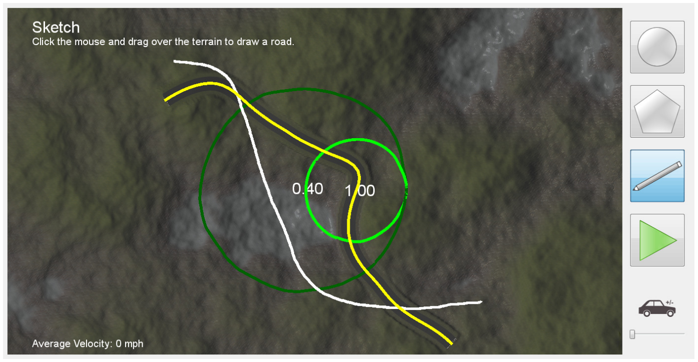


Figure 4.16: More complex scenarios can be modelled using multiple 'Influence Points'. In this example, a large 'Influence Point' is used to attract the original user-sketch (white) towards its centre. A smaller region with higher influence is used to offset the attraction, producing the road in yellow.

'Influence Polygon'.

4.2.3 Accumulative regions

To model more complex scenarios 'Influence Regions' can be combined, as the total error evaluation at a particular point is the summation of all local influence errors. For example, if two identical 'Influence Points' were created at the same position, influence is doubled. Figure 4.16 illustrates how the union of 'Influence Points' can alter the user's sketch.

The true potential of the algorithm is realised when there are a large number of

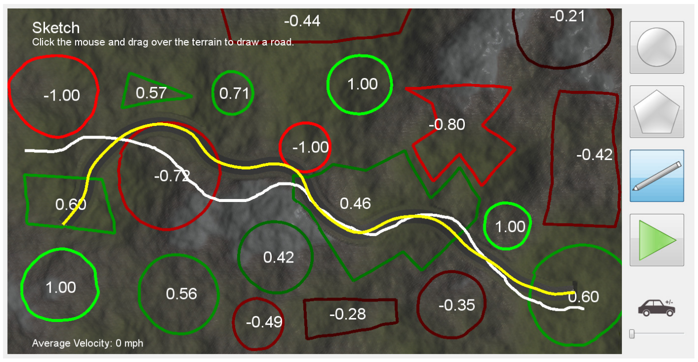


Figure 4.17: The original user-sketch (white) is manipulated in curvature-space to find an optimal path for the given constraints, producing the road in yellow.

'Influence Regions' within the environment and the designer wishes to create a road from location A to location B that avoids or is attracted to certain obstacles or areas. In this scenario, the user can construct the necessary 'Influence Regions' and sketch an approximate path for the road. The algorithm will then iteratively manipulate the sketch to produce an optimal path for the given constraints (Figure 4.17).

4.3 System evaluation

A user-study was conducted to evaluate the usability of the system and the quality of roads generated in a diverse range of scenarios. User feedback was recorded, relating to usefulness and desirability of positive and negative 'Influence Regions' when generating roads. The results of this study were used to improve the system as detailed in Section 4.3.3. Two goals have been defined which can be used to measure success:

1. Provide a natural and intuitive user-interface for designing roads in virtual environments.

2. Provide a practical road design tool that is true to the user's intentions.

4.3.1 Participants

A group of 19 individuals participated in the user-study. This sample included 16 male and 3 female participants between the ages of 16 and 49 years old, with varying levels of computer experience. Of those participants 5 had experience using 3D modelling software, such as 3D Studio Max, Maya, and Blender. Each participant was given a short introduction to the system and instructed to complete a set of tasks. Each user interacted with the system for approximately 20 minutes and was asked to complete a post-study survey, which formed the primary source of user feedback. Participants were also encouraged to comment throughout the study, particularly if they had difficulties when attempting to complete the required tasks.

4.3.2 Tasks

To guide the users through the various features of the system, a set of 13 test cards were prepared (Figure 4.18). Each card illustrates a particular road design viewed from above and users were asked to replicate these designs on the terrain shown in Figure 4.8. Users were presented with each test card in an increasing order of complexity, instructed to inspect the roads generated, and determine whether the resulting roads matched their initial intention and/or met their expectation in relation to quality. The test cards detailed a diverse range of designs that included scenes with simple roads, junctions, and loops. Users were also encouraged to repeat each task multiple times to test the system with sufficient variability and, where appropriate, with varying levels of influence. Throughout the evaluation users were instructed to sketch using the mouse.

At the end of the session participants were asked to complete a two-part survey. The first part utilises the industry recognised System Usability Scale (SUS), a ten-item scale that provides a global view of subjective assessments of usability [Bro96].

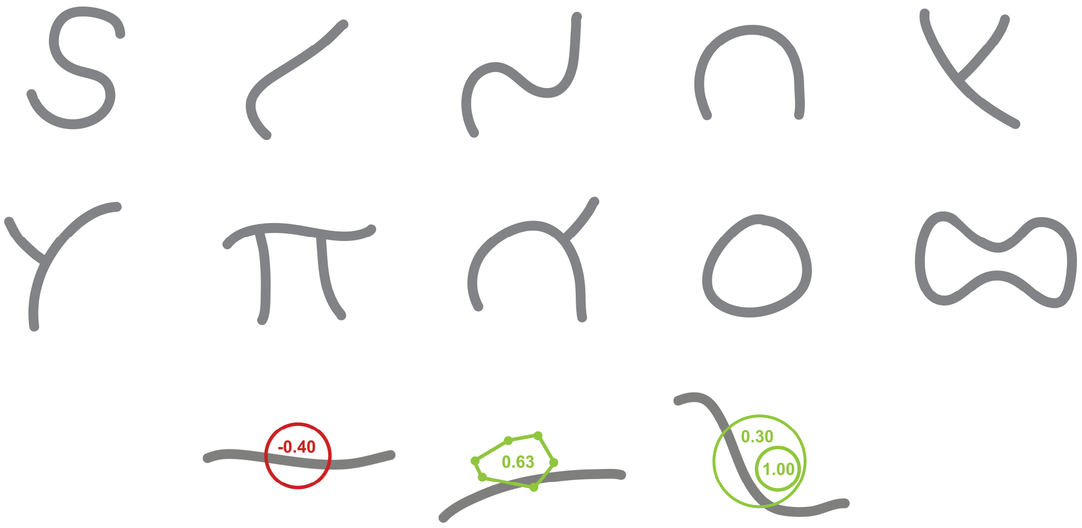


Figure 4.18: A set of 13 test cards were prepared. Each card illustrates a particular road design viewed from above and users were asked to replicate these designs using the software.

The second part of the survey includes a five-item scale, in which participants were asked to score roads generated by the system, in relation to their initial intention and their expectation in relation to quality. At the end of the survey participants were encouraged to comment on both positive and negative aspects of the system.

4.3.3 Results and discussion

The ten SUS statements are presented in Table 4.1 and the corresponding scores are presented in Table 4.2 and Figure 4.19. For each statement, users were asked to score from 1 (strongly disagree) to 5 (strongly agree). A single SUS score is then computed using the algorithm presented in [Bro96], within the range of 0 to 100, with higher scores indicating greater usability. As the SUS is commonly used in industry, the system presented can be compared to both existing and future similar systems with relative ease. The results show that the system has an average SUS score of 89.34. Recently there has been extensive research on how to best interpret

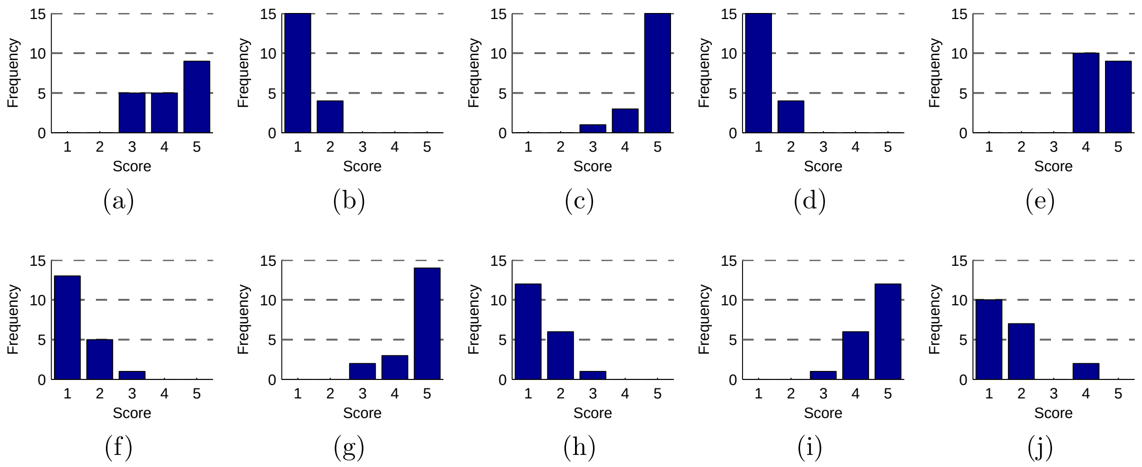


Figure 4.19: Participants were asked to score the ten SUS statements (Table 4.1) from 1 (strongly disagree) to 5 (strongly agree) (Table 4.2).

a	I think that I would like to use this system frequently
b	I found the system unnecessarily complex
c	I thought the system was easy to use
d	I think that I would need the support of a technical person to be able to use this system
e	I found the various functions in this system were well integrated
f	I thought there was too much inconsistency in the system
g	I would imagine that most people would learn to use this system very quickly
h	I found the system very cumbersome to use
i	I felt very confident using the system
j	I needed to learn a lot of things before I could get going with this system

Table 4.1: The ten System Usability Scale (SUS) statements [Bro96].

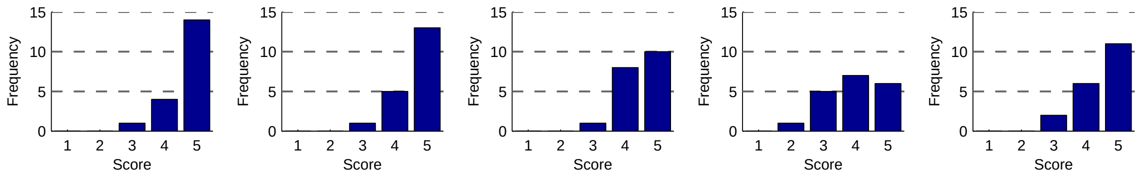
SUS scores. Bangor et al. [BKM08] present an empirical evaluation of the System Usability Scale and believe that the grading standard adopted by industry, which is analogous to the grading scale of many schools, is well supported by both their own research and collective experience. In this grading scale a score of 90 or above is representative of 'truely superior products' and is analogous to a grade A, a score of 80 to 89 is equivalent to a grade B, and any score below 50 should be cause for 'significant concern'. With a high SUS score of 89.34, the first system goal has been successfully met, providing a natural and intuitive interface for designing roads.

Statement	Frequency				
	1	2	3	4	5
a	0	0	5	5	9
b	15	4	0	0	0
c	0	0	1	3	15
d	15	4	0	0	0
e	0	0	0	10	9
f	13	5	1	0	0
g	0	0	2	3	14
h	12	6	1	0	0
i	0	0	1	6	12
j	10	7	0	2	0

Table 4.2: The frequency scores of the ten SUS statements (Table 4.1) evaluated by the 19 participants.

The results of the second part of the survey are presented in Table 4.3 and Figure 4.20. Participants were asked to score the quality of roads generated by the system, from 1 (poor quality) to 5 (high quality). The results demonstrate that each road category received high scores with the majority of users awarding top marks, with the exception of the category of roads that were affected by positive 'Influence Regions'. For this category the most frequent score was 4, with an average scoring of 3.95. This lower score was reflected in the participants' comments, as many individuals found that the effect of positive 'Influence Regions' with default parameters did not always produce the best results. On occasions roads were 'pulled' towards the centres of these regions beyond user-expectation, replicating the scenario presented in Figure 4.10. Therefore, this lower score was attributed to the choice of the deviation factor γ. It was generally considered that better results would be produced with more experience using the system. As a result of this evaluation, a slider has since been added to the user interface to provide a more intuitive method of controlling the deviation factor.

Many of the participants complemented both the simplicity and usability of the



(a) (b) (c) (d) (e)

Figure 4.20: Participants were asked to score the quality of roads generated by the system, from 1 (poor quality) to 5 (high quality) (Table 4.3). This included simple roads (a), road containing junctions (b), 'looped' roads (c), roads generated using positive 'Influence Regions' (d), and roads generated using negative 'Influence Regions' (e).

Road category	Frequency				
	1	2	3	4	5
Simple roads	0	0	1	4	14
Roads containing junctions	0	0	1	5	13
'Looped' roads	0	0	1	8	10
Roads generated using positive 'Influence Regions'	0	1	5	7	6
Roads generated using negative 'Influence Regions'	0	0	2	6	11

Table 4.3: The frequency scores of the five road categories evaluated by the 19 participants.

system. In particular, one of the users with 3D modelling experience was highly impressed by the flexibility and ability to "create complex road networks within a very short timeframe" and commented that they would like to use this system in the future. Another user said that the software was "fun to use" and was impressed by the "ability to create, then play". In addition, several of the participants commented on the usefulness of 'Influence Regions', realising that its application had potential to drastically increase productivity, particularly when utilised in larger and more complex projects. The results show that the system was given high scores in this part of the survey, with an overall average scoring of 4.4. With these results, coupled with the positive feedback from the participants, the system has successfully met its second goal.

To compliment this evaluation, a user study could be conducted with experts in the field, where each participant would score roads generated by other users against input specifications. The roads generated could then be evaluated against both formal and objective criteria, such as those described in [Hig02]. System productivity could also be evaluated to provide an additional metric for comparison against similar software packages.

4.4 System performance

The performance of the road generation algorithm is variable as it primarily depends on the length of the road, the number of constraints applied, the complexity of their evaluation, and the height of the underlying terrain. The most computationally demanding aspect of the algorithm is the converging process, as for each iteration of the algorithm, the accumulated constraint error ϵ was evaluated for each sample as its value was incremented and decremented by β_κ, β_A, and β_Y. To increase performance constraint evaluations could be performed in parallel as each evaluation is independent. Furthermore, β values could be increased so that the algorithm converges to a solution within fewer iterations, but at the cost of reducing accuracy.

4.5 Summary

In this chapter a novel sketch-based tool to semi-automate the design, creation, and visualisation of road networks for virtual environments was presented. The tool is guided by input sketches and a combination of prioritised constraints. These constraints can be tailored to generate roads that exhibit particular characteristics, such as roads with little inclination which cut through the terrain or roads with high curvature that meander through the environment.

The work presented both compliments and extends McCrae and Singh's research on sketch-based path design [MS09b, MS09a] by developing a flexible system that produces roads in three-dimensions across both flat and undulating terrain. The concept of 'Influence Regions' has also been introduced, and a demonstration of how they can be constructed and utilised to attract or repel the path of generated roads to/from certain obstacles or designated areas, has been provided. A user study was conducted to evaluate the usability of the system and to evaluate the quality of roads generated in a diverse range of scenarios. The results indicated that the system is both user-friendly and able to produce roads that are true to the user's intention.

Chapter 5

Vehicle behaviour

One of the major challenges faced by researchers in the computer graphics industry involves the design and implementation of algorithms that allow large-scale simulations to run at real-time frame rates. This is a typical requirement of traffic simulators, as it enables designers to modify and interact with an environment in real time which can reduce design and development times, and the overall cost of urban planning projects. However, many applications will compromise behavioural accuracy or visual quality to obtain the required performance. These challenges also translate to the entertainment industry, as real-time traffic simulation is required in computer games, films, and virtual tourism applications. In this chapter, a model for simulating high-detail traffic networks in real time is presented. The model is agent-based and operates on rules influenced by the pioneering work of Gipps [Gip81]. Furthermore, Nagel and Schreckenberg's cellular automata traffic model [NS92] is extended to provide varying levels of low level-of-detail simulation.

5.1 High-detail simulation

In the proposed system, two types of vehicle are defined with different levels of detail. A high-detail vehicle, v, is rendered with a complex three-dimensional model for

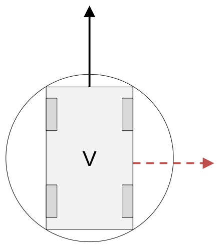


Figure 5.1: A vehicle, v, is rendered with a complex three-dimensional model for visualisation, but for behavioural purposes, is represented in the xz-plane as a two-dimensional rectangle whose bounds encompasses the xz-projection of its geometry to reduce the complexity of collision detection. Each vehicle stores its normalised forward-vector (black arrow), and its normalised right-vector (red arrow). A bounding radius is also stored to optimise collision detection.

visualisation, but for behavioural purposes, is represented in the xz-plane as a two-dimensional rectangle whose bounds encompasses the xz-projection of its geometry to reduce the complexity of collision detection. Each vehicle stores its normalised forward-vector and its normalised right-vector (Figure 5.1). A bounding radius is also stored to optimise collision detection, and it is assumed that vehicles are not permitted to travel off-road for simplification. In the system a left-handed traffic model is adopted where each road consists of two lanes of traffic moving in opposing directions. For navigational purposes, each lane of the road is divided into a number of cells that are equally spaced and tangential to the road centre line (Figure 5.2). Each cell stores a reference to connected previous and next cells, enabling the network to be traversed on a cell-by-cell basis. An 'entry cell' is defined as a cell without a reference to a previous cell, and an 'exit cell' is defined as a cell without a reference to a next cell. These cells are used as potential entry and exit locations on the road network respectively.

Each vehicle plans its route ahead-of-time on a cell-by-cell basis. When travelling between cells, vehicles follow the trajectory defined by interpolating the positions of

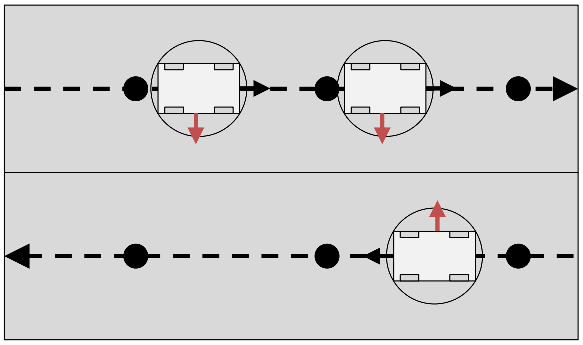


Figure 5.2: Each road lane is divided into a number of cells (black dots) that are equally spaced and tangential to the road centre line.

its four surrounding cells using a Catmull-Rom Spline:

$$q(t) = 0.5 \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}^T \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

where $q(t)$ is the interpolated position of the spline between points p_2 and p_3, at interval $t \in [0, 1]$, where $q(0) = p_2$ and $q(1) = p_3$. Points p_1, p_2, p_3, and p_4 are control points that define the spline. A Catmull-Rom Spline is used because it is guaranteed to pass through the control points (cell locations) and it is C^1 continuous, i.e. where the tangent vectors of joining curve segments are equal at the join point.

To prevent vehicles from colliding, at each update cycle the stopping distance for each vehicle is estimated as described below:

$$s = -\frac{u^2}{2b} + \mu, b \neq 0 \tag{5.1.1}$$

where s is the stopping distance of a vehicle, u is the vehicle's current velocity, b is the vehicle's braking rate, and μ is a safety threshold. In this system a constant braking rate is assumed. Each vehicle models its current speed, desired speed, acceleration, deceleration, and braking rates, and uses time-based animation techniques to update its position along the path defined by the spline. At each frame, vehicles compute a new speed by comparing their current speed and acceleration rate to their preferred speed for the limit of the road. The stopping distance for the vehicle is then updated and the corresponding number of cells along the vehicle's planned route is then checked for occupancy. If there are no other vehicles within the stopping distance, the car is permitted to travel at the new speed, otherwise the vehicle applies its brakes. Fluctuations in speed are modelled by random deceleration.

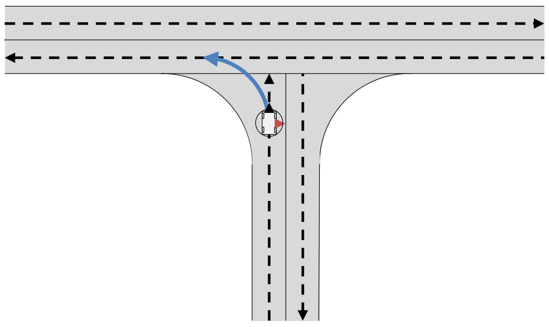
5.1.1 Junctions

When a vehicle approaches its next junction, it reduces its speed to stop at the junction if necessary. A vehicle will only commit to turning at a junction if there is adequate time and space to complete the required manoeuvre and accelerate to a reasonable speed without collision. The particular manoeuvre will determine the time and space that is required for a collision free turn. For example, when modelling a T-junction, 4 manoeuvres are defined as illustrated in Figure 5.3.

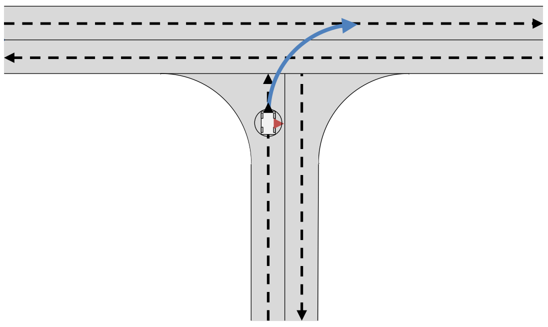
Vehicles performing the first manoeuvre will only consider other vehicles in the lane moving from right-to-left. The time required to complete the first manoeuvre, E, is estimated as described below:

$$E = \phi + \psi$$

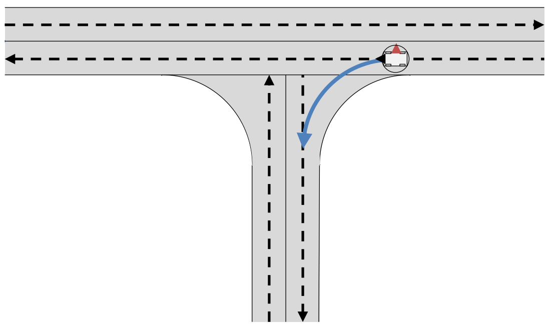
where ϕ is the time required to navigate the turn, and ψ is the time required to accelerate to a reasonable speed. In the system it is assumed that vehicles turn at a constant speed η. Therefore, ϕ is estimated using equations of constant acceleration for up to speeds of η, and equations of constant speed for the remainder of the turn.



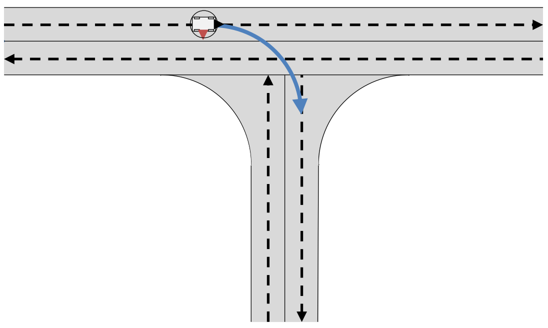
(a) Manoeuvre 1



(b) Manoeuvre 2



(c) Manoeuvre 3



(d) Manoeuvre 4

Figure 5.3: The 4 manoeuvres for a T-junction.

The minimum desirable gap in traffic flow, D, is defined below:

$$D = rES_{max}$$

where S_{max} is the speed limit of the road, and r is a risk/aggression parameter that models diversity by simulating drivers that take unnecessary risks, causing other motorists to brake, and drivers that are over cautious who wait for larger gaps in traffic flow before turning. The times required to complete the other manoeuvres are estimated using the same assumptions, but may have greater constraints. For example the second manoeuvre considers the time required to cross the lane moving from right-to-left, and the time required to join the lane moving from left-to-right (Figure 5.4).

5.1.2 Additional features

To add further realism to the behaviour model, vehicle headlights, rear lights, brake lights, and indicators are modelled. Furthermore, wheel rotation is simulated using standard equations. Environment mapping is used to produce the illusion of accurate reflections [BN76] (Figure 5.5).

5.2 Low-detail simulation

The Nagel and Schreckenberg cellular automata traffic model [NS92] could be used to simulate the behaviour of vehicles that are within the view frustum, but are a sufficient distance away from the viewer. As discussed in Chapter 2, the Nagel and Schreckenberg traffic model can simulate large-scale networks with minimal computational cost, as it operates at discrete locations on a road network at discrete time intervals, and it is highly parallelisable. Despite its discrete nature, the model is able to produce realistic behaviour by exploiting time coherence, i.e., it is assumed that the

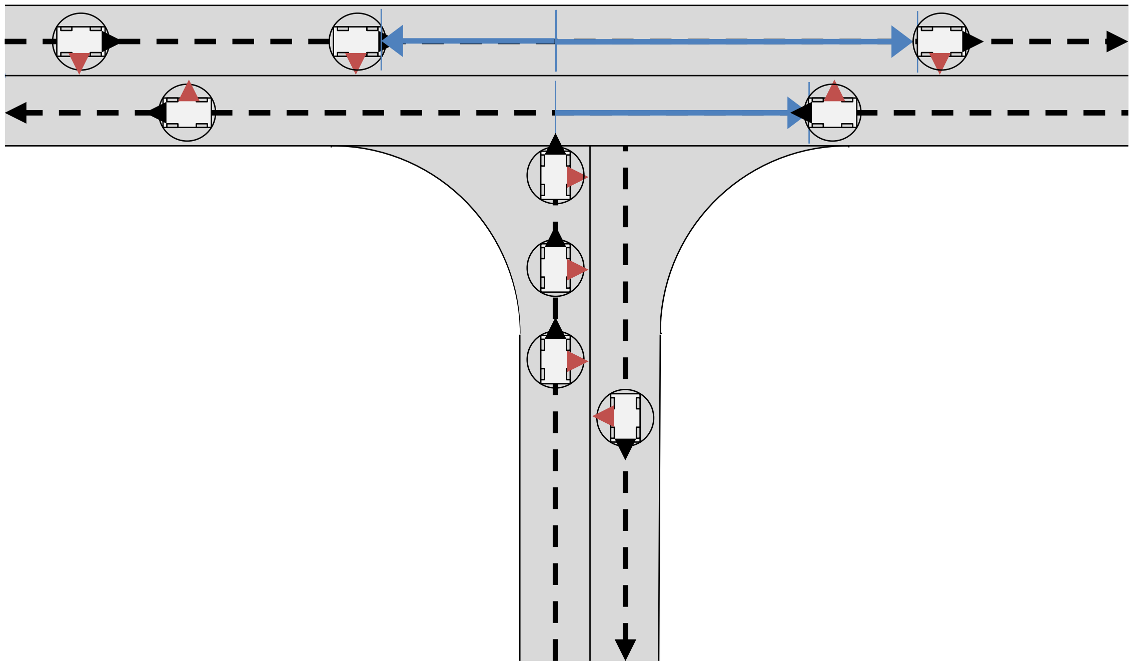


Figure 5.4: The second manoeuvre considers the time required to cross the lane moving from right-to-left, and the time required to join the lane moving from left-to-right. Blue arrows indicate the gaps in traffic flow that the turning vehicle must consider in order to avoid collision.

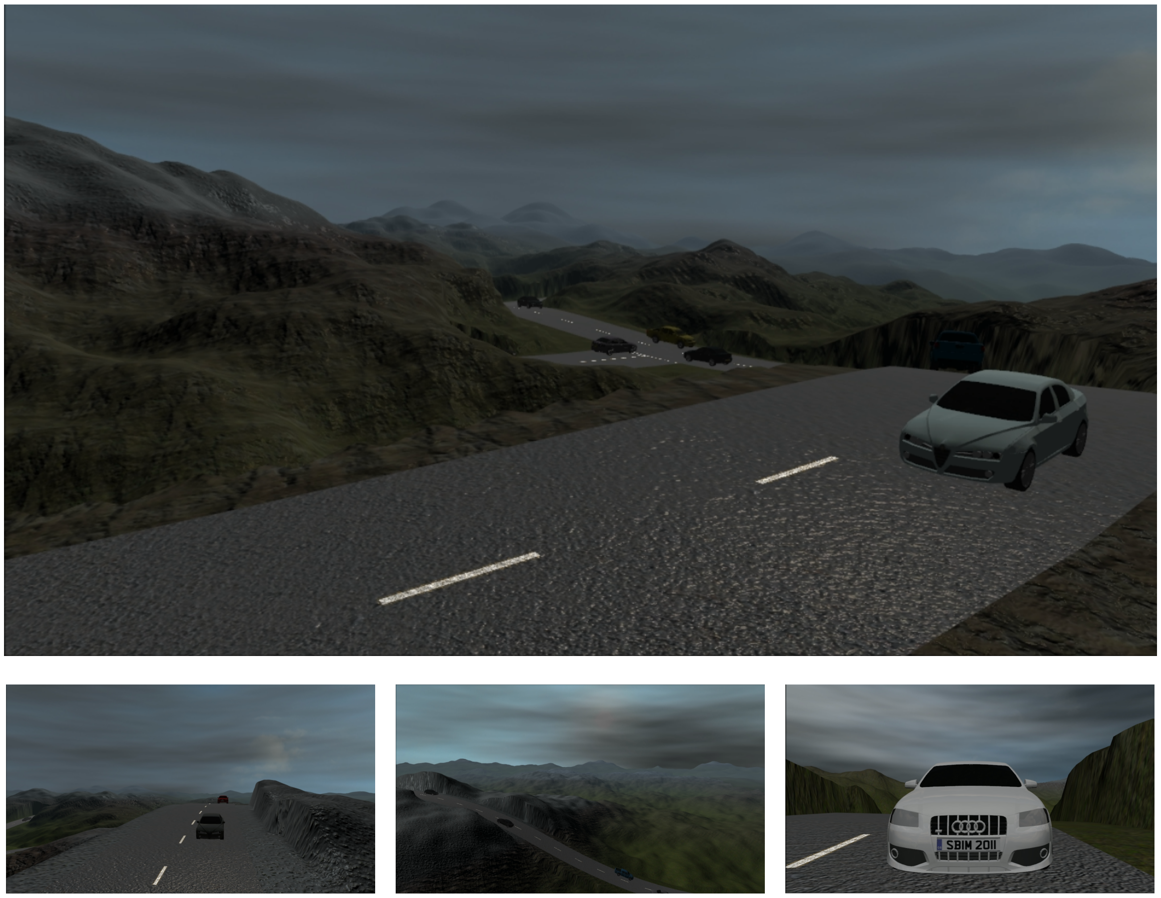


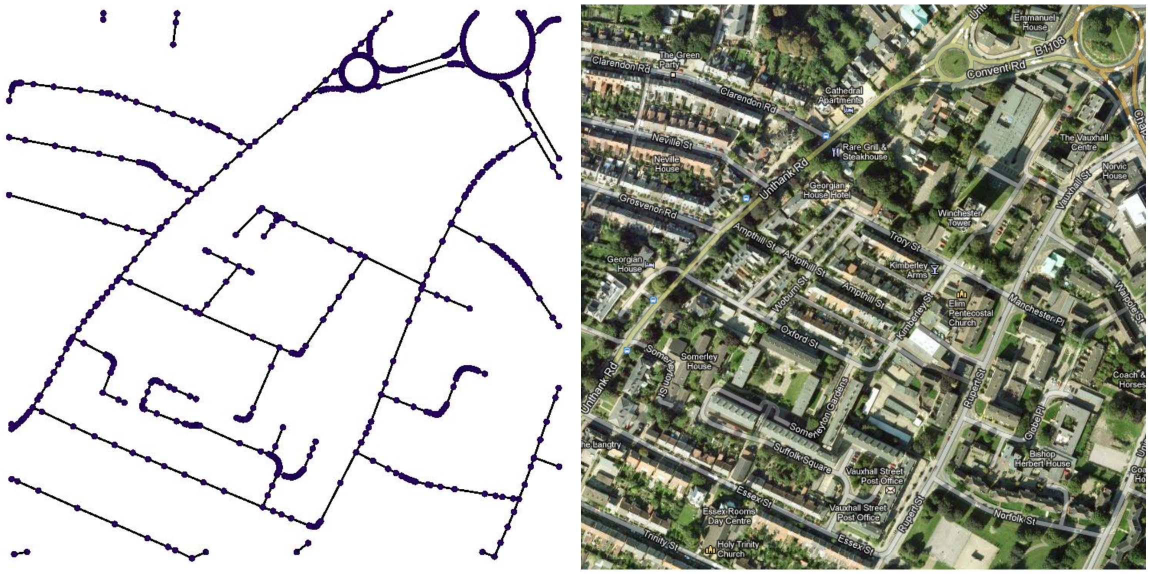
Figure 5.5: High-detail traffic simulation.

behaviour of a vehicle will remain constant in-between update intervals. One benefit of this model is that performance can be increased through the use of a more coarse discretisation, but at the cost of reducing behavioural realism. However, even at fine discretisations the simulation can appear visually disturbing, as vehicles appear to 'pop' from one cell to the next in synchronisation. To resolve this issue the position of each vehicle is interpolated between consecutive update steps so that vehicles appear to move on a continuous path. This technique could also be used for simulations that do not require high behavioural realism, as vehicles will exhibit continuous motion, even if, coarse discretisations are used. For vehicles that are completely outside of the view frustum, the original Nagel and Schreckenberg behaviour model could be used to update their behaviour at discrete time intervals.

5.2.1 Performance

To evaluate the performance of the low-detail traffic model, traffic was simulated on a virtual road network representing a 0.25 km^2 area of Norwich, United Kingdom (Figure 5.6). Performance was measured for simulations containing 50, 100, 200, 400, and 600 vehicles, and were performed on a computer with an Intel Core i5 3.2 GHz processor with 4 GB RAM. The simulation was written using the C++ programming language. All simulations were performed over 420 time steps (7 minutes at 1 Hz). For all results, the first 120 time steps have been discarded, as this data represents the initial transient stage of the simulation where the road network is populated with vehicles. The road graph used in the simulation consists of 38 roads and 74 junctions.

The performance of the system was analysed by measuring the computation times required to update the original Nagel and Schreckenberg traffic model (Behaviour A), and the times required to interpolate all vehicles in the proposed extension to the Nagel and Schreckenberg model (Behaviour B), which interpolates vehicle positions



(a) (b)

Figure 5.6: Traffic was simulated on a virtual road network representing a 0.25 km^2 area of Norwich, United Kingdom. (a) Ordnance Survey road centre line data for this region. (b) A hybrid satellite image of the same region. Imagery ©2013 DigitalGlobe, Getmapping Plc, Infoterra Ltd & Bluesky, and The GeoInformation Group. Map data ©2013 Google.

between cells to simulate a higher-level of realism. The results are presented in Table 5.1 and in Figure 5.7.

The results show that the average computation time required by the interpolation (Behaviour B) is less than 0.2 ms and it remains constant as the number of vehicles increases. Behaviour A is updated at discrete time intervals (1 Hz), and the interpolation is performed per-frame. Therefore, the additional computation required to exhibit continuous vehicle movement is typically bound by the larger of the two computation times, except in instances where the interpolation must be performed simultaneously with Behaviour A.

No. of vehicles	ρ'	Behaviour A Mean ($\pm\sigma$)	Behaviour B Mean ($\pm\sigma$)
50	0.07	0.383 $\pm$ 0.273	0.101 $\pm$ 0.024
100	0.14	0.501 $\pm$ 0.369	0.122 $\pm$ 0.026
200	0.28	0.550 $\pm$ 0.371	0.143 $\pm$ 0.034
400	0.55	0.840 $\pm$ 0.564	0.132 $\pm$ 0.038
600	0.83	2.576 $\pm$ 1.246	0.141 $\pm$ 0.041

Table 5.1: Computation times required to update the original Nagel and Schreckenberg traffic model (A), and the times required to interpolate all vehicles in the proposed extension to the Nagel and Schreckenberg model (B). ρ represents the desired network density. All times are shown in milliseconds.

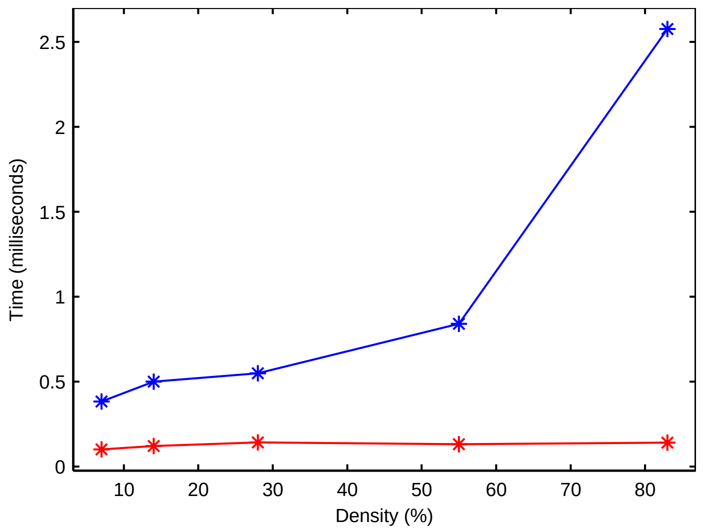


Figure 5.7: The computation times required to update the original Nagel and Schreckenberg traffic model (Behaviour A - blue), and the times required to interpolate all vehicles in the proposed extension to the Nagel and Schreckenberg model (Behaviour B - red).

5.3 Summary

This chapter presented traffic behaviour models that could be used in real-time simulations. For high-detail simulations, an agent-based traffic model was proposed which was influenced by the work of Gipps [Gip81]. In the presented model, each vehicle is represented as a two-dimensional entity that models properties including its current speed, desired speed, acceleration, deceleration, and braking rates. Time-based animation techniques are used to update the position of each vehicle as it traverses along a Catmull-Rom spline constructed between defined cell locations, and a set of rules were defined to model vehicle behaviour at junctions. Furthermore, the work of Nagel and Schreckenberg [NS92] was extended to produce a cellular automata model for low-detail simulations, using interpolation techniques to overcome the animation continuity issues that are typically associated with discrete behaviour models. The performance of the algorithm was evaluated with varying numbers of simulated vehicles. The results showed that the average computation time required for interpolation was less than 0.2 ms, which remained constant as the number of vehicles increased. Furthermore, it was concluded that due to the discrete nature of the Nagel and Schreckenberg cellular automata model, the inclusion of the interpolation technique does not significantly impact performance, as performance is typically bound by the component with the highest computational cost, except in instances where both behaviours must be updated simultaneously. The work presented in this chapter has provided methodologies to model large-scale traffic simulations at real-time frame rates, which can be applied to road networks generated by the techniques presented in previous chapters.

Chapter 6

Pedestrian behaviour

In this chapter Reynolds' work on steering behaviours for autonomous characters [Rey99] is extended to introduce a basic framework for modelling the interaction between vehicles and pedestrians in urban environments. This is no trivial task and therefore the aim of this chapter is to introduce the subject to motivate continued research in this area. Performance of the system is evaluated and it is demonstrated that the proposed system operates at interactive frame rates, even with a multitude of advanced rendering features.

In the proposed framework a pedestrian p is represented in the xz-plane as a two-dimensional circle whose radius encompasses the xz-projection of its geometry. Each pedestrian stores its normalised forward-vector, its normalised right-vector, and three intersection vectors that are used to steer the pedestrian away from the boundaries of the scene (Figure 6.1). One of the intersection vectors is aligned with the pedestrian's forward-vector and has a magnitude of 10 metres, the other two vectors are rotated $\pm\frac{\pi}{4}$ radians about the forward-vector and each has a magnitude of 5 metres. The lengths of the intersection vectors were assigned arbitrarily, but could be scaled proportionally with the speed of the pedestrian to increase its sight-distance when travelling at higher speeds.

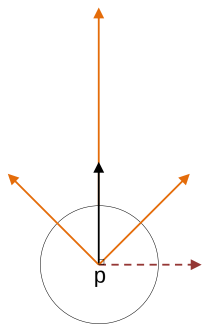


Figure 6.1: A pedestrian p is represented in the xz-plane as a two-dimensional circle whose radius encompasses the xz-projection of its geometry. Each pedestrian stores its normalised forward-vector (black arrow), its normalised right-vector (red arrow), and three intersection vectors (orange arrows).

6.1 Steering

When the behaviour model is updated, a steering vector is computed for each pedestrian to direct it towards its target whilst avoiding static and dynamic objects in the scene. The pedestrian is then rotated so that its forward-vector aligns with the generated steering vector. To prevent sudden and severe changes in direction, rotation is limited over time. In the system there is an upper bound on this limitation, $\theta = \frac{18}{5}\pi$ radians per second, selected arbitrarily.

When modelling pedestrian behaviour, four of Reynolds' steering behaviours are used to compute the steering vector:

1. Unaligned collision avoidance
2. Seek
3. Steer for separation
4. Containment

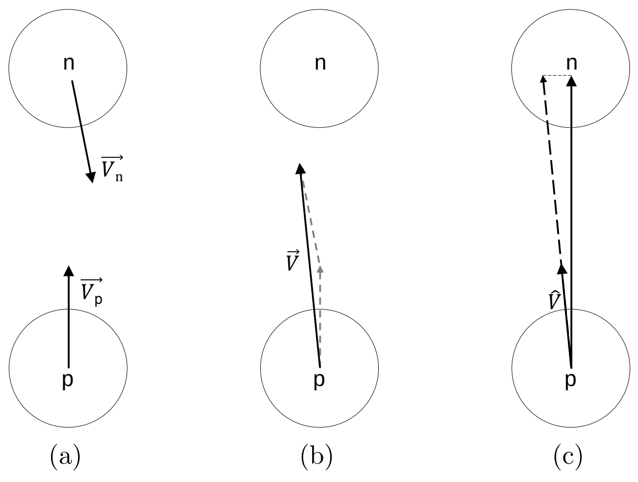


Figure 6.2: (a) The velocity vectors $\overrightarrow{V_p}$ and $\overrightarrow{V_n}$. (b) These vectors are defined as a single vector, $\overrightarrow{V}$, relative to p. (c) An offset vector from $\overrightarrow{P_p}$ to $\overrightarrow{P_n}$ is then projected onto the unit vector $\widehat{V}$ to produce a scalar which is divided by $|\overrightarrow{V}|$ to estimate the time until collision.

6.1.1 Unaligned collision avoidance

Unaligned collision avoidance is used to steer pedestrians to avoid collisions with other individuals in the scene. In the proposed behaviour model a pedestrian, p, will identify the neighbouring pedestrian that poses the most significant collision threat, T. If a threat exists, p will update its behaviour in an attempt to avoid the threat. When identifying the threat pedestrian, p will predict the time until collision for each of its neighbours based on current velocities as described below (Figure 6.2):

$$\overrightarrow{V} = \overrightarrow{V_p} - \overrightarrow{V_n}$$

$$t_n = \frac{(\widehat{V}) \cdot (\overrightarrow{P_n} - \overrightarrow{P_p})}{|\overrightarrow{V}|}, |\overrightarrow{V}| \neq 0$$

where $\overrightarrow{V_p}$ and $\overrightarrow{V_n}$ are the current velocities of pedestrian p and its neighbour n respectively, $\overrightarrow{P_p}$ and $\overrightarrow{P_n}$ represent vectors from the origin to the centre positions of pedestrians p and n respectively, and $\widehat{V}$ is the unit vector of $\overrightarrow{V}$. When $t_n \geq 0$, pedestrians p and n are either colliding or a collision could occur between these entities in the future. There is a special case when the magnitude of the relative velocity $|\overrightarrow{V}| = 0$. This occurs when the two pedestrians are travelling with the same velocity. If these pedestrians are colliding the value of t_n is set to 0, otherwise it is assumed the pedestrians will not collide and the value of t_n is set to a negative value ($t_n = -1$). If $t_n \in [0, tMax]$, the future positions of pedestrian p and its neighbour n are estimated after time t_n using the equation below:

$$\overrightarrow{P'} = \overrightarrow{P} + t_n \overrightarrow{V}$$

where $\overrightarrow{P'}$ represents the estimated future position of a pedestrian at position $\overrightarrow{P}$ with a constant velocity $\overrightarrow{V}$, after time t_n. If the Euclidean distance between the future positions of pedestrians p and n is less than a collision threshold, $cMax$, pedestrian n is considered as a candidate for the threat pedestrian. After all neighbouring pedestrians have been analysed, the candidate with the smallest value of t_n is identified as the threat pedestrian. In the system, $cMax = 2(r_p + r_n)$, where r_p and r_n are the bounding radii of p and n respectively. The value of $tMax = 6$ seconds, which was selected using a brute-force parameter optimisation. This process involved recording the number of inter-pedestrians collisions with varying values for this parameter. $tMax$ was evaluated between 0.5 and 10, in increments of 0.5. The parameter value that resulted in the fewest number of inter-pedestrian collisions was selected.

When updating the behaviour of pedestrian p to avoid the threat pedestrian, the dot product, d, is computed between the two pedestrians' forward vectors. If the pedestrians are not heading towards each other and are not travelling in a similar direction ($|d| < \cos(\pi/4)$), the dot product is computed between p's right-vector and T's forward-vector to determine pedestrian p's steering vector, as defined below

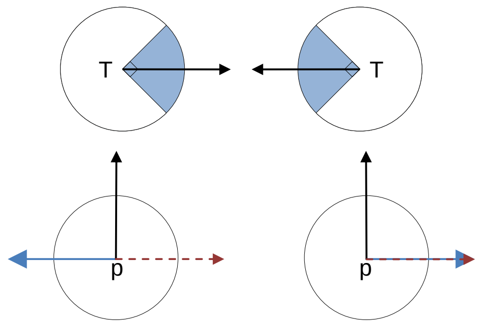


Figure 6.3: When pedestrian p and the threat pedestrian T are not heading towards each other and are not travelling in a similar direction (i.e. when the forward vector of T lies within the blue quadrant), the dot product is computed between p's right-vector (red arrow) and T's forward-vector (black arrow). The result of this calculation is then used to determine p's steering vector (blue arrow).

(Figure 6.3):

$$dp = \widehat{F_T} \cdot \widehat{R_p}$$

$$\widehat{v_{uca}} = \begin{cases} -\widehat{R_p} & \text{when } dp > 0 \\ \widehat{R_p} & \text{when } dp \leq 0 \end{cases}$$

where $\widehat{F_T}$ is the forward-vector of T, $\widehat{R_p}$ is the right-vector of p, and $\widehat{v_{uca}}$ is the returned steering vector for the 'unaligned collision avoidance' behaviour.

If pedestrians p and T are heading towards each other ($d \leq -\cos(\pi/4)$), the offset vector from p to the estimated position of T at the predicted time of collision, is projected onto p's right-vector to determine p's steering vector, as defined below (Figure 6.4):

$$dp = (\overrightarrow{P_{T'}} - \overrightarrow{P_p}) \cdot \widehat{R_p}$$

$$\widehat{v_{uca}} = \begin{cases} -\widehat{R_p} & \text{when } dp > 0 \\ \widehat{R_p} & \text{when } dp \leq 0 \end{cases}$$

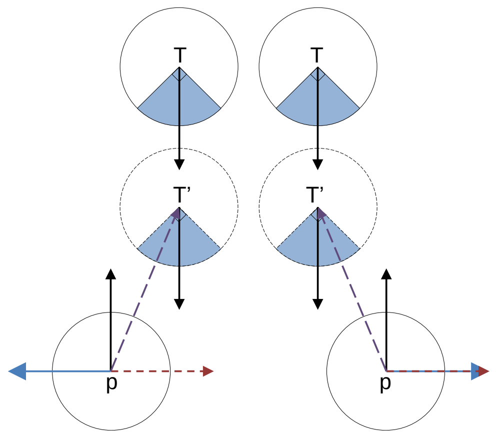


Figure 6.4: When pedestrians p and T are heading towards each other (i.e. when the forward vector of T lies within the blue quadrant), the dot product is computed between p's right-vector (red arrow) and the offset vector from p to the estimated position of T at the predict time of collision (purple arrow). The result of this calculation is then used to determine p's steering vector (blue arrow).

where $\overrightarrow{P_{T'}}$ is the estimated position of T at the predicted time of collision, $\overrightarrow{P_p}$ is the position of p, $\widehat{R_p}$ is the right-vector of p, and $\widehat{v_{uca}}$ is the returned steering vector for the 'unaligned collision avoidance' behaviour.

Alternatively, if pedestrians p and T are travelling in a similar direction ($d \geq \cos(\pi/4)$), the dot product is computed between p's right-vector and T's forward-vector to determine pedestrian p's steering vector, as defined below (Figure 6.5):

$$dp = \widehat{F_T} \cdot \widehat{R_p}$$

$$\widehat{v_{uca}} = \begin{cases} -\widehat{R_p} & \text{when } dp > 0 \\ \widehat{R_p} & \text{when } dp \leq 0 \end{cases}$$

where $\widehat{F_T}$ is the forward-vector of T, $\widehat{R_p}$ is the right-vector of p, and $\widehat{v_{uca}}$ is the returned steering vector for the 'unaligned collision avoidance' behaviour.

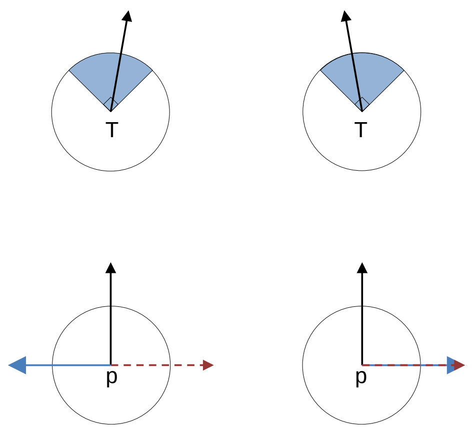


Figure 6.5: When pedestrians p and T are travelling in a similar direction (i.e. when the forward vector of T lies within the blue quadrant), the dot product is computed between p's right-vector (red arrow) and T's forward-vector (black arrow) to determine pedestrian p's steering vector (blue arrow).

6.1.2 Seek

The 'seek' steering behaviour is used to steer a pedestrian p towards a target tgt. An offset vector from p to tgt is projected onto the pedestrian's right-vector to determine the steering vector, as defined below (Figure 6.6):

$$dp = (\overrightarrow{P_{tgt}} - \overrightarrow{Pp}) \cdot \widehat{R_p}$$

$$\overrightarrow{v_{sek}} = \begin{cases} \widehat{R_p} & \text{when } dp > 0 \\ NULL & \text{when } dp = 0 \\ -\widehat{R_p} & \text{when } dp < 0 \end{cases}$$

where $\overrightarrow{P_{tgt}}$ is the position of the target relative to the origin, $\overrightarrow{P_p}$ is the position of pedestrian p relative to the origin, $\widehat{R_p}$ is the right-vector of pedestrian p, $NULL$ is vector of no magnitude, and $\overrightarrow{v_{sek}}$ is the returned steering vector for the 'seek' behaviour.

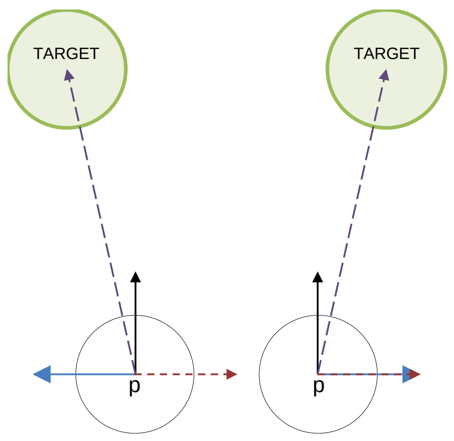


Figure 6.6: An offset vector between pedestrian p and the target is computed (purple arrow), which is then projected onto the pedestrian's right-vector (red arrow) to determine the steering vector (blue arrow).

6.1.3 Steer for separation

The 'steer for separation' behaviour is used to maintain a separation distance between a pedestrian p and its neighbours. For each neighbour n that lies within a certain radius of p, an offset vector is computed from n to p. Each offset vector is normalised and then divided by its original magnitude to produce a repulsion vector whose magnitude is inversely proportional to the distance from n to p. This scaling is applied to ensure that closer neighbours have a greater influence on the resulting steering direction. Repulsion vectors are summated and the resulting vector is projected onto p's right-vector to determine its steering vector, as defined below (Figure 6.7):

$$dp = \overrightarrow{rep} \cdot \widehat{R_p}$$

$$\overrightarrow{v_{sep}} = \begin{cases} \widehat{R_p} & \text{when } dp > 0 \\ NULL & \text{when } dp = 0 \\ -\widehat{R_p} & \text{when } dp < 0 \end{cases}$$

where $\overrightarrow{rep}$ represents the summation of all repulsion vectors, $\widehat{R_p}$ is the right-vector

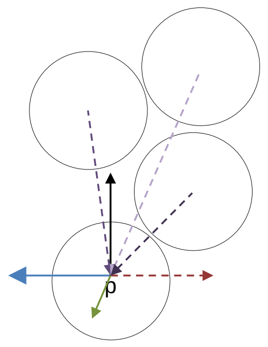


Figure 6.7: For each neighbour n that lies within a certain radius of p, an offset vector is computed from n and p (purple arrows). Each offset vector is normalised and then divided by its original magnitude to produce a repulsion vector whose magnitude is inversely proportional to the distance from n to p. Repulsion vectors are summated (green arrow) and the resulting vector is projected onto pedestrian p's right-vector (red arrow) to determine its steering vector (blue arrow).

of pedestrian p, $NULL$ is vector of no magnitude, and $\overrightarrow{v_{sep}}$ is the returned steering vector for the 'steer for separation' behaviour.

6.1.4 Containment

The 'containment' behaviour is used to steer pedestrians away from static objects and the boundaries of a scene (entry into and exit from the scene is permitted at designated locations). Each of pedestrian p's intersection vectors are tested for intersection against these boundaries. The boundary normal at the closest point of intersection along each vector is added to an accumulation vector, which is then projected onto p's right-vector to determine its steering vector, as defined below (Figure 6.8):

$$dp = \overrightarrow{acc} \cdot \widehat{R_p}$$

$$\overrightarrow{v_{con}} = \begin{cases} \widehat{R_p} & \text{when } dp > 0 \\ NULL & \text{when } dp = 0 \\ -\widehat{R_p} & \text{when } dp < 0 \end{cases}$$

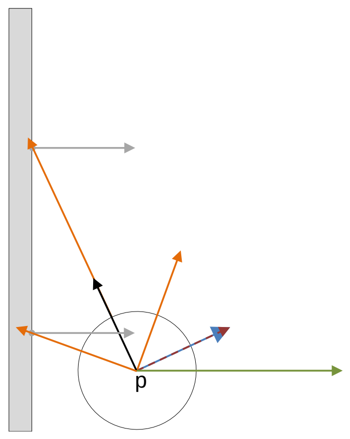


Figure 6.8: Each of pedestrian p's intersection vectors (orange arrows) are tested for intersection against static objects and the boundaries of the scene (grey area). The boundary normal at the closest point of intersection along each vector (grey arrows) is added to an accumulation vector (green arrow), which is then projected onto p's right vector (red arrow) to determine its steering vector (blue arrow).

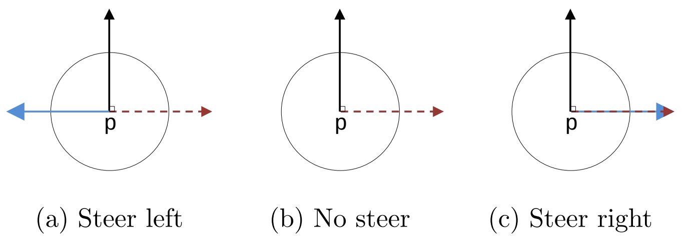


Figure 6.9: Each of the four behaviours will return one of the three steering requests listed above.

where $\overrightarrow{acc}$ represents the accumulation vector, $\widehat{R_p}$ is the right-vector of pedestrian p, $NULL$ is vector of no magnitude, and $\overrightarrow{v_{con}}$ is the returned steering vector for the 'containment' behaviour.

6.1.5 Combining behaviours

Each of the four behaviours will return one of the three steering requests listed in Figure 6.9. When all of the steering behaviours return the same request, the most suitable steering direction is obvious. However, there may be conflicting requests. For example, a pedestrian may need to turn left to avoid a collision with another pedestrian, but turn right to progress towards its target. A naive solution to handle a conflict would be to apply an average of all the steering requests. In the example above this would result in no change in direction and the two pedestrians would collide. Reynolds suggested prioritising the steering requests using a weighted average, but it is not necessarily obvious how priority should be assigned. When simulating the interaction between pedestrians there are two fundamental goals that each individual attempts to satisfy:

1. Travel towards the target via an optimal route

2. Minimise the number of collisions

If priority is assigned to route optimisation, a pedestrian will travel directly towards its target ignoring other pedestrians. In a densely populated scene, a large number of inter-pedestrian collisions is expected, reducing pedestrian flow. Adversely, if collision minimisation is assigned priority, a pedestrian may deviate from its route in order to avoid other pedestrians. In a densely populated scene a pedestrian may make successive deviations from its route, increasing travel times. In a worse-case scenario a pedestrian may travel in a circular path and never reach its destination. Therefore, prioritising any one of these goals can result in unrealistic behaviour. From general observation it appears that pedestrians are typically willing to deviate from an optimal route temporarily, in order to avoid collisions or crowded areas. However, significant deviations are less frequently undertaken, which suggests that priorities are dynamic over time and are situation dependant.

Each pedestrian in the simulation records the distance to its next target, $DTarget$, and the number of times it has collided with other pedestrians, $NCollisions$, over a time period t. Samples are taken every x seconds. This data provides a measure of success for the two goals defined above. For illustration purposes, a typical plot of

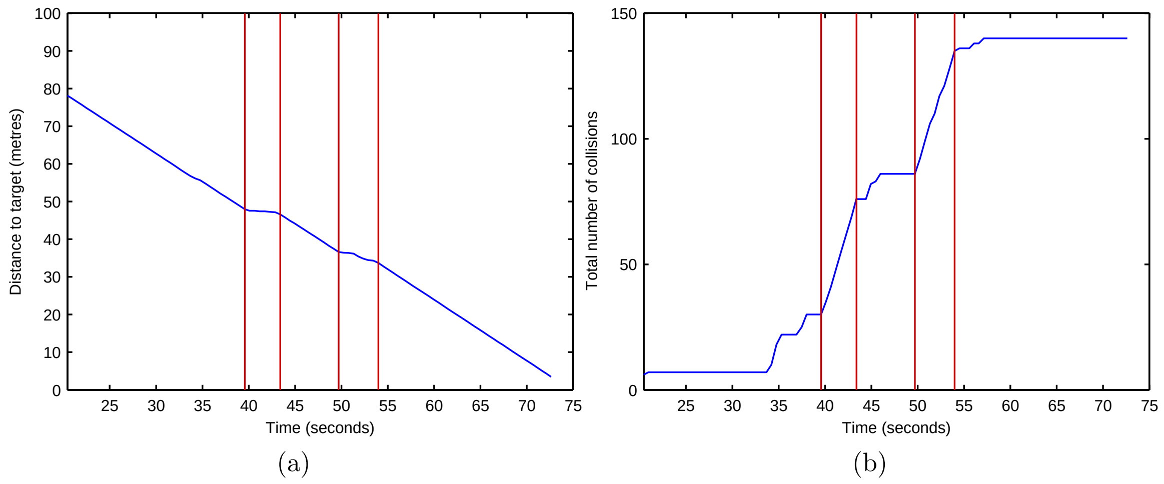


Figure 6.10: Each pedestrian records the distance to its next target (a) and the number of times it has collided with other pedestrians (b), over time. The time periods highlighted in red demonstrate that when a pedestrian is involved in a significant number of collisions within a short interval, little progress is made towards its target.

this data is shown in Figure 6.10.

Goal success can be established by considering the gradient of each of these plots between particular intervals. Figure 6.10(a) shows the distance a pedestrian is from its next target over time, and produces a gradient g_1 defined as $\frac{\Delta DTarget}{\Delta t}$. In this plot a large negative gradient is most desirable. Figure 6.10(b) shows the total number of collisions over time, and produces a gradient g_2 defined as $\frac{\Delta NCollisions}{\Delta t}$. In this plot a small gradient is most desirable. In the behaviour model a maximum acceptable gradient is defined for each measure of success. The maximum acceptable gradient for g_1 is $g_1 Max = -1$, and the maximum acceptable gradient for g_2 is defined as $g_2 Max = 3$. When a pedestrian computes its steering vector, it calculates these gradients over the last n samples taken. If $g_1 > g_1 Max$, a pedestrian is considered to be deviating too far from its intended route, and therefore its priority is increased for route optimisation by multiplying its weighting, w, by a factor of f. However, if $g_2 > g_2 Max$, the pedestrian is considered to be colliding with too many other pedestrians, and therefore w is multiplied by a factor of $1/f$ to decrease priority

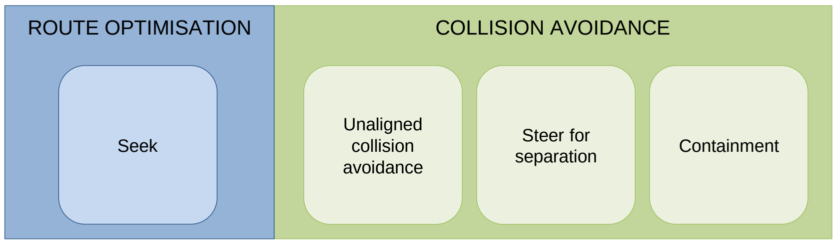


Figure 6.11: Each of the four steering behaviours is considered to prioritise either route optimisation or collision avoidance. Their classification is shown above.

for route optimisation (effectively increasing priority for collision avoidance). In the system $x = 0.5$ seconds, $n = 20$ samples, and $f = 2$.

Each of the four steering behaviours is considered to prioritise either route optimisation or collision avoidance. Classifications are shown in Figure 6.11, and are used to compute the steering vector, $\overrightarrow{s}$, as defined below:

$$\widehat{v} = \text{normalise}(\widehat{v_{uca}} + \overrightarrow{v_{sep}} + \overrightarrow{v_{con}})$$

$$\overrightarrow{s} = w(\overrightarrow{v_{sek}}) + \widehat{v}$$

6.2 Collision detection and response

Although the steering behaviour system is designed to guide pedestrians through the environment, it does not guarantee collision-free routes. Therefore, an efficient collision detection and response system has been implemented to prevent entities from intersecting. In the proposed system the scene is divided into a regular grid of cells, where each cell stores a dynamic list of pedestrians contained within its bounds. When computing a pedestrian's steering vector, only neighbouring pedestrians within the same cell and its adjacent cells are considered. The pedestrian's next desired position relative to the origin, $\overrightarrow{D_p}$, is defined below:

$$\overrightarrow{D_p} = \overrightarrow{P_p} + \Delta t \overrightarrow{V_p}$$

where $\overrightarrow{P_p}$ is the pedestrian's current position relative to the origin, $\overrightarrow{V_p}$ is the pedestrian's current velocity, and Δt is the time elapsed since the last update cycle. This position is then tested for collision with static and dynamic objects within the scene. If no collisions are identified the movement is applied, otherwise the pedestrian slides across the surface of the intersection plane, as described in [Fau03].

6.3 Interaction between pedestrians and vehicles

This section presents a novel framework that supports the interaction between vehicles and pedestrians in urban environments for use in real-time applications. In scenes that contain both vehicles and pedestrians, pedestrians in the system will operate in one of two modes:

1. Pedestrian mode
2. Interaction mode

Pedestrians operate in 'Pedestrian mode' when in pedestrianised areas or public pathways. In this mode steering behaviours are as defined in Section 6.1. A pedestrian will operate in 'Interaction mode' when in areas that are navigable by vehicles. When a pedestrian enters this mode, for example when a pedestrian crosses a road, several routing options are considered. Ideally the pedestrian would like to travel directly towards its next target to minimise travel time. However, when crossing a busy or dangerous road, a less direct route may be more desirable for increased safety. In the system, pedestrians in 'Interaction mode' will consider up to N_r routes when crossing a road, in an increasing order of travel time. The first route that is considered is a direct route to the target, and the last route considered is the shortest distance

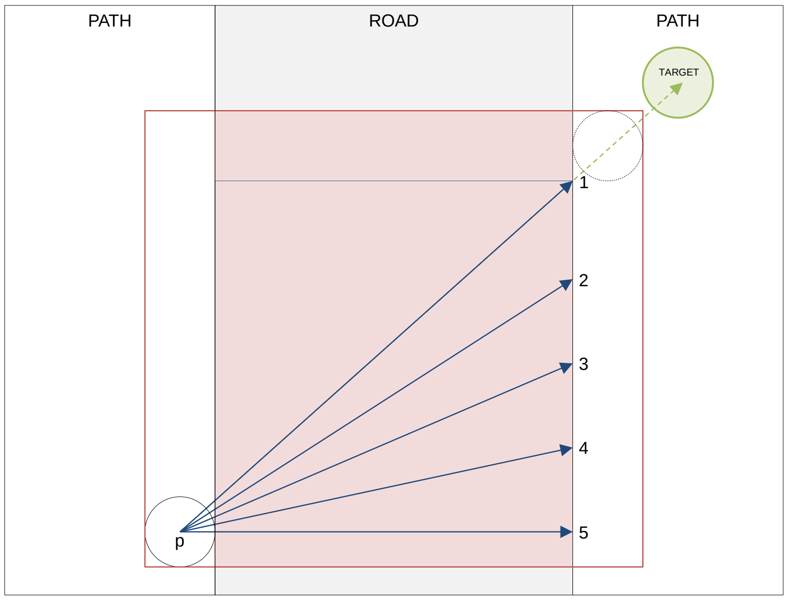


Figure 6.12: When pedestrian p crosses the road several routing options are considered (blue arrows). The first route is a direct route to the target (1), and the last route is the shortest distance across the road (5). When considering a route, a danger zone is defined as a bounding box that encapsulates the pedestrian for its entire journey across the road (pink region). This region is then tested for potential collisions with vehicles for the crossing duration to establish crossing suitability.

across the road (i.e. a route perpendicular to the road centre line). Using the first and last routes to define a bounding area, the remaining routes are computed using linear interpolation (Figure 6.12). In the system $N_r = 5$.

A pedestrian will consider each of these routes in order, until a collision-free route is found. When considering a route, a danger zone is defined as a bounding box that encapsulates the pedestrian for its entire journey across the road (Figure 6.12). This region is then tested for potential collisions with vehicles for the crossing duration to establish crossing suitability. The crossing duration is estimated using the length of the route and the pedestrian's current and desired velocities. A sweep test is used to establish whether any vehicle will enter the danger zone. If no route yields a collision-free path, the pedestrian will wait at the side of the road. A safety factor could be simulated by scaling the estimated crossing duration by a risk parameter, where a risk value greater than 1 would result in over-cautious behaviour, and a risk value less than 1 would result in pedestrians taking unnecessary risks.

If a pedestrian takes longer than expected to cross a road, it could increase the risk of a vehicle-pedestrian collision. This situation could occur when a pedestrian reduces their speed or takes a diversion to avoid colliding with other pedestrians. Therefore, vehicles are programmed to react to pedestrians in the same way that they react to other vehicles and will apply their brakes accordingly to avoid collision. To add further realism, if a pedestrian is within ϱ metres of a vehicle, it will ignore its crossing route and take a direct route towards the pavement. In the system, a value of 5 was selected for ϱ to produce results that appeared visually acceptable. However, further analysis on the interactions between vehicles and pedestrians may provide a greater insight into a more optimal value for this parameter.

6.4 Performance

The interaction between pedestrians and vehicles was simulated on a 100 metre stretch of road. The scene contained approximately 250,000 triangles. 5 vehicle and 5 pedestrian models were used. The Cal3D animation library was used to animate pedestrians. System performance was measured using a machine with an Intel Core i7 2.93 GHz processor with 8 GB RAM. The machine included an NVIDIA GeForce GTX 580 graphics card with 1.5 GB of dedicated video RAM. The simulation was written in C++ and utilises the OpenGL graphics library. To enhance the visualisation of the simulation, several real-time rendering techniques were implemented such as, environment mapping [BN76], normal-mapping [KL96, COM98], and deferred shading [DWS$^+$88]. Furthermore, Fast Approximate Anti-Aliasing (FXAA) is applied as a post-process to reduce aliasing artifacts [Lot09] (Figure 6.13). Even with a multitude of advanced graphical features, the simulation performs at interactive frame rates ($\geq$

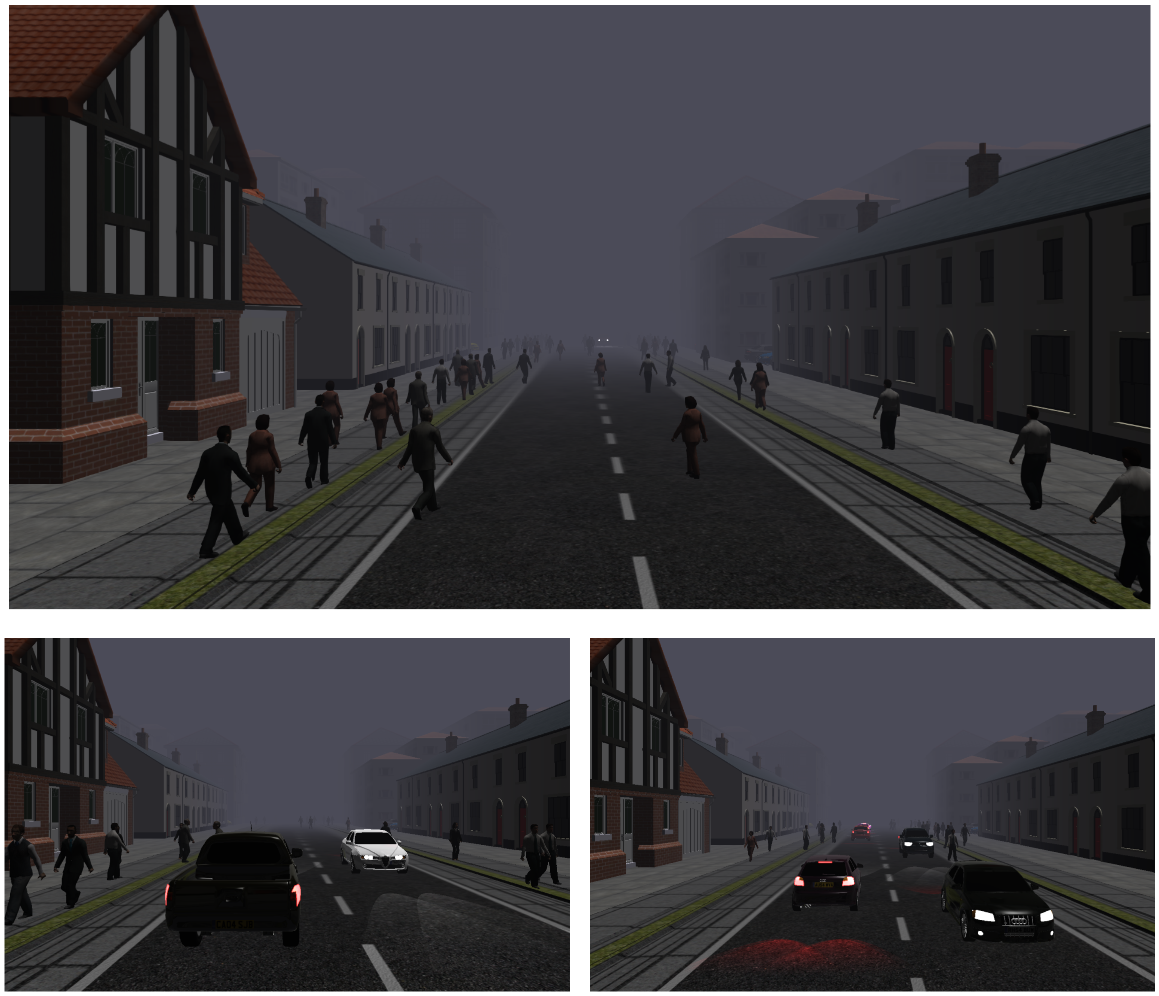


Figure 6.13: Interactions between vehicles and pedestrians.

20 FPS) as demonstrated in Table 6.1.

To better demonstrate the performance of the behaviour algorithm, the rendering process is decoupled from the behaviour model. The computation times required to update the behaviour of all pedestrians and vehicles in the simulation, for varying numbers of desired pedestrians and vehicles, are shown in Table 6.2.

The results show that the computational performance of the proposed model scales with the complexity of the simulated level of interaction, as expected. However, when simulating the mixed interactions between pedestrians and vehicles, computational

No. of pedestrians	Mean ($\pm\sigma$)
0	67.881 $\pm$ 11.660
25	62.191 $\pm$ 7.527
50	35.383 $\pm$ 0.930
75	25.815 $\pm$ 1.162
100	20.345 $\pm$ 0.711

Table 6.1: Frames-per-second (FPS) of the simulation with varying numbers of desired pedestrians. In all tests the desired number of vehicles was set to 10.

times are less than inter-pedestrian and inter-vehicle computational times combined. This phenomena is attributed to pedestrian and vehicle density. For example, as the pedestrian population increases, there is less traversable space on the path causing pedestrians to move onto the road. Vehicles reduce their speed and often come to a complete stop to avoid collision, enabling more pedestrians to cross the road. In this scenario, vehicle behaviour is greatly simplified. Alternatively, when traffic density increases, there is less opportunity for pedestrians to cross the road and pedestrian behaviour is greatly simplified. These factors contribute to the performance discrepancy.

6.5 Summary

In this chapter Reynolds' work on steering behaviours for autonomous characters [Rey99] was extended to introduce a basic framework for modelling the interaction between vehicles and pedestrians in urban environments. In the presented algorithm, each pedestrian is represented as a two-dimensional entity that stores its normalised forward-vector, normalised right-vector, and three intersection vectors that are used to steer the pedestrian away from the boundaries of the scene. The basic framework supports two modes for pedestrian behaviour: 'Pedestrian mode' and 'Interaction mode'. In 'Pedestrian mode', steering vectors are computed for each pedestrian for basic behaviours that include 'unaligned collision avoidance', 'seek', 'steer for separation', and

No. of pedestrians	No. of vehicles	Mean ($\pm\sigma$)
0	0	0.001 $\pm$ 0.000
0	2	0.006 $\pm$ 0.005
0	4	0.007 $\pm$ 0.004
0	6	0.009 $\pm$ 0.003
0	8	0.010 $\pm$ 0.003
0	10	0.011 $\pm$ 0.002
25	0	1.539 $\pm$ 0.376
25	2	1.567 $\pm$ 0.336
25	4	1.531 $\pm$ 0.297
25	6	1.502 $\pm$ 0.334
25	8	1.393 $\pm$ 0.213
25	10	1.356 $\pm$ 0.074
50	0	3.164 $\pm$ 0.665
50	2	3.164 $\pm$ 0.620
50	4	2.998 $\pm$ 0.421
50	6	3.179 $\pm$ 0.491
50	8	3.066 $\pm$ 0.543
50	10	2.781 $\pm$ 0.313
75	0	4.707 $\pm$ 0.927
75	2	4.732 $\pm$ 0.715
75	4	4.613 $\pm$ 0.533
75	6	4.402 $\pm$ 0.434
75	8	4.394 $\pm$ 0.520
75	10	4.561 $\pm$ 0.704
100	0	6.185 $\pm$ 1.169
100	2	6.223 $\pm$ 0.632
100	4	6.233 $\pm$ 0.663
100	6	6.320 $\pm$ 0.822
100	8	5.832 $\pm$ 0.385
100	10	6.023 $\pm$ 0.743

Table 6.2: Computation times required to update the behaviour of all pedestrians and vehicles in the simulation. Times are shown in milliseconds.

‘containment’, which are then prioritised and combined to produce final steering vectors that are used to update behaviour to avoid collisions with other pedestrians and boundaries of the scene. ‘Interaction mode’ extends ‘Pedestrian mode’ to model the interactions between vehicles and pedestrians. In this mode, several routing options are evaluated such that that a pedestrian will only cross a road when it is considered safe. To measure the performance of the model, an evaluation was conducted, which demonstrated that the system is able to operate at interactive frame rates (≥ 20 FPS), even with a multitude of advanced rendering features.

The aim of this chapter was to introduce the subject of vehicle-pedestrian interaction to motivate continued research in this area. The model presented is not a comprehensive solution, but provides a basic level of interaction, derived from well-established and recognised behaviour models for simulating vehicles and pedestrians independently. It is believed that this work provides a foundation for future projects that could be extended to develop behaviour models that offer a greater level of realism.

Chapter 7

Conclusions

Developing efficient and effective methodologies to design and simulate traffic networks within virtual environments is significant to applications across a multitude of industries. In the urban planning and transportation engineering industries, microsimulation software is widely used to design complex road networks and evaluate their effectiveness and suitability with an environment, prior to their actual construction. Furthermore, these software packages can be used to analyse the aggregate effects of driver behaviour in order to better understand the causes of traffic congestion and road collisions. In the entertainment industry, road networks and traffic simulations are integrated into film and video games to enhance the realism and believability of the simulated scenario. For each of these applications, a road network needs to be constructed within a virtual environment, and a traffic behaviour model needs to be designed and implemented to produce a suitable simulation.

7.1 Final summary

A variety of techniques for generating road networks in virtual environments have been explored. Procedural methods dominate the literature, and are able to produce extensive and complex road networks with minimal input. However, resulting networks appear to follow conventional city centre design patterns and require significant fine-tuning to obtain non-standard or customised layouts. Alternatively, sketch-based systems provide a greater level of control in the design process, but current techniques are limited to extremely flat environments.

Computational models used to simulate the behaviour of drivers and pedestrians have also been studied. Macroscopic traffic models are highly-suited to large-scale traffic simulations where aggregate approximations of vehicle speed, traffic flow, and traffic density can be estimated with minimal error. In contrast, microscopic traffic models provide greater accuracy, but are typically more computationally demanding. Mesoscopic models are able simulate traffic in high-detail and are more computationally efficient compared to microscopic alternatives. However, their discrete nature can make these models unsuitable for high-quality visualisation. For pedestrian simulation, rule-based crowd simulations can produce a variety of behaviours at moderately interactive frame rates, with more-complex behaviours demonstrated through the use of hierarchical structures and the incorporation of sociological effects. However, as with any rule-based system, the realism of the resulting behaviours are limited to the complexity and accuracy of the rules themselves. Additionally, the behaviour of the crowd has to be defined explicitly by the rules, which makes it difficult to author the overall characteristics of a crowd. Simulations based on cellular automata divide the traversable environment into a regular grid of cells, where each cell represents a discrete position within the environment that a pedestrian can occupy. Due to their discrete nature, these systems are usually highly parallelisable and therefore suitable for real-time applications. Additionally, coarse cell discretisations reduce both realism and accuracy of the model. Data-driven referenced behaviour systems can achieve high levels of realism, but have limited suitability for real-time applications, due to the costs associated with memory and query performance. Computational performance could be improved using a learned behaviour model by extracting parameters from the data. However, performance will need to be improved for the technique to work with large populations in real time.

The interactions between vehicles and pedestrians were also studied. The majority of reviewed work is primarily concerned with the analysis of recorded incidents, where reconstruction or statistical evaluation of these events can be used to improve road safety. There is little work that focuses on the development of behavioural models that simulate the interactions between vehicles and pedestrians, especially for scenarios where interaction does not result in an incident. Furthermore, a significant volume of work focuses on interactions at intersections or crossings where pedestrians typically adhere to crossing rules, but there appears to be an absence of research on developing behaviour models that simulate general interactions between vehicles and pedestrians in regions where pedestrian behaviour is less defined.

The research highlights several significant challenges in the field. Therefore, two challenges related to the design and simulation of traffic networks for use in virtual environments were identified and were investigated throughout this thesis:

1. The development of intuitive techniques to assist the design and construction of high-fidelity three-dimensional road networks for use in both urban and rural virtual environments.

2. The implementation of computational models to accurately simulate the behaviour of drivers and pedestrians in transportation networks, in real time.

Chapters 3 and 4 presented novel techniques to generate road networks for use in virtual environments. The algorithm presented in Chapter 3 enables users to automatically generate real-world road networks from digital map data. The input to the algorithm is a connected set of vertices and edges of an existing road network. Topology is determined using a set of rules that considers the number of connections at each vertex and the angle between connecting edges. A road graph suitable for a mesoscopic traffic behaviour model was then constructed by segmenting each road into a series of cells that can be used as waypoints during a simulation. The performance of the algorithm was evaluated by comparing the topologies of generated road networks with their real-world counterparts using OpenStreetMap data over 1 km^2 tile areas of the United Kingdom. The results indicated that the algorithm was able to correctly determine the layout of over 86% of all junctions in each of the real-world test scenes. If digital map information is not available, for example, when designing a new road infrastructure, an alternative solution is required. Therefore Chapter 4 presents a novel sketch-based tool to semi-automate the design, creation, and visualisation of road networks for virtual environments that is guided by input sketches and a combination of prioritised constraints, such as the curvature of roads, their inclination, and the volume of ground that would be displaced during construction. The technique extends previous work in the field by developing an algorithm that constrains the geometric properties of roads in three-dimensions to generate road networks across both flat and undulating terrain. Constraints can be tailored to generate roads that exhibit particular characteristics, such as roads with little inclination that cut through the environment and roads with high curvature that meander across the terrain. Furthermore, 'Influence Regions' are introduced, which are user-specified areas of the environment that influence the path of generated roads. These regions provide guidance throughout the road design process and can be tailored to either attract or repel roads to/from certain obstacles or designated areas, such as forestation, listed buildings, marshland, etc. A user study was conducted to evaluate the usability of the system and to evaluate the quality of roads generated in a diverse range of scenarios. The results demonstrated that the proposed system is both user-friendly and able to produce roads that are true to the user's intention.

Chapter 5 presented a model for simulating high-detail traffic networks in real time. The model presented is agent-based and operates on rules influenced by the pioneering work of Gipps [Gip81]. Furthermore, Nagel and Schreckenberg's cellular automata traffic model [NS92] has been extended to provide varying levels of low level-of-detail simulation. Chapter 6 extended Reynold's work on steering behaviours for autonomous characters [Rey99] to introduce a basic framework for modelling the interaction between vehicles and pedestrians in urban environments. The model presented is not a comprehensive solution to vehicle-pedestrian simulation, but provides a basic level of interaction, derived from well-established and recognised behaviour models for simulating vehicles [Gip81] and pedestrians [Rey99] independently. The aim of this chapter was to introduce the subject in order to motivate continued research in this area. It was demonstrated that the proposed system operates at interactive frame rates, even with a multitude of advanced rendering features.

7.2 Final conclusions

There are several applications that require road networks to be designed within a virtual environment. Two novel systems for creating road networks were presented in this thesis. The system presented in Chapter 3 enables user to replicate a real-world road network from digital map data, and the system presented in Chapter 4 permits the user to design roads from scratch using sketch-based tools. Each of these systems was designed for a different purpose. When developing algorithms and methodologies to design road networks, it is important to consider the context in which they are used, and their final application. Our system for generating road networks from digital map data was shown to correctly determine the layout of over 86% of all junctions in each of the test scenes. The algorithm performed with less accuracy in regions containing more complex structures such as roundabouts, which are key components of the road infrastructure in the United Kingdom, but are not as common in other countries, such as the United States. Therefore, when selecting software to design roads in virtual environments, context and application are critical factors. Likewise, there are a number of variable parameters in the sketch-based system presented in Chapter 4. If this software was employed by urban planners to design highly-realistic roads that meet government regulations, a different set of parameters would be used to those selected if the software was used within a video game environment to allow players to design their own challenging circuits for a car racing game. Often there is not a single set of optimal parameters and therefore general theories have been presented in this thesis which can be customised to meet the contextual requirements of the application of the software.

Furthermore, when designing and implementing a behaviour model for vehicles and/or pedestrians, there are a number of established behaviour models to select from. In Chapter 5, extensions to high and low-detail vehicle behaviour models were presented. The models vary in both accuracy and computational efficiency, but are suitable for different applications. For an application that is primarily focused on relaying details of aggregate road statistics, such as the average velocity of vehicles, traffic density, traffic flow, etc., it may not beneficial to provide a highly-detailed three-dimensional simulation. However, when redesigning a junction, a detailed simulation may be able to provide a more in-depth understanding of how it would impact critical factors, such as driver visibility. Therefore, implementation specifics are application dependant.

7.3 Future directions

There are several directions for future work in this field. As highlighted throughout this thesis, it is important to model the interactions between vehicles and pedestrians in high-detail traffic simulations. Techniques for modelling vehicle-pedestrian interaction were reviewed in Chapter 2, and Chapter 5 provided an introduction to the subject in order to motivate continued research. Future work could evaluate the behavioural realism of the presented model, and the results could be used to improve its accuracy. More complex pedestrian behaviour could be modelled using higher-level goals, such as meeting and chatting with friends, or sitting on a bench, as described in [ST07]. Secondary actions such as pointing, looking around, and talking on a cell phone, could also be introduced to add further realism to the simulation [LFCCO09]. The techniques presented could also be refined to optimise their computational performance so that larger-scale and more detailed simulations can operate within the constraints of real time performance. This could be achieved by exploiting the mass-scalability of the proposed models, performing updates in parallel and utilising acceleration techniques on the CPU and GPU simultaneously. Furthermore, different levels-of-simulation could be investigated where simplified behaviour models could be used for entities that are further from the view frustum. This would increase system performance without compromising visual quality.

An alternative future direction could focus on the further development of algorithms to generate road networks for virtual environments. A user study was conducted to evaluate the usability of the algorithm presented in Chapter 4 and to evaluate the quality of the roads generated in a diverse range of scenarios. The results indicated that the system is both user-friendly and able to produce roads that are true to the user's intention. To compliment this work, a more thorough user study could be conducted with experts in the field to evaluate its suitability for use in the urban planning industry. Generated roads could be evaluated against both formal and objective criteria, such as those described in [Hig02]. Future work could also incorporate real-world geographical data to automatically determine the shapes and weightings of ‘Influence Regions’ in the algorithm. The incorporation of this feature would rapidly increase system productivity and ultimately reduce development costs.

Bibliography

[AAAG95] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gaertner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.

[ALD11] C.S. Applegate, S.D. Laycock, and A.M. Day. A Sketch-Based System for Highway Design. *ACM/Eurographics: Sketch-Based Interfaces and Modeling, Vancouver, Canada*, 2011.

[BH97] D.C. Brogan and J.K. Hodgins. Group behaviors for systems with significant dynamics. *Autonomous Robots*, 4(1):137–153, 1997.

[BKM08] A. Bangor, P.T. Kortum, and J.T. Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[BKSZ01] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3 - 4):507 – 525, 2001.

[BN76] J.F. Blinn and M.E. Newell. Texture and reflection in computer generated images. *Commun. ACM*, 19(10):542–547, October 1976.

[Bro65] D.E. Broadbent. Information processing in the nervous system. *Science*, 150:457–462, 1965.

[Bro96] J. Brooke. SUS: a quick and dirty usability scale. *Usability Evaluation in Industry*, pages 189–194, 1996.

[BSAR09] A. Blessing, T.M. Sezgin, R. Arandjelovic, and P. Robinson. A Multimodal Interface for Road Design. In *2009 Intelligent User Interfaces Workshop on Sketch Recognition. ACM Press*, 2009.

[CEW$^+$08] G. Chen, G. Esch, P. Wonka, P. Müller, and E. Zhang. Interactive procedural street modeling. *ACM Trans. Graph.*, 27(3):103:1–103:10, August 2008.

[COM98] J. Cohen, M. Olano, and D. Manocha. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 115–122, New York, NY, USA, 1998. ACM.

[DH72] R.O. Duda and P.E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[Doy03] A. Doyle. The Two Towers. *Computer Graphics World*, 26(2), 2003.

[DWS$^+$88] M. Deering, S. Winner, B. Schediwy, C. Duffy, and N. Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, pages 21–30, New York, NY, USA, 1988. ACM.

[ELSVG08] A. Ess, B. Leibe, K. Schindler, and L. Van Gool. A mobile vision system for robust multi-person tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[EMP$^+$02] D.S. Ebert, F.K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and modeling: a procedural approach.* Morgan Kaufmann, 2002.

[EP09] C. Ennis and C. Peters. Modeling groups of plausible virtual pedestrians. 2009.

[EPO08] C. Ennis, C. Peters, and C. O'Sullivan. Perceptual evaluation of position and orientation context rules for pedestrian formations. In *Proceedings of the 5th symposium on applied perception in graphics and visualization*, pages 75–82, Los Angeles, California, 2008. ACM.

[Fau03] K. Fauerby. Improved collision detection and response, 2003.

[FO98] P. Felkel and S. Obdrzalek. Straight Skeleton Implementation. In *Proceedings of Spring Conference on Computer Graphics*, pages 210–218, 1998.

[Ger55] D.L. Gerlough. *Simulation of freeway traffic on a general-purpose discrete variable computer*. PhD thesis, University of California, Los Angeles, 1955.

[Gip81] P.G. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.

[Gip86] P.G. Gipps. A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 20(5):403–414, 1986.

[GPMG10] E. Galin, A. Peytavie, N. Maréchal, and E. Guérin. Procedural Generation of Roads. *Computer Graphics Forum (Proceedings of Eurographics)*, 29(2), 2010.

[HB94] J.K. Hodgins and D.C. Brogan. Robot herds: Group behaviors for systems with significant dynamics. In *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, page 319. MIT Press, 1994.

[Hea85] M.A. Heald. Rational Approximations for the Fresnel Integrals. *Math. Comp*, 44(170):459–461, 1985.

[Hig02] Highways Agency. *Design Manual for Roads and Bridges: Volume 6: Road Geometry*, February 2002.

[HM95] D. Helbing and P. Molnor. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282, 1995.

[ISSL09] K. Ismail, T. Sayed, N. Saunier, and C. Lim. Automated analysis of pedestrian-vehicle conflicts using video data. *Transportation Research Record: Journal of the Transportation Research Board*, 2140(-1):44–54, 2009.

[JJLC10] J. Jeon, H. Jang, S.B. Lim, and Y.C. Choy. A sketch interface to empower novices to create 3D animations. *Computer Animation and Virtual Worlds*, 21(3-4):423–432, 2010.

[JW06a] R. Jiang and Q.S. Wu. Interaction between vehicle and pedestrians in a narrow channel. *Physica A: Statistical Mechanics and its Applications*, 368(1):239–246, 2006.

[JW06b] R. Jiang and Q.S. Wu. Interaction between vehicle and pedestrians in a narrow channel. *Physica A: Statistical Mechanics and its Applications*, 368(1):239 – 246, 2006.

[JW07] R. Jiang and Q.S. Wu. Pedestrian behaviors in a lattice gas model with large maximum velocity. *Physica A: Statistical Mechanics and its Applications*, 373:683–693, 2007.

[KKN$^+$04] A. Kirchner, H. Klüpfel, K. Nishinari, A. Schadschneider, and M. Schreckenberg. Discretization effects and the influence of walking speed in cellular automata models for pedestrian dynamics. *Journal of Statistical Mechanics: Theory and Experiment*, 2004:P10011, 2004.

[KL96] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd annual conference on Computer* *graphics and interactive techniques*, SIGGRAPH '96, pages 313–324, New York, NY, USA, 1996. ACM.

[KM07] G. Kelly and H. McCabe. Citygen: An interactive system for procedural city generation. In *Fifth International Conference on Game Design and Technology*, pages 8–16, 2007.

[KNS03] A. Kirchner, K. Nishinari, and A. Schadschneider. Friction effects and clogging in a cellular automaton model for pedestrian dynamics. *Physical review E*, 67(5):56122, 2003.

[LAA05] C. Lee and M. Abdel-Aty. Comprehensive analysis of vehicle-pedestrian crashes at intersections in florida. *Accident Analysis & Prevention*, 37(4):775 – 786, 2005.

[LCF05] Y. Lai, S. Chenney, and S. Fan. Group motion graphs. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 281–290, Los Angeles, California, 2005. ACM.

[LCHL07] K. Lee, M. Choi, Q. Hong, and J. Lee. Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 109–118, San Diego, California, 2007. Eurographics Association.

[LCL07] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. In *Computer Graphics Forum*, volume 26, pages 655–664, 2007.

[LD03] R.G. Laycock and A.M. Day. Automatically generating large urban environments based on the footprint data of buildings. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, SM '03, pages 346–351, New York, NY, USA, 2003. ACM.

[LFCCO09] A. Lerner, E. Fitusi, Y. Chrysanthou, and D. Cohen-Or. Fitting behaviors to pedestrian simulations. In *Proceedings of the 2009 ACM*

SIGGRAPH/Eurographics Symposium on Computer Animation, pages 199–208. ACM, 2009.

[LGJZ09] X.G. Li, Z.Y. Gao, B. Jia, and X.M. Zhao. Modeling the Interaction Between Motorized Vehicle and Bicycle by Using Cellular Automata Model. *International Journal of Modern Physics C*, 20:209–222, 2009.

[Lie12] M. Liebl. John Carmack guarantees Xbox 720, PS4 will target 30 FPS, Dec 2012. http://www.gamezone.com/.

[Lot09] T. Lottes. FXAA. Technical report, NVIDIA, 2009.

[LW55] M.J. Lighthill and G.B. Whitham. On Kinematic Waves. II. A Theory of Traffic Flow on Long Crowded Roads. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, pages 317–345, 1955.

[LWR+04] T. Lechner, B. Watson, P. Ren, U. Wilensky, S. Tisue, and M. Felsen. Procedural modeling of land use in cities. 2004.

[LWW03] T. Lechner, B. Watson, and U. Wilensky. Procedural city modeling. In *In 1st Midwestern Graphics Conference*, 2003.

[MDDZ07] J. Meng, S. Dai, L. Dong, and J. Zhang. Cellular automaton model for mixed traffic flow with motorcycles. *Physica A: Statistical Mechanics and its Applications*, 380(0):470 – 480, 2007.

[MH04] R. Metoyer and J. Hodgins. Reactive pedestrian path following from examples. *The Visual Computer*, 20(10):635–649, 2004.

[Mil56] G.A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81–97, 1956.

[MS09a] J. McCrae and K. Singh. Sketch-Based Path Design. In *Proceedings of Graphics Interface 2009*, pages 95–102. Canadian Information Processing Society, 2009.

[MS09b] J. McCrae and K Singh. Sketching piecewise clothoid curves. *Computers & Graphics*, 33(4):452 – 461, 2009.

[MT01] S. Musse and D. Thalmann. Hierarchical model for real time simulation of virtual human crowds. In *Biomedical Imaging, 2002. 5th IEEE EMBS International Summer School on*, page 13, 2001.

[MWH$^+$06] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. In *ACM SIGGRAPH 2006 Papers*, pages 614–623. ACM, 2006.

[New61] G.F. Newell. Nonlinear effects in the dynamics of car following. *Operations Research*, 9(2):209–229, 1961.

[NHS05] A. Nakayama, K. Hasebe, and Y. Sugiyama. Instability of pedestrian flow and phase structure in a two-dimensional optimal velocity model. *Phys. Rev. E*, 71:036121, Mar 2005.

[NS92] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *J. Phys. I France*, 2:2221–2229, 1992.

[NT99] K. Nishinari and D. Takahashi. A new deterministic CA model for traffic flow with multiple states. *Journal of Physics A: Mathematical and General*, 32:93–104, 1999.

[NWWS98] K. Nagel, D.E. Wolf, P. Wagner, and P. Simon. Two-lane traffic rules for cellular automata: A systematic approach. *Physical Review E*, 58(2):1425–1437, 1998.

[Ols04] J. Olsen. Realtime procedural terrain generation. *Department of Mathematics And Computer Science (IMADA) University of Southern Denmark*, 2004.

[Par12] K. Parrish. John Carmack: Next-Gen Consoles Will Still Target 30 FPS, Dec 2012. http://www.tomshardware.com/.

[PEM08] C. Peters, C. Ennis, and R. McDonnell. Crowds in context: Evaluating the perceptual plausibility of pedestrian orientations. 2008.

[Per85] K. Perlin. An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3):287–296, 1985.

[PL90] P. Prusinkiewicz and A. Lindenmayer. The algorithmic beauty of plants. *New York*, 1990.

[PM01] Y.I.H. Parish and P. Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 301–308, New York, NY, USA, 2001. ACM.

[PPD07] S. Paris, J. Pettre, and S. Donikian. Pedestrian reactive navigation for crowd simulation: a predictive approach. 2007.

[Rey87] C. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34. ACM, 1987.

[Rey99] C. Reynolds. Steering behaviors for autonomous characters. 1999.

[Rey06] C. Reynolds. Big fast crowds on PS3. In *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, page 121. ACM, 2006.

[Rey10] D. Reyonlds. High Fidelity Simulation of Urban Terrains, 2010.

[RFD05] G. Ryder, P. Flack, and A.M. Day. A framework for real-time virtual crowds in cultural heritage environments. *Vast 2005, Short Papers Prceedings*, pages 108–113, 2005.

[Ric56] P.I. Richards. Shock Waves on the Highway. *Operations research*, 4(1):42–51, 1956.

[RNSL96] M. Rickert, K. Nagel, M. Schreckenberg, and A. Latour. Two lane traffic simulations using cellular automata. *Physica A: Statistical and Theoretical Physics*, 231(4):534–550, 1996.

[SBFF11] H.B. Shitrit, J. Berclaz, F. Fleuret, and P. Fua. Tracking multiple people under global appearance constraints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 137–144. IEEE, 2011.

[SM06] N. Siebel and S. Maybank. Fusion of multiple tracking algorithms for robust people tracking. *Computer Vision - ECCV 2002*, pages 373–387, 2006.

[SMK05] T. Sakuma, T. Mukai, and S. Kuriyama. Psychological model for animating crowded pedestrians. *Computer Animation and Virtual Worlds*, 16(3/4):343, 2005.

[ST07] W. Shao and D. Terzopoulos. Autonomous pedestrians. *Graphical Models*, 69(5 - 6):246 – 274, 2007. Special Issue on SCA 2005.

[SWML10] J. Sewall, D. Wilkie, P. Merrell, and M.C. Lin. Continuum Traffic Simulation. In *Computer Graphics Forum*, volume 29, pages 439–448. John Wiley & Sons, 2010.

[TLL$^+$11] J.O. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun. Metropolis procedural modeling. *ACM Trans. Graph.*, 30, April 2011.

[TT94] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, page 50. ACM, 1994.

[UT01] B. Ulicny and D. Thalmann. Crowd simulation for interactive virtual environments and VR training systems. In *Computer Animation and Simulation 2001: Proceedings of the Eurographics Workshop in Manchester, UK, September 2-3, 2001*, page 163, 2001.

[UT02a] B. Ulicny and D. Thalmann. Crowd simulation for virtual heritage. In *Proc. First International Workshop on 3D Virtual*, pages 28–32, 2002.

[UT02b] B. Ulicny and D. Thalmann. Towards interactive Real-Time crowd behavior simulation. *Computer Graphics Forum*, 21(4):767–775, 2002.

[XGZL09] D.F. Xie, Z.Y. Gao, X.M. Zhao, and K.P Li. Characteristics of mixed traffic flow with non-motorized vehicles and motorized vehicles at an unsignalized intersection. *Physica A: Statistical Mechanics and its Applications*, 388(10):2041 – 2050, 2009.

[XLC$^+$11] J. Xu, Y. Li, X. Chen, D. Ge, B. Liu, M. Zhu, and T. Park. Automotive windshield pedestrian head impact: Energy absorption capability of interlayer material. *International Journal of Automotive Technology*, 12:687–695, 2011. 10.1007/s12239-011-0080-2.

[XLLZ09] J. Xu, Y. Li, G. Lu, and W. Zhou. Reconstruction model of vehicle impact speed in pedestrian-vehicle accident. *International Journal of Impact Engineering*, 36(6):783 – 788, 2009.

[ZJG07] X.M. Zhao, B. Jia, and Z.Y. Gao. A new approach for modelling mixed traffic flow with motorized vehicles and non-motorized vehicles based on cellular automaton model. *Arxiv preprint arXiv:0707.1169*, 2007.