An agent-based framework for modelling the impact of landscape change on recreational behaviour

Author(s):
Cavens, Stuart Duncan

Publication Date:
2008

Permanent Link:
https://doi.org/10.3929/ethz-a-005779318

Rights / License:
In Copyright - Non-Commercial Use Permitted
An agent-based framework for modelling the impact of landscape change on recreational behaviour

A dissertation submitted to
ETH ZURICH

for the degree of
Doctor of Sciences

presented by
STUART DUNCAN CAVENS
M. Sc., University of British Columbia
born July 21, 1975
citizen of Canada

accepted on the recommendation of
Prof. Dr. W. A. Schmid, examiner
Prof Dr. K. Nagel, co-examiner

2008
# Contents

Abstract ................................................................. v
Zusammenfassung ....................................................... vii

1 Introduction ............................................................ 3
   1.1 Introduction to Pilot Study: Schönried, Switzerland .......... 6
   1.2 Dissertation Outline ........................................... 7

2 Related Work .......................................................... 9
   2.1 Recreation Modelling ............................................ 9
   2.2 Pedestrian Modelling for Evacuation Simulation .............. 12
   2.3 Vehicular Simulation ............................................ 13
   2.4 Landscape Perception Research ................................. 15

3 The Framework - Overall Simulation Approach .................... 19
   3.1 Overview ......................................................... 19
   3.2 An Iterative Approach .......................................... 20
   3.3 A Modular Approach ............................................ 21
      3.3.1 Communicating between Modules: the Event Broker . 23
   3.4 Description of Key Module Types .............................. 24
   3.5 Setting up and Running the Model .............................. 26
      3.5.1 Geographic (GIS) Data Requirements .................... 26
      3.5.2 Synthetic Population of Agents .......................... 27
      3.5.3 Agent Characteristics .................................... 27
      3.5.4 Activity Chains ............................................ 30
      3.5.5 Running the Model ........................................ 31
7.3.3 Start Times ................................................................. 73
7.4 Calibrating the Model ..................................................... 73
7.5 Testing the Model on New Scenarios ................................. 74
  7.5.1 Scenario Results ..................................................... 76
  7.5.2 Model Performance .................................................. 78

8 Conclusions ................................................................. 81
  List of Figures ............................................................... 82
  Bibliography ............................................................... 86
  Curriculum Vitae .......................................................... 95
Abstract

This work presents an agent-based framework for modelling how a changing landscape can impact the recreational value of the landscape. The framework, nicknamed Vizagent, simulates the movements of individual pedestrians as they move through the landscape. They interact with the landscape, both physically and visually, in order to ascertain if the particular recreational area meets their recreational needs.

The framework is a significant advancement over other existing agent-based recreational models, in that it explicitly allows for different transportation modes (such as walking, chairlifts and other forms of public transportation) within the same simulation. It is also unique in that it integrates visibility analyses and visual quality modeling within the agents’ decision making process.

The agents employ an iterative approach to learning about their environment where the agents begin with no knowledge of the simulated landscape. As they explore the landscape, the simulation learns about the area’s capacity to support the agents’ individualized expectations.

The framework, while designed to be flexible enough to represent different kinds of recreational landscapes, is demonstrated using a pilot study in south western Switzerland. The study area is modelled under three very different future conditions representing different possible planning futures. The movements of the agents demonstrate that the framework is able to react in a realistic way to real planning situations.
Zusammenfassung


Dieser Rahmen ist ein bedeutsamer Fortschritt im Vergleich zu derzeitig existierenden Agenten-basierten Erholungsmodellen, da er ausdrücklich verschiedene Transportmöglichkeiten (wie zum Beispiel zu Fuß gehen, Sessellift und andere Arten von öffentlichem Transport) in einer Simulation in Betracht ziehen kann. Er ist außerdem einzigartig darin, daß er die Analyse der Sicht und die Modellierung der visuellen Qualität innerhalb des Entscheidungsprozesses des Agenten integriert.

Die Agenten nutzen einen iterativen Lernprozess, um sich mit ihrer Umwelt vertraut zu machen, wobei sie ohne jegliches Grundwissen über die simulierete Landschaft beginnen. Während sie die Landschaft erkunden, lernt die Simulation, ob die Kapazität des Gebietes die individuellen Ansprüche der Agenten unterstützen kann.

Obwohl der Rahmen flexibel genug entworfen wurde um verschiedene Arten von Erholungslandschaften zu repräsentieren, wurde er in einer Pilotstudie im Südwesten der Schweiz demonstriert. Das Studiengebiet wurde unter drei verschiedenen Zukunftsszenarien modelliert, welche verschiedene Planungsmöglichkeiten für die Zukunft darstellen. Die Bewegung der Agenten beweist, daß der Rahmen fähig ist, zu wirklichen Planungssituationen real zu reagieren.
Chapter 1

Introduction

Managing recreational landscapes is a complex task. Most, if not all, recreational landscapes are managed for multiple, and often conflicting, priorities. These priorities range from providing high quality recreational experiences to maintaining ecological values to creating economic opportunities for landowners. Understanding how proposed changes impact all of these priorities simultaneously is a key task for resource managers. A critical priority for recreational areas, and one that has been very difficult to measure, is understanding how any changes to the landscape affect the choices and satisfaction of recreational visitors: do policies that change a landscape’s structure and composition change hikers’ experience? If yes, does it change it enough to change their route choice, and, would it influence their decision to return next year? While it is easy to anticipate how significant changes, such as the siting of a major new industrial facility in the middle of a significant recreational area, would impact an area’s attractiveness for recreation, the impact of more subtle changes, such as changes to vegetation management policies or changes to an area’s transportation infrastructure, are more difficult to predict.

A key problem with understanding the impacts of landscape change on recreational behaviour is that the entire subject area is, by definition, complex. While it is obvious that scenic quality has a large impact on a hikers’ choice of location (given that there are more hikers in scenic areas than in areas with lesser scenic quality), once one moves down from coarse choices at the regional scale, it is unclear how visitors prioritize the many factors that influence their choice of specific trails. There are many variables which influence a hiker’s path choices: these include physical factors (i.e. steepness, views), infrastructure factors (number of path choices, accessibility by public transport), as well as social factors (such as the number of other hikers that one encounters, past experiences with the same area, and even cultural attitudes towards different kinds of landscape.) While it is possible to study how each of these factors influence hikers’ decision-making, (i.e. [74], [14]), or even a few factors simultaneously using choice experiments(i.e. [35]), it is very difficult to design a feasible study design that would allow one to examine all of these factors simultaneously using traditional inductive social science research methods.
Chapter 1. Introduction

Computer simulation, however, offers an opportunity to test how many different variables interact with each other, and allows the researcher to develop a framework in which to test different hypotheses. There is a recognition in the social science literature that simulation is a new method for doing science [3]—and in many cases the only way to answer complex behavioural questions about groups of individuals.

There has recently been a revival in the use of computer simulation in many research areas related to natural resource management, including recreation. Encouraged by the rapidly increasing computing resources available to researchers, and by the dispersion of theoretical and technical ideas from other disciplines, increasingly complex models are being developed to assist researchers and resource managers understand the implications of different management options [76].

A particularly powerful technique that has been begun to be used in recreation modelling is individual agent-based modelling [49]. Using this technique, software agents, each representing an individual or small groups of individuals, are given individual goals, preferences and attributes. Using GIS formats and tools, a synthetic landscape is created that represents the landscape being simulated. Depending on the modeller’s goals, the synthetic landscape can either represent the current state of the recreational area, or a scenario that represents a future changed landscape. A set of rules is developed by the modeller which describe how the agents react to each other and to their environment. These agents are then introduced into a synthetic environment where they strive to complete their goals. They interact with each other and the environment, and make decisions (in the case of recreation modelling, this is usually their movement choices) based on their individual experiences. The modeller can observe how the agents react, either as individuals or as a system. By changing either the modelled environment or the calibration variables, the modeller can explore how the agents react. As Itami, Raulings et al. [40] describe, an advantage of this technique is that often complex system behaviour emerges that is difficult or impossible to predict based on the actions of the individuals. This is particularly true when individuals’ behaviour is highly dependent on their interaction with other individuals in the area.

As described in Chapter 2, existing models for simulating recreational behaviour, including the few agent-based ones, were designed to simulate relatively simple behaviour in areas where recreational demand exceeds the location’s infrastructure capacity. These models are limited to examining how changes to the available capacity of the recreation infrastructure (such as trails, campsites, and parking lots) impact the experience of users. This kind of model, while very useful for certain questions and applications, assumes that recreational infrastructure is the limiting factor that influences recreational choice.

Admittedly, this is often the case in wilderness areas, especially where visitors have a low tolerance for encounters with other people. It is also more likely to be the case in wilderness areas in North America, where most existing recreational models originated. However, in many recreational landscapes, particularly those that are not uniquely attractive or are facing non-recreational development pressures, the situation is far more complex. For communities depen-
dent on tourism, and in particular those not operating at capacity, the concern is often how the qualities of the surrounding landscape, and any proposed land use changes (such as increased development or changes to agricultural policy) will affect the experiences of their visitors.

Requiring a model to include the effects of landscape perception significantly increases the complexity of the simulation system. Unlike existing recreational models (see section 2.1) that assume that agents’ desired destinations and paths are constant, it requires a model that can adapt an agents’ desired movements to their perceptions and individual desires. This requires a simulation system that is sensitive to the agents’ landscape perceptions.

Others, in particular Bishop [9] [8] [41], have described the need for a recreational modelling system that incorporates agent perception and has advocated for additional research to complete one. To our knowledge, however, there has been to date no complete implementation that allows the modeller to explore how agents actually react to perceptual data and alter their movement decisions.

This dissertation presents one approach to how one could build such a simulation system. It is intended as a proof of concept, demonstrating the feasibility and complexity required in such an approach, rather than as usable tool in a planning context. While this is certainly the long term goal, the complexity of the modelling decisions required to make such a system functional and credibly calibrated mean that substantial basic research is still needed.

This dissertation describes an advanced agent-based modelling framework designed to simulate hiking behaviour within more typical recreational landscapes, where individuals have a wide range of route choices. It is designed to allow the modeller to simulate the simultaneous movements of many individuals, and model how they change their behaviour based on their explorations of a simulated landscape and on their interactions with other individuals. Unlike other existing recreational frameworks, this one is designed around the modeller specifying agent properties and expectations, rather than specific origins, destinations and routes. This results in a simulation where agent movements are truly responsive to changes in the simulated landscape. The framework also explicitly includes functionality that allows the agents to “see” their surroundings, and use this visual information to evaluate their experiences.

As a result, this framework has been nickname the ‘Vizagent’ framework, as it is the only recreational agent-based modelling system that explicitly models what agents can see.

Other significant advances between the framework described here and other existing recreational modeling frameworks include:

- a hierarchical approach to modelling agents’ decision-making, allowing for the simultaneous simulation of tactical and physical decision-making processes;
- integration of multiple physical simulations, which allows one to simulate recreation trips that use more than one mode of transportation in a single trip;
- a well-defined modular structure, which allows researchers to easily replace algorithms
and sub-models in order to test alternative modelling approaches.

1.1 Introduction to Pilot Study: Schönried, Switzerland

This work provides an overview of the Vizagent framework, and presents a pilot implementation that was applied to specific test landscape in the Gstaad-Saanenland region of Switzerland.

The specific test site is a valley in the Gstaad-Saanenland region of south-western Switzerland. The pilot area is a valley surrounding the communities of Schönried and Saanenmöser; their economies are highly tourism dependent. While the primary tourism draw to the area used to be winter skiing, long term climate change is forcing the community to focus its efforts on building up a more diversified tourism economy. This includes capitalizing on its already strong reputation for summer hiking. The landscape is a mixture of pasture and coniferous forests. The test site is characterised by significant topography and is considered ideal for walking and hiking. The trails are very accessible to a wide range of hiking abilities due to the summer operation of one chair-lift and two gondolas. In the high season, the area is busy with hikers and walkers who easily fill the two main parking lots in Schönried.

A recent study in the area [52] identified that the biggest attraction for summer tourists are the

Figure 1.1: Representational photograph of the study site. The site is a typical mix of forested areas mixed with grazing areas for cattle. Many of the grassy areas double as downhill skill trails in winter.
area’s scenic qualities. Hiking and walking is the primary recreational activity in the summer months. The focus on visual elements was confirmed by our own study (see section 7.2.1), where views and landscape variety were identified as the most important factors that influenced hikers in their choice of hiking routes.

The area represents a fairly typical recreational area, in that it’s recreational attractiveness is somewhat typical of the surrounding areas. While many hikers visit on a summer day, it is not internationally known as a hiking destination: its visitors tend to be regional or long time visitors. Unlike areas typically simulated using recreational simulators (which are typically wilderness areas in North America where the desire to avoid encounters with others is extreme), there is not a great concern about the number of encounters with other visitors given current use levels.

In addition to the community’s desire to diversify its recreational economy, there are landscape policy issues that have the potential to change the desirability of the area for summer tourism. These issues include changes to the pattern of the landscape due to changing agricultural policy, shifts in forestry practices, closing of the gondolas and/or chairlifts, and increased holiday home construction. All of these changes will impact on how the valley is perceived by visitors, and any of these changes would have complex repercussions for the tourism industry: the scenarios tested in this dissertation (see Section 7.5) were based on real situations that could face the area.

1.2 Dissertation Outline

Chapter 2 examines the existing literature in the various fields (recreation modelling, pedestrian simulation, traffic modelling, visual perception research, etc.) that this work draws on.

Chapter 3 introduces the overall simulation framework, and describes the framework’s key design concepts and major modules. Chapter 4 describes the parts of the framework that deal with agents’ strategic decision-making.

Chapter 5 describes how the framework simulates the agents’ physical movements within the simulated landscape.

Chapter 6 describes the visual modelling module, which provides agents’ with the ability to “see” their surroundings. In Chapter 7, the pilot study near Schönreid, Switzerland and its results are described. Conclusions and future work are presented in Chapter 8.
Chapter 2

Related Work

This dissertation draws upon a few major streams of previous work that has been described in the literature: recreation simulation models, pedestrian simulation methods, transportation (vehicle) simulation and visual quality models.

2.1 Recreation Modelling

While recreation simulation has existed since the 1970s, there have only been a limited numbers of distinct implementations from a few research groups. The first documented implementation was the development of the Wilderness Use Simulation Model (WUSM) [36]. This model, which was refined subsequently throughout the 1970s, was based on the particularly North American idea [68] that visitors’ enjoyment of wilderness experiences was inversely related to the number of encounters with other visitors. As a result, the model is focused on modeling and measuring how many times recreationists interact with other recreationists.

The simulation uses a very simple network representation of the recreation areas’ trails and campsites [73] and required that the modeller input summary data about groups of recreationists. This included information about their party size, start location, end location, start time, and travel speed. In order to represent the attractiveness of particular routes, modelers are required to input the probability that recreationists would chose a given path. Based on these parameters, the model generates random groups of recreationists. The model simulates a set period of time (typically a day), and inserts hikers into the network of trails at an appropriate time. The recreationists move at a constant rate along segments, and chose between paths using a weighted random choice whenever they reach a junction. The simulation records the number and location of interactions between visitors.

The model was applied to a number of locations and scenarios, including the Desolation Wilderness in California [61], Yosemite Park [73], the Colorado River in Grand Canyon National Park [71] and to the Appalachian Trail [59]. Van Wagendoonk and Cole describe its various
applications in some detail [75].

Although it predates the term, the Wilderness Use Simulation Model can be described as an early rudimentary example of an agent-based modelling, in that it simulates overall behaviours by modeling the activitives of distinct individuals within a system. Owing at least partly to the limitations of computing power in the 1970s, it is limited in the kinds of situations it can model. Its wayfinding algorithms are primitive, and only work on very limited trail networks where choices are few. As the agents do not modify their behaviour in response to the presence or activities of other agents, the model is not capable of demonstrating emergent behaviour. The model simply records the number and location encounters. Its primary use has been the evaluation of management schemes for limiting trail access, such as limiting the number of visitors entering a wilderness area over a given time period, or staggering their entries. The models were used to justify park management decisions based on these schemes.

Using advances in personal computing and the development of general-purpose modeling frameworks, Manning and associates at the University of Vermont developed successors to the Wilderness Use Simulation Model [76]. They used the Extend [39] framework to re-implement the Wilderness Use Simulation Model (which was originally developed on IBM Mainframes) and modify it for different applications. The main advances of this generation of implementations include probabilistically modelling entire routes, rather than individual segments, and incorporating different kinds of users (such as horseback riders, hikers and bicyclers) in a single simulation. It has been used in a variety of situations [45] [76] [15]. Like the WUSM, these models are primarily designed for public wilderness areas in North America where there are concerns about available capacity and regulatory frameworks for assessing carrying capacity.

The third major recreational model is the RBSim model, developed by Itami and Gimblett [40]. RBSim is a purpose-built agent-based model that is designed to simulate recreational behaviour. It differs from the earlier models in two major ways: it adds a number of features to make it more user-friendly, and it introduces the capacity for agents to change their local behaviour using rule-based decision making. While the WUSM does use a spatial representation of the landscape, RBSim makes the link between geography and the modelling framework more explicit by using GIS formats to import and save data. This ability, coupled with a graphical user interface, is intended to make the framework more accessible to a wider user base. To date, however, its use has been limited to graduate students and associates of the current authors.

The more important advance is the inclusion of rule-based decision making in RBSim. In addition to specifying individual characteristics of agents, modelers can also specify a set of behavioural rules that dictate how agents make decisions when faced with different situations. These decisions generally relate to which path an agent will take, and whether or not it will stop at a particular rest or camping site. For example, for a study that modelled multi-day river rafting trips in the Grand Canyon [29], agents, representing groups of river rafters, made decisions about where to stop for breaks or to camp. These decisions were based on the attractiveness of a given site, the number of people already visiting that particular site, and whether or not the
2.1. Recreation Modelling

Figure 2.1: Diagram of route choices from the Broken Arrow RBSim simulation. Path networks modelled using RBSim, like WUSM before it, are relatively small and have few path choices.

The RBSim model, unlike the WUSM models described above, is not deterministic, in that many of its decision-making rules are probabilistic. For many of its decisions, choices are assigned a probability of being chosen, and the simulation decides between them using a random number generator. As a result, no two runs of a given simulation are precisely the same, as an agent’s previous decisions have an impact on its current location at any given time in the simulation. As a result, Gimblett et al. run the simulation over many iterations, and analyze the results statistically to determine the impact of particular model parameters.

RBSim is designed in a modular fashion, which allows it to be modified relatively easily and used to analyze different kinds of recreational situations. One module of particular interest is
Chapter 2. Related Work

the visibility analysis module [41]. Using standard GIS raster-based line of sight analysis [7], it determines if each agent can see other agents at any given time, and if so, how many agents are in its field of view. Even though the visibility test incorporates the height of vegetation and terrain in its calculations, like all raster-based visibility analyses, the results are, like all GIS-based raster analyses [26], coarse.

Even though the above indicates interest in the recreation community for simulation modelling, there are relatively few examples of distinct simulations in the research. Cole [19] provides a comprehensive survey of the existing literature and describes less than 20 distinct research studies. One limiting factor is the relative dearth of supporting research into observed behaviour in recreational areas: in particular quantitative research in a form that could be used to populate the many assumptions required in recreation modelling. While there is some [74], it has largely been collected by the same small research community that uses simulation as a tool to explore recreational behaviour. This contrasts sharply with the two other simulation research areas that this dissertation draws on: traffic modelling, where there have been thousands of studies measuring individuals’ traffic behaviours, and, to a much lesser extent, small scale pedestrian simulation modeling.

2.2 Pedestrian Modelling for Evacuation Simulation

There is a much larger body of research exploring pedestrian simulation in non recreational contexts. This work has largely been interested in how crowds of pedestrians move in constricted environments such as large buildings, public spaces or public transportation vehicles. The research is precipitated primarily by safety concerns: understanding how pedestrians behave in emergency situations is critical to designing spaces that can be evacuated quickly.

While there are a wide range of models for simulating pedestrian evaluations [34], their focus tends to be on the implications of physical design parameters on pedestrian movements: the width of corridors or the number of available exits for example [56]. Like most areas of simulation research, existing pedestrian models use a wide range of simulation techniques to represent the pedestrians and their surroundings. These include cellular automata [17], and continuous space agent-based models (i.e [37], [38]).

These models use very sophisticated techniques for modelling how pedestrians react to each other and physical objects, and are usually carefully calibrated using observations of video capture data. The resulting results are becoming very realistic, in that pedestrians avoid physical barriers in a realistic manner, and avoid other pedestrians plausibly. It is important to point out, however, that while evacuation simulations are very realistic at the physical level, the pedestrians being simulated have very simple motivations: to escape from a given area as quickly as possible.

As one zooms out to larger landscapes, the physical realism of agents’ movements at the sub-
2.3. Vehicular Simulation

meter (or even at scales approaching 10m, depending on the scale of the landscape) become increasingly irrelevant. One is no longer concerned with how agents swerve to avoid each other or a tree: indeed, these movements are invisible when zoomed out to view the entire landscape, and have little impact on the agents’ movements over the course of an afternoon or day. One is more concerned if they are moving at an appropriate speed given the terrain and, most importantly, if they are making plausible decisions about where their path choices.

As a result, while evacuation models can provide information on how to simulate pedestrian’s physical movements, they are less useful for providing insight into how pedestrians make tactical decisions about where to go in an unconstrained environment.

2.3 Vehicular Simulation

Modern activity-based traffic modelling systems, on the other hand, deal with both the physical and tactical aspects of a simulated system simultaneously. In order to model traffic flows across an urban region, modellers are required to simulate both how individual traffic on a given street behaves (at least in terms of capacity and speed), and, at the same time, how drivers make decisions about which route to take to get them from their origin to their destination.

The Vizagent framework is directly inspired by two related traffic simulation frameworks: TRANSIMS [65], and MATSIM [50]. TRANSIMS, developed in the 1990s, was one of the first activity-based systems that used microsimulation techniques to simulate traffic across large-scale urban regions such as Dallas/Fort Worth [4] and Portland. Lawson provides one of the best overviews of TRANSIMS [44]. MATSIM (Multi-Agent Transportation Simulation) is a research-based offshoot of TRANSIMS that is currently under active development. While there are substantial differences between the two systems, particularly in their software implementation, they are conceptually similar.

One particular characteristic of these frameworks that was adopted by the Vizagent framework is how they use an iterative approach to achieving a simulated result [53]. Traffic simulations assume that individual travellers seek to minimize their time spent travelling, and chose their routes accordingly. However in typical traffic situations individual travel times are are highly dependant on the behaviour of other travellers. A route that would be the fastest in an uncongested state might not be the fastest when many other vehicles are using it at a given time. As each traveller has different origins and destinations, optimizing for every vehicle’s fastest route is difficult without an iterative approach. The typical approach in traffic modelling, including traditional 4-step models that predate agent modelling [55], is to iterate through a number of simulated model runs, with each run representing the time period of interest (typically the morning or afternoon peak travel period.) At the end of each run, the model changes a subset (typically the worst performing ones) of traveller's routes to reflect the conditions encountered on the previous model run. Depending on the sophistication of the model, this can include either changing the route or
modifying a traveller’s departure time. After a number of iterations, the simulation output should converge on a solution that represents a realistic set of route choices that fulfil the vast majority of traveller’s desires.

Originally, this iterative approach was used as a computational response to a difficult optimization problem (determining the optimal route and departure time for millions of travellers is extremely difficult.) It can also be interpreted as a simulation technique that models how individuals learn about their environment and make decisions based on their past experiences [54]. In this view, the iterative approach represents individuals testing different solutions to their daily commute: by varying their routes, departure time, and, in more complex simulation systems, even the locations of their work and homes, individuals learn which solutions best solve their particular goals. Depending on how one structures the learning algorithm, the approach can simulate individual learning processes (where the agents learn about their environment from their own experiences), or use collective learning (where agents learn from the experiences of others.) The latter seems more realistic to how individuals in society learn about the physical characteristics: they learn from each other, whether in individual conversations, or via shared media such as traffic reports, or from collective information embedded in maps, signage, etc.

The Vizagent framework’s hierarchical approach to modelling agent-decision making is derived from both TRANSIMS and MATSIM’s approach, which are conceptually similar. During each of their model runs, both TRANSIMS and MATSIM step through the following broad tasks:

- **activity generation**: each individual agent decides which activities to undertake on a given day, and at what time these activities should occur.

- **route selection**: each individual agent makes decisions about which route to take that best fulfills its chosen activities

- **microsimulation / physical simulation**: the framework simulates all agents moving through the road network simultaneously, using each’s chosen start time and travel route. The speed at which they move through the network is dependent on the choices of the other

![Conceptual Diagram of the TRANSIMS framework](image)

Figure 2.2: Conceptual Diagram of the TRANSIMS framework [54]. Each of the main tasks in the simulation are implemented as independent modules that communicate with each other. The framework invokes the modules sequentially, moving from left to right.
agents.

During the simulation run, data is collected on the travel times on each section (link) of the road network, and is used to determine the agents' behaviour for the next simulation run, with a given percentage of agents being assigned new activities and routes.

Like the Vizagent framework, MATSIM is a modular framework implemented such that software modules can be easily replaced to test different approaches within a research setting. MATSIM's design also separates the support functions (such as parsing, input and output) from the main algorithms, which allows researchers to focus on experimenting with algorithms and parameter values, rather than having to spend effort rewriting basic interfaces between modules. In order to accomplish this, a set of well-defined interfaces is required. The Vizagent framework adopts this approach.

### 2.4 Landscape Perception Research

As a key requirement of the Vizagent framework is its ability to react to landscape changes, and in particular have agents adapt to visible landscape change, it is necessary to review the literature in landscape perception. Even though there have been only a few attempts to integrate visual quality into a systematic model for landscape planning, there are numerous methods that have been used to evaluate the public's perception of the visual quality of a particular landscape.

While everyone has an intuitive idea of what makes a landscape scene visually attractive, it is not something that most people are used to quantifying. However, there is a long history of studying the visual preferences of individuals in natural settings [42] [79]. This has been an active area of research in environmental psychology for many years, using techniques such as the scenic beauty assessment method [21]. The standard technique is to have individuals assess photographs of a particular location, and use their ratings to determine the visual quality of a view or landscape [62] [31]. Many studies (e.g. [24], [67], [20]) have demonstrated that evaluations made of photographs correlate closely with people's assessments of the places they represent. While Gobster and Chenoweth [31], among others, have demonstrated that there are consistencies within the constituent elements of preferred landscapes, they have also indicated that these elements vary significantly between different types of landscapes. It is unclear, however, how generally applicable these results are across continents and cultures: can one take results from the southern United States that asked users to evaluate a landscape for its suitability for camping and apply them to hikers in the Swiss Alps?

A refinement of using photographs to evaluate landscape quality and to calibrate predictive models is to use computer generated visualizations instead of photographs [43] [48]. Using computer graphic techniques, subjects are shown a series of computer generated images in which minor elements are altered. The impacts of these changes on the subject's perceived
scenic beauty are measured. This allows the researcher to measure the impact of specific elements and/or spatial arrangements (i.e. [47]; for example if a tree is in the foreground rather than the background, it could potentially increase the visual quality of a scene.

Another related approach is to use a virtual environment to allow subjects to assess visual quality. This involves creating a three-dimensional computer model of the landscape and displaying the model to subjects in some form of immersive environment. Subjects are asked to move within the landscape, and their movements are tracked. The subjects are either asked to evaluate the landscape in terms of visual quality, or else their preferences are inferred from their path choices. Bishop et al. ([5] [12]) have demonstrated that this technique shows potential as an assessment method, although it has yet to provide comprehensive and replicable results.

A drawback of all of these studies is that the techniques used were intended primarily as comparative assessment tools. These tools allow a resource manager to compare the relative scenic values of particular landscapes, but they are not capable of extrapolating their results to new locations, or, most importantly to this study, to different management approaches to the same location. The above techniques rely on asking the subjects to directly assess the visual quality of the depicted landscape. This means that they are extremely labour intensive, and, in the case of evaluating individual photographs, do not produce results that are easily translated into a spatial model.

A number of researchers have proposed different approaches to use scenic beauty estimation as a basis for predictive models. The general approach of these models is to analyze the images or landscape being assessed and, using statistical techniques, determine how much each elements of the landscape contribute to its perceived scenic beauty. These visual quality models can be divided into two broad categories: image-based, and GIS based. Image-based visual preference models were first introduced by Shafer et al. [63]). This class of model involves directly measuring perspective images, in order to calculate statistics about the view. In Shafer's case, these statistics included the area and length of edge for different permutations of landscape type and distance from the viewer. Using regression analysis against test subjects’ stated preference, Shafer found that well over 60% of the viewer's preference could be explained by the variation of six relatively simple variables. These variables include the perimeter of foreground/middleground and background vegetation, the area of middleground vegetation, the area of any kind of water, and the are area of background non-vegetation.

While Shafer's model is intuitively quite simple to apply, as it is based on an analysis of perspective images it is conceptually and practically rather difficult to extrapolate it to an entire landscape. In order to overcome this limitation, and to enable visual quality to be integrated into standard GIS-based planning processes, a number of GIS based visual quality models have been developed [69] [46] [57] [10] [6]. While these models differ in their details, such as their computational approaches, these models all use a raster representation to classify the landscape according to its visible features. This classification is derived from existing mapping or site visits and combined with traditional GIS-based visibility analyses. Results from tradi-
tional image-based scenic beauty estimations in the area are analysed, and, using a variety of
techniques (ranging from multivariate statistics to neural networks), scenic beauty values are
generated for the entire landscape.

While useful for some kinds of landscapes and planning problems, the fact that these mod-
els rely upon raster representations of land types (usually at a coarseness of at least 30m),
means that GIS-based visual quality models are not able to capture how small variations in fea-
tures (such as a copse of trees that provides screening for a housing development) can have
a significant impact on perceived landscape quality. For agent-based models that operate at a
considerably smaller spatial resolution the results might end up being nonsensical.

Another more recent approach is the use of image rendering techniques to predict scenic qual-
ity based on the properties of specific images [11]. This technique, which is a more modern
version of Shafer’s approach, uses computer graphics technology to automatically calculate the
properties of a particular view. Computer graphic images of views are automatically generated
based on available 3D/GIS information. Additional information, such as the distance of each
pixel from the viewer and types of land cover, are calculated automatically by the computer
graphics software. Bishop et al. demonstrated that, even if one only looks at very abstract vari-
ables (such as the average depth of each pixel), one can significantly predict the scenic beauty
of a given scene.

One question that is unanswered in the quantitative visual quality literature is how scenic beauty
assessments function when applied to a route, instead of focusing solely on individual locations
(as is typically done.) Is a path that contains many moderately scenic views preferable to a a
trail that provides a single stunning view at a particular point?

A much bigger problem, however, is that while these models can predict how scenic individuals
perceive a particular view or location to be, they do not provide any information on how recre-
ationists would react to viewing one landscape over another: would a hiker change their route
to see a view that is 5% better than another view? While the literature clearly indicates that it
is possible to state that a particular view and/or location has more scenic value than another,
there is nothing in the literature that relates these comparisons to path choices in recreational
areas. It is easy to understand why: it is hard to conceive of an experimental design that would
be even remotely feasible to test all of the permutations of variables required. It is, however, a
task that is well suited to experimentation using simulation methods.
Chapter 3

The Framework - Overall Simulation Approach

3.1 Overview

This chapter presents the overall modelling framework- specific implementation details are described in later chapters. Due to the modular nature of the implementation, many of the specific details can be relatively easily modified to reflect the needs of a particular simulation; however the framework itself contains numerous modelling assumptions that are much more difficult to modify.

The Vizagent simulation framework is an agent-based approach, with each software agent representing an individual pedestrian or small groups of pedestrians. Each agent has a set of unique characteristics such as age and physical fitness. Agents are also assigned specific expectations and desires. These characteristics, expectations and desires are assigned at the beginning of the simulation, and are inputs to the overall model. They are set based on available demographic data, on-site interviews, or as part of the hypotheses being investigated by the researcher.

In order to simulate agents interacting with each other, all agents are inserted into the model at the same time. The framework uses very small timesteps, with each step representing 10 seconds of “real” time. This can be reduced for simulation environments where interactions between individuals are particularly frequent and/or important. It can also be increased in situations where the planning question does not require such a small resolution i.e. for large spatial areas where agents interact only rarely with each other. Each agent’s position and heading is updated each timestep, in response to its own goals and expectations, and to the current positions and heading of other agents in its vicinity.

A key distinction between the Vizagent framework and other agent-based recreational models is that the model operator does not explicitly set agents’ destination or paths. At the same time,
agents are not given any information about the landscape in which they will be moving: the model is designed to have the agents’ explore the landscape and learn, via the simulation, which routes, destinations and sequence of activities best meet their own goals and expectations.

3.2 An Iterative Approach

In order to facilitate agents’ learning about the simulated landscape, the simulation is run iteratively many times. Each iteration of the simulation represents a specific length of time (typically one day). At the beginning of each iteration, each agent is assigned a unique plan by the simulation framework. This plan, which describes the agent’s activities for the day, includes information on the location and time of the agent’s entry into the simulation, key waypoints and activities, and its exit location. Initially, the agents’ plans are largely random, reflecting the fact that the agents have no knowledge of the simulated landscape. During each run, data about an agent’s activities and experiences are collected and analyzed. Before each subsequent run, this data is used to determine if the agent’s latest plan met its goals and expectations: if so, the same plan is re-used for this particular agent; if not, a new plan is suggested. Over the course of many runs, the simulation converges on a set of plans that meet all of the agents’ goals and expectations.

This iterative approach allows agents’ to respond to properties of the simulated landscape (such as paths, facilities and views) as well as to the behaviour or other agents in their vicinity.

This approach is substantially different from all other recreational agent-based simulation approaches described in the literature. These systems, such as implementations like Itami and Gimblett’s RBSim [40], and in descriptions of other hypothetical recreational models [49] [8], are rule-based systems, wherein agents are given sets of rules that they use to evaluate path choices when they are confronted with them. This means that agents only make decisions about which particular path to take when they are at path junctions. While this allows the agents to react to their immediately prior experiences, it constrains the kinds of decisions the agents can make and is not representative of how hikers typically make decisions in the field. Rather than setting out randomly and waiting until they reach a junction before making a plan for the day or afternoon, hikers generally make a plan for their entire route before they begin their hike. While whether or not they change their route is likely to be dependent on the specific recreational area, in most situations they do not change their plan over the course of a day. This assumption was confirmed in the survey done in 2002 on the Schönreid study site, where only 44% of interviewed groups changed their routes from what they had planned when they had set out. Most changes, however, were extremely minor, and involved changes based on physical conditions or making poor estimations of hiking time rather than being in response to what they specifically saw in the landscape.

It is also true, however, that hikers do not typically explore a given landscape in a semi-random
fashion, such as they do in the Vizagent framework. They typically select their route based on previous experience (77% of hikers surveyed in the pilot area were repeat visitors), existing signage, or by asking for suggestions from others. As a result, visitors base their current plans on previous experiences (either their own or that of others, including those who designed and installed the signage.) As the intent of the Vizagent framework is to evaluate the impacts of a changed landscape on recreational behaviour, it is not desirable to use existing experience and recommendations as inputs to the model. If one did so, the results would be biased by the previous knowledge of what would be a status quo scenario, rather than based entirely on the experience of the modified landscape. Instead, the framework uses semi-random exploration of the landscape as a way to generate this collective experience. Under the current implementation, all agents learn from the experiences of every other agent, as the framework assumes that all knowledge is shared between the agents. While the agents' movements in initial runs is frequently (if not always) non-sensical, as they learn about their environment over the course of many runs, their movements eventually converge on a set of plausible plans that meet their expectations.

The Vizagent framework is not an optimizing framework- it does not necessarily produce results where every agent in the simulation finds the optimal route that would satisfy its requirements. This is partially due to practical concerns: given the large number of variables and spatial permutations, it would not be feasible to exhaustively explore the solution space given current computational resources. It is also true, however, that it is not likely that hikers in the real world chose their ideal route: they choose the best one they can determine, given a limited set of information.

3.3 A Modular Approach

As the Vizagent framework is intended to be used to test a variety of experiential and behavioural hypotheses within a recreational context, it was designed to be easily adaptable to different kinds of recreational experiences and hypotheses. A key design feature which allows for future flexibility is its modular nature: the framework is designed around a series of independent software modules that communicate with each other using well-defined interfaces. Each module is responsible for a specific task, and its particular implementation can be as simple or as complex as necessary given the task and the particulars of the situation being modelled.

A basic overview of the framework's core modules is given in figure 3.2.

Modules can either be implemented as distinct software classes within a large application, or as distinct software programs that communicate with each other via network protocols. Each approach has advantages or disadvantages: modules implemented as software classes are easier to manage for smaller application areas, in that they can more easily share data structures, functions and build parameters. Implementing the modules as separate applications provides more
Figure 3.1: Simple example showing how agents start out randomly, but learn better routes over time. After one iteration (top), the agents follow the shortest route on the way to the peak. After multiple iterations (bottom), they 'learn' that other routes are better suited to their individual expectations. Figure from [30]

flexibility in terms of scalability, as the different applications can be distributed across multiple compute nodes. For complex landscapes involving thousands of agents, this can significantly reduce calculation time. For smaller simulations, such as the pilot site described in chapter 7, the added complexity required to co-ordinate the different applications and communicate be-
3.3. A Modular Approach

Figure 3.2: An overview of the Vizagent framework’s modules, divided by category.

... between them is not worth the added complexity. The initial design and early prototype [30] was based entirely around a distributed approach. Unfortunately, the additional overhead required proved unwieldy in developing a fully functional system. The current implementation is complete redesign and reimplementation of this early prototype - it is implemented as a series of software classes in a single python application. The only exception to this is the visual analyzer module, which is implemented as a separate c++ program.

3.3.1 Communicating between Modules: the Event Broker

Whether or not the modules are implemented as classes or separate programs, the modules communicate with each other via events. For example, events are used to indicate agents’ positions, that they have successfully completed their plan, that they have encountered congestion, or that the whole framework is starting another simulation run. The framework uses an event-broker approach [28] [58], whereby individual modules register themselves with a central event broker module, indicating which kinds of events they are interested in. During the simulation run, modules send events to the broker module, which in turn re-broadcasts them to all other
modules which have asked for events of this type.

The event broker also provides a mechanism to allow those modules that are implemented outside of the main program (in this case, the visual analyzer) to communicate with the rest of the framework: distinct programs can register themselves with the event broker, also identifying which types of events they are interested in. Over the course of the simulation, the event broker listens to events broadcast from both internal and external sources: if it determines that a message needs to be translated from an internal (python class instance) to external message (to be sent to an external program), it translates the message into an XML message and sends it as a TCP packet to the external program.

Using an event broker approach allows the framework’s modules to be loosely coupled- as modules do not communicate directly with each other, the various modules do not need to know information about which modules are receiving their messages, or anything about their implementation. One particular advantage of this approach is that it allows one to create second tier modules that react to events generated by other modules, compute secondary information, and insert further events into the event stream, without other modules knowing that a new module has been added. While it is not possible to completely decouple modules from one another, as modules need to be aware of what kinds of events are being generated elsewhere in order to react to them, the event-broker approach significantly reduces the amount of code required for communication between modules.

3.4 Description of Key Module Types

Each module has a distinct role in the overall simulation system, but they can be classified broadly as decision-making modules, physical simulations, analyzer modules, or control modules.

Decision-making modules simulate how the agents make strategic decisions. The primary responsibility of these modules is elucidating the agents’ plans before each simulation run. The modules listen for events from other modules that describe agents’ experiences, and summarize these experiences. These modules determine how an agent can best fulfil its goals and expectations, based on their experience on previous simulation runs. These modules also receive events from the other modules, in order to refine their knowledge of the area being simulated. Their operation and function is described more fully in Chapter 4.

Physical simulations are responsible for executing the plans of all agents simultaneously. The modules are responsible for modelling how the agents react to their physical environment such as slow-downs due to congestion or path characteristics. While the mobility simulation is running, it constantly emits messages (called events) stating the status of each agent. Most of these events are simply status messages (containing the agents’ locations), but some messages contain additional information about the agents’ surrounding environment, such as steepness or
3.4. Description of Key Module Types

Figure 3.3: A comparison of the principal conceptual data flows (top) and how the data actually flows in the framework via events (bottom). While this appears to add an extra layer of complexity, in reality, it makes writing the software much more consistent as there is only a single interface for passing messages. It also means that new modules can be inserted very easily into the framework without significant software development effort.
congestion. Physical Simulations are described in detail in Chapter 5.

Analyzer Modules examine events generated by the physical simulation module, compute secondary information, and re-insert that secondary information into the same event stream. The visual analyzer is the only analyzer module currently implemented, although previous test implementations have also included local weather simulations [30]. The visual analyzer module is described more fully in Chapter 6.

Control Modules keep track of the state of the simulation and other modules, and coordinate between the rest of the modules to ensure that the various simulations are synchronized. There are two main controller modules:

- the **Overall Controller** coordinates the entire simulation. This includes setting up the simulation at the beginning of each model run (by requesting plans for each agent from the decision making modules) and informing the physical simulations when to begin and end each simulation run.

- the **Physical Simulation Controller** is responsible for examining the agents’ plans and current location, and determining which physical simulation module (depending on mode of transport and destination) should move each agent.

### 3.5 Setting up and Running the Model

In order to run a recreational scenario through the Vizagent framework, input data needs to be assembled and collected by the modeller. As the framework was implemented as a proof of concept, it does not have a graphical user interface (GUI) to facilitate setting up a model: all inputs are specified using text files (some converted from standard GIS formats using custom-written scripts). The outputs are similarly text files. The user is required to interpret the results based on these text files.

There are two main types of data required to begin a simulation: geographic data describing the simulated landscape and a synthetic population of agents who will be introduced into the simulated landscape.

#### 3.5.1 Geographic (GIS) Data Requirements

The geographic data requirements vary, depending on the kinds of questions being asked of the simulation, but at the minimum the framework requires a digital elevation model (DEM) and information about the area’s recreational infrastructure. The DEM provides information about the slope of the landscape, and is used by the physical simulations to determine agents’ speed and the relative difficulty of different paths. It is also used by the Visual Analyzer module to determine what is visible from any given point in the simulation.
In addition, the modeller is required to specify key activity locations in the simulated landscape. These activity locations designate important places that the agents might visit: parking lots, transfer points to public transit infrastructure, restaurants, and key junction points between major paths. (The latter is not strictly required, but does assist the Decision-Making modules discover key areas more quickly in a given simulation.)

The third type of geographic information required is the path network- a GIS coverage that describes all of the available walking trails in the simulated area. This coverage describes all of the paths where a hiker/pedestrian would typically walk in the landscape, and can include walking paths, country roads and sidewalks in more urbanized environments.

If the framework is using the Public Transportation module, an additional GIS coverage of public transit routes is also required. As described in section 5.2.2, schedule and capacity information are required for each public transit segment. These are specified in a separate text file that are related to the GIS coverage via unique ids for each segment.

It is worth pointing out that the activity locations and walking routes do not necessarily have to be integrated into a single network- it is anticipated that some, if not all of the activity locations do not line up precisely with the path network. When searching for a route from a given activity location to another, the framework searches for a route along a path that comes closest to the desired location.

Currently, all of the geographic data is specified using standard ESRI GIS formats (ASCII Grid for the DEM, and shapefiles for the activity points and path networks.)

### 3.5.2 Synthetic Population of Agents

The second major task for the modeller is to define a synthetic population of agents. This is the population of agents that will be introduced into the simulated landscape at the beginning of each model run. The modeller is responsible for determining how many agents belong in the population, and what their characteristics and expectations are. Characteristics are physical properties of the agents (such as their fitness level), while their expectations specify what kinds of experiences they are seeking. The agents’ characteristics are used by the physical simulations to determine how the specific agents interact with the physical world (in particular how quickly they move on sloped terrain.) The expectations are used by the decision-making modules to determine how well a particular route satisfies its expectations.

### 3.5.3 Agent Characteristics

As part of the generation of the synthetic population during the setup of the simulation, each agent is assigned a series of characteristics. Some of these, such as a speed modifier, are intended to reflect an agent’s physical characteristics such as age or fitness level. More important
Figure 3.4: The types of GIS data that are required by the simulation. To the top is the underlying terrain data; in the middle are the key locations; at bottom is the path and public transit network. All are specified in standard GIS formats (ESRI Shapefile and ASCII Grid.)
3.5. Setting up and Running the Model

for the agents’ decision making, however, are their expectations. Expectations are assigned to individual agents, and reflect their particular requirements. By assigning different expectations, the modeller can designate different types of recreational visitors, such as those desiring strenuous physical activity or those who only wish to walk on flat terrain. The expectations are implemented as a list of expectations, their target values and their respective importance (both the target value and importance are floating point values in the range of 0 to 1.) These characteristics are set based on the requirements of a particular modelling situation, either from calibration data or based on questions the modeller wishes to investigate. In the current implementation for the Schönried pilot the agent expectations include:

- **difficulty**: represents how physically difficult a hike the agent desires. 0 indicates no elevation changes, while 1 represents the most elevation change over a given hike.

- **isolation**: represents the agents expectations for meeting other agents. An isolation value of 0 means that the agent does not wish to encounter others on its trip; a value of 1 indicates that it desires frequent interactions with other hikers.

- **view**: the agent’s expectations for scenic views. While it does not make a lot of sense for an agent to actively seek less scenic routes (the expectation value should generally be 1), it is important to set an importance value, so that the system can choose between views and other expectations.

- **landscape variety**: a measure of how many different kinds of landscapes (meadow, forest, etc.) an individual wishes to hike through. Similar to the view expectation, the importance value is generally more important that the target value.

- **punctuality**: how important is it that the agent meets its targets for trip length. The target values represent the percentage of time variation that an agent expects (e.g. a value of 0.05 indicates that the agent will be content with a trip time of 57 to 63 minutes when its plan specifies an hour long hike.

The above expectations, while not exhaustive, allow the modeller to specify many different types of recreationists: those seeking an active hike far from others can be specified with high difficulty and isolation target values and importance; individuals seeking primarily a short walk with a view can be given high importance to views and a low difficulty target.

Careful consideration and choice of which expectations to include are a key task when designing a simulation that uses the Vizagent framework. The expectations define what needs to be measured by the various software modules as the agents explore the landscape. As a result, expectations need to be chosen based on what the modeller wants to investigate (e.g. in the Schönried study, the influence of views and landscape variety - both identified as key in hiker interviews (see section 7.2.1)) and refined based on what is possible to model and/or measure. While the presence of wildlife was identified as an important factor in choice of routes, modelling
the presence or absence of wildlife was outside of the scope of this project, as it would have involved developing a separate wildlife model to the overall framework. However, the design of the Vizagent framework is well suited to integrating more external models in the future, using the Visual Analyzer module as a template.

### 3.5.4 Activity Chains

In addition to specifying agents’ physical characteristics and expectations, the modeller is also responsible for defining the agents’ desired activities. This is specified as an activity chain, which lists, in order, which activities the agent wishes to accomplish over the course of a simulation run. The activities included in the chain represent both activities that take place in a single location (e.g. eating in a restaurant) and activities that take place while moving (e.g. hiking). The activity chain also specifies the agent’s starting location and its start time, which is when the agent is inserted into the physical simulation. Each activity can have a suggested duration, which the decision-making modules use as a guideline to determine possible routes and combinations of locations for each agent.

It is important to note that these activity chains are not spatialized by the modeller: while the modeller is required to specify that an agent wishes to eat lunch at a restaurant, it is up to the simulation to find a specific restaurant that meets its expectations and fits within its desired activities.

<table>
<thead>
<tr>
<th>Agent</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characteristics</td>
<td>Fitness</td>
</tr>
<tr>
<td>Expectations</td>
<td>Value</td>
</tr>
<tr>
<td>difficulty</td>
<td>1</td>
</tr>
<tr>
<td>isolation</td>
<td>0.9</td>
</tr>
<tr>
<td>view</td>
<td>1</td>
</tr>
<tr>
<td>landscape variety</td>
<td>0.5</td>
</tr>
<tr>
<td>punctuality</td>
<td>0.5</td>
</tr>
<tr>
<td>Activities</td>
<td>Start Time</td>
</tr>
<tr>
<td>arrival at parking lot</td>
<td>10:00am</td>
</tr>
<tr>
<td>hike</td>
<td></td>
</tr>
<tr>
<td>eat (on trail)</td>
<td></td>
</tr>
<tr>
<td>hike</td>
<td></td>
</tr>
<tr>
<td>departure from parking lot</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Agent</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characteristics</td>
<td>Fitness</td>
</tr>
<tr>
<td>Expectations</td>
<td>Value</td>
</tr>
<tr>
<td>difficulty</td>
<td>0.2</td>
</tr>
<tr>
<td>isolation</td>
<td>0.2</td>
</tr>
<tr>
<td>view</td>
<td>1</td>
</tr>
<tr>
<td>landscape variety</td>
<td>1</td>
</tr>
<tr>
<td>punctuality</td>
<td>0.8</td>
</tr>
<tr>
<td>Activities</td>
<td>Start Time</td>
</tr>
<tr>
<td>arrival at parking lot</td>
<td>10:30am</td>
</tr>
<tr>
<td>hike</td>
<td></td>
</tr>
<tr>
<td>eat (at restaurant)</td>
<td></td>
</tr>
<tr>
<td>departure from parking lot</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.5: Example of all of the data that needs to be specified for each agent in the simulation. Agent 1 (on the left) represents a hiker who is interested in the physical activity of a hike, in addition to the landscape’s scenic qualities, while Agent 2 (on the right) is only interested in a short walk with a nice view and a lunch in a restaurant.
Figure 3.5 gives examples of the kinds of data that are required for each of the agents, including characteristics, expectations and a basic activity chain. While at first glance this seems like a lot of information to have to provide for a population of many hundred agents, it is relatively easy to create a sophisticated synthetic population by grouping agents. By defining a few agent types, and giving them similar or identical values, one can quickly generate hundreds of agents in a synthetic population. The only aspect of the agents' description that needs to vary is the start time: creating large groups of agents with a similar or identical start time could cause unrealistic congestion events as large groups of hikers are introduced into the landscape at precisely the same time.

3.5.5 Running the Model

Once the modeller has described the simulated landscape and created a synthetic population of agents, the modeller starts the Vizagent program, which starts the simulation. At this point, the *Overall Controller* module begins the running of model iterations. For each model iterations, this controller performs the following steps:

- for each agent in the synthetic population, it requests a new plan from the decision-making modules. The location generator goes first, and allocates specific locations to the agents' activities. The plan is then passed to the router module, which selects a specific route that connects the locations in the agent plans.

- once all agents have plans, it informs (via a *start run* event) all of the modules that a new run is about to begin. After each module acknowledges that it is ready to begin, it passes all of the agent information (including their newly specified plans) to the *Physical Simulation Controller*, and informs it to start the run.

- the *Physical Simulation Controller* begins advancing through time (in increments that represent 10 seconds of real time), checking each agent's plan to see if it should be inserted into the simulation. If the simulation time has reached an agent's start time, the *Physical Controller* determines, based on the agents' starting location and next waypoint in its plan, which physical simulation module to pass it to (i.e. if an agent's start location and next waypoint are both public transit stops, it passes the agent to the public transit simulator. If both points are not public transportation stops, it passes the agent to the pedestrian simulation.) The individual physical simulations are given only the agents' current (initially their starting) location and their next waypoint, not the entire plan.

- the *Physical Simulation Controller* module continues advancing through simulated time, sending *advanceTo* messages to the physical simulations, indicating that they should move any agents that they are currently responsible for. The physical simulations determine how far, if at all, its agents have moved in the recently elapsed time. As they move...
the agents, the physical simulations broadcast the locations of their agents using **Agent Position** events. The physical simulations also broadcast experiential information (such as when agents encounter other agents, or information about the difficulty of a particular route) using **experience** events. As the agents reach their next waypoint, the physical simulations inform the **agentController** (and the mental modules) that the agent has reached its destination. The **agentController** examines the agent’s plan, and sends the agent (with its current location and next waypoint) to the appropriate physical simulation.

- During this time, the analyzer modules (such as the visual analyzer) receive movement events that indicate the agents’ location, and use this information to generate additional interpretations about the agents’ experiences (e.g. indicating what they can see). This information is reinserted into the framework as additional **experience** events. The mental modules use this information to make assessments of particular agent routes.

- As agents complete their plans, the **Physical Simulation Controller** removes them from the physical simulations, and continues advancing through simulated time. Once either all of the agents have completed their plans, or if the agents’ plan is impossible to complete within a preset limit (generally a single day), the **Physical Simulation Controller** indicates to the **Overall Controller** module that the simulation iteration is complete.

- The **Overall Controller** begins the process again by asking the mental modules for a new plan that best satisfies the agents’ expectations and activity chains. As the mental modules have been listening to agents movements and **experience** events, their understanding of the landscape improves with each subsequent simulation iteration, causing them to suggest different routes.

Over many simulation iterations, the agents plans and movements should converge on a result that reflects the agents’ expectations and activity chains. Determining when a simulation has reached “convergence” is generally straightforward with sparsely populated recreational environments where congestion is not a major concern, as agents’ best routes are quickly discovered via exploration. Simple measures, such as the percentage of agents whose plans and routes change between simulation runs, can indicate if a simulation has reached convergence.

### 3.5.6 Viewing the Model Runs

The Vizagent program, as it lacks a GUI, outputs the agent positions from each simulation run to text files. These contain information about all of the agent locations over the course of a simulation. In order to examine the agents positions over time, a python-based viewer was developed. It displays GIS-based information (such as paths, transport routes, orthophotos), and loads in the agents’ positions from the simulation’s text files. It offers a simple GUI, which allows the user to step through the a simulation and see the agents’ movements during a simulation iteration.
In order to facilitate debugging and understanding the behaviour of the model, the viewer also allows the user to examine agents’ plans.

### 3.5.7 Model Results / Output

The final results of an entire simulation is a text file that describes all of the agents’ movements on their final iterations of the simulation (their movements on previous runs are also recorded) and a text file which indicates how this run meets each agents’ expectations. Even though the framework may have converged on a solution that best meets its these expectations, it is entirely possible that this solution is not adequate for fully meeting their expectations. This indicates that the given scenario does not provide the kinds of recreational opportunities that the agents’ require, or that the modeller has, during the setup and calibration phase, set the agent expectations too high.

![Graphical user interface of the python-based viewer program.](image)

Figure 3.6: Graphical user interface of the python-based viewer program. It loads in the results of a simulation run and shows the agents movements over the course of the simulation. It can also display underlying GIS information, such as path networks and points of locations. The user can step through the simulation and examine each agent to determine its current plan and expectations.
Chapter 3. The Framework - Overall Simulation Approach
Chapter 4

Modelling Agents Strategic Decision Making

4.1 Introduction

As described in Chapter 3, the Vizagent framework divides the agent decision making into two: tactical/physical decisions are made by the physical simulations while strategic decisions are the responsibility of the decision-making modules. By this distinction, which was adopted from MATSIM and Transims, tactical decisions are immediate decisions by the agents in response to their immediate physical environment, such as how fast to go and how to avoid obstacles in their near vicinity. Strategic decisions are higher-level decisions about which route to choose, requiring some planning “forethought”. This chapter describes the role of the decision-making modules, how they make strategic decisions for the agents, and details of their implementation.

The mental modules’ main role is to elaborate “plans” for each agent before each model run. A plan lists, in order, the specific locations that an agent will visit over the course of the model run. The plan also contains information about when an agent will start the simulation, and how long, if at all, an agent will pause at each of its designated locations. Each agents’ plan represents the simulation’s estimation of the route that would best satisfy the agent’s particular requirements, given the overall simulation system’s current knowledge of the landscape.

While elaborating plans is the mental modules’ key responsibility, an important secondary task is their role in observing and interpreting the agents’ environment. As the simulation runs, the physical simulations broadcast events that indicate the agents’ experiences at a particular location and time of day. The mental modules listen to these events, interpret them and store the resulting interpretation for use to generate the next day’s plans.

The initial plans generated by the decision-making module for the first model iteration are quite coarse and unrealistic. The mental modules’ knowledge is limited to expected travel times based on the distances between points in cartographic space. The system has no knowledge of the
At the end of this elaboration process, the Agent Database contains a plan for each agent. Each plan represents the overall system's current best solution to the agent's goals and expectations. Over the course of many simulation runs, this solution will generally improve as the agents have the opportunity to explore the simulated landscape and discover more appropriate solutions.

A simplified representation of an agent's plan is contained in figure 2. Once the Agent Database has received elaborated plans for each agent, they are submitted to the Physical Simulation for execution. At this point, the Agent Database assumes more of a "controller" role, primarily ensuring that the various modules are able to keep up with each other. It does this by throttling the entire simulation (by requesting that the physical simulation wait after each time step) if some of the modules are not able to process events and/or requests quickly enough.

```
<plan agent="1" plan_id="1">
  <activity id="1-1" type="enter_simulation" time="324000">
    <location id="1-1-1" type="parking_lot" x="512432.2" y="508343.5" />
  </activity>
  <activity id="1-2" type="hike" suggested_duration="3600">
    <waypoint id="1-2-1" type="node" node_id="1246" x="512438.5" y="507834.3" />
    <waypoint id="1-2-2" type="node" node_id="1247" x="512436.0" y="507820.9" />
    (...)
    <location id="1-2-1" type="hike_waypoint" x="512450.0" y="508012.3" />
    <waypoint id="1-2-12" type="node" node_id="1281" x="512470.5" y="507950.3" />
    <waypoint id="1-2-13" type="node" node_id="1284" x="512322.5" y="507912.8" />
    (...)
  </activity>
  <activity id="1-3" type="eat" duration="1800">
    <location id="1-3-1" type="restaurant" x="514432.0" y="505323.0" />
  </activity>
</plan>
```

Figure 2: Simplified XML Plan. The simulation system dynamically generates a new plan for each agent every day.

underlying topography, and no knowledge of the particular characteristic of each route: aside from the rough estimate of distance, each route is considered equally as attractive from the perspective of the agents’ expectations. As the agents explore the landscape through subsequent iterations, the plans become more and more realistic.

Even though the agents learn about their simulated landscape by exploration, their exploration is not random. Given the large numbers of path choices in even the simplest simulated area, random exploration would take far too many iterations to achieve plausible results. The hierarchical approach used by the decision-making modules, where each agents’ plan is elaborated hierarchically from large decision (e.g. which public transit route to choose) to small (e.g. which specific trail to select), means that the agents’ exploration is targeted towards areas that will most likely meet their individual requirements. In order to avoid situations where the agents quickly discover an acceptable route and no longer explore to find a better solution, a proportion of agents make some of their decisions randomly each model iteration. The number of agents making random exploratory decisions declines over subsequent iterations, according to a user-specified function.

There are two decision-making modules in the current implementation. They are responsible for generating different portions of the agent plans, with each representing different spatial and temporal decisions. The two currently implemented mental modules are:
4.2. Implementation of the Decision-Making Modules

- The **location generator** assigns specific locations to the agents’ list of activities. This includes assigning key points in the middle of an agent’s hike.

- The **router** generates specific routes between the locations specified by the location generator.

While not currently implemented, a third, more abstract mental module is also considered. This module, the activity generator, would be responsible for trying different chains of activities. Implementing an activity generator would make it possible for the simulation to test if trying different activities in different order (such as reducing the amount of hiking and increasing the amount of time spent having lunch) would increase their satisfaction. At present, however, the agents are given fixed activity schedules as inputs to the simulation. (see section 3.5.4 for more detail.)

Figure 4.2 displays the relationships of the mental modules and their contributions to the agents' plans.

### 4.2 Implementation of the Decision-Making Modules

The location generator and router modules share a lot of similarities (they are implemented as closely related software classes). Each:

- maintains an internal representation of all possible agent choices at their respective spatial scales (for the router this representations is akin to a geographic map of intersections and paths, while the location generator’s network represents key locations and all connections between them.)

![Figure 4.2: Schematic representation of each Decision-Making Module’s contribution to the plan generation process. As the Overall Controller queries the modules from top to bottom, the agent’s plan gains increasing resolution. In the current implementation, the Activity Generator’s output (a non-spatially specific activity chain) is specified manually by the modeller.](image-url)
• listens to the agent experience events generated by the physical simulation and summarizes this information using an events analyzer algorithm. The summaries are stored based on the spatial and temporal scale of the module (e.g. per location pair by the location generator and per node-pair for the router).

• Contains a route selector function which chooses a route along their respective network that satisfies an agent’s requirements.

The two mental modules are implemented as different subclasses of a common parent. As a result, all mental modules share the same basic structure: each maintains a network of coordinates and links between these coordinates. These coordinates and links represent different kinds of locations and connections between them, depending on the needs of the particular mental module. For instance, the router’s network represents the path network, with each point representing either the terminus of a path or a junction where two or more paths meet. The links represent the paths that link these points. The location generator’s network has far fewer points and links: its points represent key locations such as restaurants, public transportation stops, major path junctions or locations that the simulation has determined are particularly attractive. Unlike the router’s links, which match closely with physical connections between its points, the location generator’s links are conceptual, and link every location with every other location, regardless of accessibility.

Linking every activity with each other in the location generator’s network does create some potential scaleability issues. The resulting network is very dense, which creates a very large number of possible activity chains that need to be tested by the simulation. While this would obviously pose a problem for simulated areas where there is an arbitrarily large number of activities, in real world situations, this is unlikely to be the case. The pilot site has only 16 activity locations (see figure 3.4), which is quite large when compared to other areas that have been simulated in the literature. The route selector algorithm (described in section 4.2.2) limits the possible number of choices, and therefore the length of the activity chains that need testing, by limiting the geographic distance an agent travels over a plan and by not allowing backtracking. For more complex recreational areas with more activity centres, a more detailed route selector algorithm or more nuanced network of activities would have to be developed.

Each link in the modules’ network has data associated with it: this data is a summary of all of the experiences of all agents who have traversed the particular link. This summary, generated by the events analyzer algorithm (see section 4.2.1) implemented in each of the decision-making modules, includes information on travel time (how long it takes the agents to traverse the link) and information on each of the agent expectation categories (as defined in section 3.5.3.). While some of the experiences (such as trail difficulty) will be the same for every agent using a particular link, many of the experiences, such as congestion and travel time will vary according to the attributes of the agent and the actions of others.
4.2. Implementation of the Decision-Making Modules

One significant complication is the fact that some of the agents’ experiences are time-dependent: whether a particular path is congested depends on both the time of day and the activities of other agents. A path that is congested at 8:30 might be empty at 9:00. Travel time is also dependent on the operating schedule of any public transportation modelled by the system, such as buses or trains. For example, assuming that a particular gondola operates from 9am to 4pm, a trip beginning at 9am might take only 10 minutes, while for an agent that arrives at 8:40, it might take 30 minutes, including 20 minutes of waiting for the first ride. In order to accommodate time-dependent factors, each link stores the travel times and summaries of expectations in time-dependent “bins”. Each bin represents a specific time of day, such as from 10:00am to 10:15am (the amount of time stored in each “bin” is configurable by parameter at the beginning of a simulation.)

While congestion and public transportation schedules are the only time-dependent factors currently implemented and used in the pilot study, this structure is flexible enough to accommodate numerous other submodels that could be implemented as being time dependent. These could include factors such as weather, sun location, opening hours of restaurants, etc. Adding the bins to each link does significantly increase the amount of memory required for each link the simulation, however for the kinds of sparse networks typically used in recreational simulation, the overhead is minimal.

4.2.1 Events Analyzer Algorithm

Both mental modules use the same algorithm for collecting and summarizing experiences. As the physical simulations move the agents through the landscape, the physical simulations and the analyzer modules generate experience events that provide information about the agents’ interaction with the physical environment. There are different types of experience events, corresponding to the types of agent expectations (defined in section 3.5.3).

Each event contains the following information:

- the unique identifier of each agent;
- the time when the event was generated;
- the type of experience;
- and, depending on the type of experience, any additional parameters that describe the event (i.e. the degree of steepness.)

One issue is that the physical simulations produce a lot of experience events: as an agent moves along a trail, the pedestrian simulation sends a series of events that indicate, for each every time step, how difficult a particular route is, the type of landscape the agent is moving through, the quality of its view, etc. For the pilot Schöried study, each time step represents
Figure 4.3: Comparison of the different types of networks maintained by the framework. In red (top) is the network of key locations, which is maintained by the location generator. In blue (middle) are the routes, managed by the router. Each link in the router’s network represents a path from one junction to another. The black routes (bottom) are the physical locations of the routes in geographic space. In order to more easily provide realistic movement choices, the router inserts waypoints from the actual physical routes between the nodes specified by its routing network.
10 seconds, which means that the pedestrian simulation generates thousands of experience events. It is the role of the events analyzer to listen to these events and summarize them for each link in the mental module’s network.

In order to correctly store experiences with the right link, a key task for the Events Analyzer is determining which segment a particular agent is on. As the physical simulations move the agents from place to place, they broadcast events indicating when an agent has left a waypoint in its plan (leftWaypoint message) and when the agent has reached the next waypoint (using an arriveWaypoint message) described in its plan. As an agent leaves a waypoint, the mental module notes the time and begins collecting the agent’s experience events and stores them in a collection associated with that particular link. When the agent indicates that it has reached another waypoint, the mental module compares that waypoint with the nodes in its particular network. If the waypoint corresponds to a node in the mental module’s network, the Events Analyzer summarizes the collected experiences and stores the summary for the link that connects the two nodes. If the waypoint does not correspond to a node in the mental module’s network, it continues to collect events and examine waypoints.

The experiences are summarized differently depending on the kind of how they present their information in the experience events. Those which report a scaled value for that type of experience (i.e. difficulty, congestion), are simply averaged over the total travel time. Others that are made up of composite information average their component values and store their constituent values with each node (i.e. in the case of landscape variety, the Analyzer calculates the percentage of views that belong to each landscape type. )

4.2.2 Route Selector Algorithm

The route selector algorithm is responsible for choosing a route for the agent that best meets its expectations. It is used by both the Location Generator and the Router modules: the only difference between them is the network of links and nodes they use. (See figure 4.3.)

The route selector is one area where the Vizagent framework departs significantly from the MATSIM and Transsims (see section 2.3) approach. In traffic modelling, it is assumed that individuals seek the shortest path, as determined by travel time, that meets their other requirements. As a result, current Transsims and MATSIM implementations use variations on classic shortest path algorithms such as Dijkstra’s algorithm [22].

Unlike commuters however, hikers typically do need seek the shortest possible route that meet their other requirements: for them, the route is as much of the objective as the destination. In fact, for many hikers, the route is the destination, as many plausible routes are a circle route where both the origin and destination are the same. While the route is important, a hiker’s satisfaction with a day’s hike does not increase uniformly with length: typically hikers have a given time budget within which they seek to maximize their other objectives. While their route
might be shorter or longer than they had originally intended, significant variations from their intentions are not desired unless the route is particularly attractive.

Originally, the intention was to use a modified shortest path algorithm based on Dijkstra’s. Instead of having short travel times represent maximum utility, the idea was to use a combination of an agent’s goals and objectives (such as difficulty, views, etc.) and its desired travel time to indicate utility. As shortest path algorithms can be viewed as an inverted spatialized utility maximization function, it was thought that this approach would create feasible results. Unfortunately, this was not the case. The key difficulty is that the utility of a given segment is not known until it is seen as a part of an entire route. If an agent desires a route of a particular length between two points, it is impossible to know how important this given segment is without first calculating all possible permutations of routes between the two points. The concept of maximizing variability of experience generates a similar problem: without knowing the entire route in advance, it is impossible to calculate its contribution to the route’s utility. This contrasts with shortest path problems, where the utility of a given segment is only a function of the segment’s properties.

Another difference between recreational route generation and more typical travel problems is that doubling back on a particular path is considered acceptable, and in many cases, is even desirable. This is often true when hikers start and end their day at the same location (i.e. at the parking lot), or when their destination is reachable by only a single access route (i.e. a special viewpoint.) Shortest path algorithms assume, as makes sense given their goal, that repeating the same segment is not permitted.

The solution was to implement a custom function (see figure 4.4) that traverses the entire network, starting from the agent’s starting point and builds a list of all possible routes that might reach its destination. As the agents are also given a desired time to complete their route, the algorithm estimates (based on distance and the time required for other agents on the same route in previous simulation runs) how long the particular route will take the agent to complete. If the estimated time exceeds the agent’s desired time by a user-specified factor (2 times for the pilot study) before reaching the destination, the algorithm discards the route. This prevents the algorithm from having to calculate all possible permutations which, even given modern hardware, would not be feasible given hundreds of agents and a reasonably large path network.

In order to allow backtracking, the module assumes that different directions along a given segment are distinct links in the network (i.e. walking in one direction along a path is a different experience from walking in the opposite direction.) The route selector function does not allow the agent to use the same link twice: this means that each agent can use the same segment twice, once in each direction, but no more. This avoids situations where the algorithm would pad a trip by traversing a particular segment many times, and reflects observed practice where individuals will use the same route in different directions once, but not more often. In addition, agent expectations can be flagged so that they only contribute to an agent’s experience in one direction: landscape variety and view quality, for instance, are only counted once if the agent goes back along the same route, while travel time, difficulty and isolation experiences are
add starting_point to list_of_possible_routes
for each current_route in list_of_possible_routes do
  for each link leaving end_point of route do
    if link already exists in current_route then skip to next link
    copy current_route to new_route
    add link to new_route
    if new_route.end_point = destination then move new_route to viable_routes
    else add new_route to list_of_possible_routes
  end if
end for
remove current_route from list_of_possible_routes
end for

Figure 4.4: Pseudo-code for the traversal part of the Route Selector Algorithm

recorded in both directions.

As the algorithm discovers a route that connects the origin to destination, it calculates the route's expected value for each of the expectation categories, and calculates an estimated travel time.

In order to easily compare the performance of each route, the routes performance with respect to each expectation are normalized to a number between 0 to 1. While a key behavioural question is how experiences over the course of a route affect its overall performance (e.g. is a particularly steep but short section less suitable than a long moderately steep section), the current implementation assumes that an overall experience of a route is the average of all experiences. As some experiences (such as isolation) are not generated each time step (due to the agent not encountering other agents), the calculation assumes that time steps where no experience was received have a value of 0. For landscape variety, the percentage of landscape type being viewed is summarized for each landscape category (see section 6.2.1 for a definition and list of the landscape categories). A value from 0 to 1 is calculated for landscape variety using Simpson’s diversity index [64]:

\[
d = 1 - \left( \frac{\sum n(n - 1)}{N(N - 1)} \right)
\]  

(4.1)

where \(d\) is the resulting diversity index, \(n\) is the value for a particular category, and \(N\) is the total of all values (in this case, it is the number of landscape categories.)

Once the algorithm has calculated the performance of the route against the expectation categories, it scales them based on the individual agent’s weighting of expectations (as specified by
the modeller at the beginning of the simulation. It does this using another simple algorithm:

\[ s_a = \sum |E_x - E_{x_{des}}| * E_{x_{aw}} \]  

(4.2)

where \( s_a \) is the route’s score for this agent, \( E_x \) is the route’s score for this expectation, \( E_{x_{des}} \) is the agent’s desired value for this Expectation, and where \( E_{x_{aw}} \) is the agent’s specified weighting for this Expectation.

### 4.3 Plan Evaluator

While the mental modules listen to an agents’ experiences in order to evaluate specific routes within the simulated landscape, each agents’ movements over the course of a simulation run also need to be evaluated, in order to determine if the overall plan meets its expectations. Fortunately, the Event Analyzer can also be used to calculate this overall score without modification. An instance of the Event Analyzer is started within the Plan Evaluator module, which listens to all of the agents’ experiences. Once an agent has completed its plan, and is removed from the simulation, the Plan Evaluator summarizes the experiences and assesses them. The resulting score from 0 to 1 indicates how well the agents’ movements in that simulation run reflect its expressed expectations.

For the next simulation run, the framework uses this score to determine if another plan generated by the Route Selector algorithm would better suit the agent’s expectations. If so, the Controller provides the agent with the new plan. If the existing plan provides better performance, the Controller maintains it.
Chapter 5

Physical Simulations

As described in section 3.4, the physical simulation modules simulate the individual behaviour of the pedestrians. The physical simulations are responsible for modelling basic interactions between the agents and their physical environments. These interactions include things such as modifying speed as a result of terrain, avoiding physical obstacles such as trees or buildings, and avoiding collisions with other agents.

The framework is designed to allow for multiple physical simulations to run simultaneously within the simulation run. This is intended primarily to allow for different modes of transportation to be simulated simultaneously: agents are able to transition from one mode of transport to another during the same model run. Discrete simulations model how the agents move when they are hiking, cycling, taking public transit, or even driving private vehicles. Individual agents are assigned to the different physical simulation modules by the Physical Simulation Controller module (see section 5.1.)

The modular approach allows the physical simulations to be relatively simple- their main responsibility is to determine how the agents move between two given points. The simulations determine the agents’ exact route, their speed, and how they avoid collisions with objects and other agents in their vicinity. Depending on the type of simulation, they also broadcast events about the agents’ experiences as they are moving through the landscape.

5.1 Physical Simulation Controller

The physical simulation controller is responsible for dispatching agents to the different physical simulation modules. Its functionality is described coarsely in section 3.5.5.
5.2 Physical Simulation Modules

Like all modules within the framework, the modules communicate with others via events. All physical simulations are required to respond to the following kinds of messages:

- An **Accept agent** event is sent by the *Physical Simulation Controller* to instruct a physical simulation to accept an agent. It indicates the agent’s initial position, and contains references to an agent’s physical characteristics (mainly speed);

- A **Move To** event is sent by the *Physical Simulation Controller* to instruct the physical simulation to move the Agent as quickly as possible to the geographic coordinates included in the message.

- A **Remove agent** event is sent by the *Physical Simulation Controller* to instruct the physical simulation to remove an agent from its control. This message is typically sent when an agent has completed its plan, the run is finished, or the agent is being transferred to another physical simulation.

- An **Advance To Time** event is sent by the AgentController to instruct the physical simulation to run its simulation until the given time. This keeps all of the physical simulations synchronized.

In addition to receiving these messages, each physical simulation is required to send **Agent Position** events for every simulation time step where the agent moves. These messages inform other interested modules of the agent’s current position.

In the current implementation, two different physical simulations are implemented as separate python modules: a public transport simulator, and a continuous-space hiking simulator. The two simulations allow one to model how agents move using the three main transportation modes within the pilot site: hiking, gondola/chairlifts, and trains/buses (the latter two modes are modelled by the single public transportation simulation.)

5.2.1 Pedestrian Simulation Module

The pedestrian simulation models how individuals walk through the simulated landscape. As the agents’ final destinations and waypoints are determined by the decision-making modules, its primary responsibility is to model how the agents move from one waypoint to another. The module determines the agents’ exact position, how fast the agent is moving, and reports the physical characteristics of its immediate surroundings to the rest of the framework.
5.2. Physical Simulation Modules

Relationship to Gloor’s Model

The current implementation is based on the work of Gloor [30] and Mauron [51]. Their work examined different simulation techniques for modelling the individual movements of pedestrians. Their work is microscopic, and is concerned primarily with modelling how pedestrians interact in close proximity to other pedestrians and/or physical objects. They use a continuous space representation of the agent’s physical surroundings, meaning that the agents are able to move freely and continuously in any direction, and are not constrained to specific grid cells or path networks.

This is in contrast to other models that use a cellular automaton or a network representation to structure their agents’ movements. In cellular automata-based pedestrian models ([60], [13], [23]), the simulated area is divided into discrete grids, each of which can contain a single pedestrian. Agents are restricted to moving from one cell to another, causing, depending on the grid resolution, jagged motion. Most recreation simulations restrict agents to moving on a given path network ([40], [73])- the simulation only determines the agents’ speed along a given trajectory. Both approaches are typically used to avoid the computational complexity of continuous space models - while the simulated movement is less realistic, they are much easier to implement and require considerably less computing resources.

Gloor’s approach is based loosely on Helbing et al.’s social force model [37]. The model calculates each agent’s velocity and direction based on three factors: the difference between an agent’s current velocity and its desired velocity, its interaction with other agents, and with physical objects in its immediate vicinity. Gloor’s model, like Helbing’s, is primarily concerned with interactions at a sub-meter scale: it accurately models how pedestrians avoid each other as they approach another pedestrian or physical obstruction. Mauron’s calibration experiments, which analyzed videos of pedestrians passing each other on a sidewalk, indicated that the influence of nearby pedestrian and physical obstacles on pedestrian’s local movement choices decays quickly after 1m.

While the model developed by Gloor provides a very realistic representation of agent interactions, particularly at the sub-meter scale, this amount of detail is far in excess of that required to answer the kinds of questions posed by a landscape-level recreational simulation. Even though it is important that the simulation framework be able to measure the number of times an agents comes within close physical proximity of other agents (as the measurement of these fields is critical to the whole field of wilderness recreation modelling), it is less important that the agents avoid each other plausibly at the sub-meter scale.

Another limitation of Gloor’s implementation is that in order for it to provide meaningful interactions between agents and their physical environment, it requires that every object in that space be described in great detail. For trees, this would require each stem to be specified precisely, not to mention objects on the forest floor. Unfortunately, for natural environments, this kind of data is not available. Rather than detailed surveys of individual trees, or structures, existing
mapping only provides rough outlines of forested areas and rough outlines of built area. While it is possible to extrapolate from this information, the resulting model would then simulate movement based on this extrapolated data, creating detailed movements that are not based on real world conditions.

Despite the fact that Gloor’s model provides and requires far too much detail for the specific application in question, the original intent was to use Gloor’s implementation as the physical simulation module. However, his implementation had a few assumptions that made it difficult to use without a major re-write of the software. One omission was that in Gloor’s model, agents maintain a constant velocity, regardless of the underlying slope. As slope is known to have a significant impact on hiking speed [74] and is a key factor in hikers’ choice of routes [18], this is a major omission.

Although Gloor’s model is a continuous space simulation at the local scale, its reliance on knowledge of an underlying path network to provide forces to get the agent moving realistically along the path (using what Gloor calls shadow tags) means that in practice, the agents always closely follow the paths. A true continuous simulation would allow agents to deviate from the path if necessary or desirable (see figure 5.2.) While agents do deviate from the centre line of the path in Gloor’s model in response to the presence of other agents or to make more realistic turns, these deviations are practically never more than a meter from the centre line of the path. From the

Figure 5.1: A true continuous space simulation would allow an agent with a starting position of 'A' to cut the corner to reach its destination at 'B', if the intermediate area was passable. In Gloor’s implementation this is not possible, as it defines waypoints at every curve of the path.
perspective of the landscape-level patterns that one is interested in when simulating an entire recreational area, these deviations are neither important nor visible. The additional overhead of this implementation is considerable, for negligible benefits to a landscape level simulation.

While modifying Gloor’s source code was considered, the implementation has many features that made the code particularly difficult to adapt. These features, such as lazy initialization, were added to allow the highly realistic motion described above to scale to a huge landscape with thousands of agents. While technologically impressive, it was determined that these features were not needed for the purposes of the Vizagent framework. While the physical simulation as implemented runs much slower than had it used Gloor’s implementation, performance is acceptable for the pilot area, and is likely to be so for typical recreational areas.

Current Implementation

In the end, it was determined that it would be easier to create a clean implementation of the physical simulation, rather than adapting Gloor’s software. The current implementation is, however, based on Gloor’s core algorithm. Like most of the Vizagent framework, it is written in Python.

The Pedestrian Simulation performs a single task: it moves agents from one point to another, in as straight a line as possible. The simulation uses two GIS-based raster coverages as inputs: an elevation map, and a walkability raster. The elevation grid, or raster, is used to calculate the slope and direction of each agent’s path (by sampling the elevation at the agent’s current and previous location, and doing a geometric substraction to get the vector of the slope.) The walkability raster, which represents how easy it is for any agent to walk over a given area of the landscape. Walkability is represented by a floating-point value ranging from 0 to 1, with 0 representing impassable (walls, etc.) and 1 very easy to pass (graded paths, asphalt, etc.) Both rasters are pre-computed using standard GIS software from geographic data, and are stored using the standard ESRI Grid format [77].

Like all of the physical simulation modules, the pedestrian simulation maintains a list of agents that it is currently responsible for. As soon as the pedestrian simulation module receives an advanceTo message from the AgentController, it iterates through all of the agents it is currently responsible for and updates the agents’ position, orientation and velocity. This is done using a simplified version of Gloor’s algorithm, with some modifications outlined below.

First, the module determines each agent’s maximum speed by determining the slope underneath the agent’s current position. This is done by sampling the terrain raster at the agent’s current position, and at a small distance from the agent in the direction of the agent’s current destination). Using vanWagendonk’s formula [74], this slope is used to calculate the agent’s speed. This speed is then multiplied by the agent’s speed modifier, which is used to represent differences in an agent’s physical capabilities. Finally, the agent’s desired velocity is multiplied
Chapter 5. Physical Simulations

Figure 5.2: The two input rasters for the physical simulation. The raster on the left represents the terrain’s elevation, and is used to calculate the slope of an agents’ route. The raster on the left represents walkability, and is scaled from 0.0 (not passable) to 1.0 (easy to walk over.)

by the agent’s location’s walkability score, which represents how easy it is to traverse the given area (faster for trails, slower for grass, etc.)

The slope calculation is also used to determine how difficult an agent’s current route is. Slopes below 3% are considered to have a difficulty rating of 0.0, while slopes above 25% are considered to have a difficulty of 1.0 (slopes above 25% are generally impassable without climbing, except for short sections [27].) Intermediate slopes (between 3% and 25%) are scaled linearly between these values. If the slope is above the minimum slope threshold, the module generates difficulty experience events that indicate the degree of difficulty to the rest of the simulation.

Next, the module determines if there are any agents within its range of influence by iterating through all of the other agents in the simulation and measuring their distance from one another. While this is an expensive operation computationally, for the limited number of agents and modern hardware, it is trivial. At the same time, the module checks to see if any agents are within the agent’s “perceptual” threshold- the distance within which the agents’ register the other agents as being close enough to be an encounter. This distance is considerably larger than the distance that influences their physical behaviour (the latter distance was set using Mauron’s [51] data, which indicates that one pedestrian’s influence on another’s physical movements ends at distances larger than 2.5m. If the module determines that there are agents within the agent’s perceptual threshold, the module broadcasts an experience event indicating that the agent has had an encounter with another agent.

If there are no agents within the physical influence threshold, as is almost always the case in recreational simulations, the calculations are very simple: the module adjusts, if necessary, the agent’s orientation to line up with its current destination, and sets its velocity to be its desired velocity, as determined previously.
If there are other agents within the agent’s physical influence threshold, the module uses a modified version of Gloor’s algorithm. The algorithm calculates an agent’s desired direction and velocity based on forces that represent other agents within its vicinity and on forces that represent the pressures from its surrounding environment. These forces are very strong at very short distances, but decay quickly as the agent moves away from other agents or obstacles. This causes agents to avoid collisions with each other and with obstacles. At the local scale, it provides very realistic behaviour, as approaching agents swerve to avoid other while avoiding physical objects in their vicinity. The main differences between Gloor’s implementation and the one used in the pedestrian module in this work include:

- Rather than using Gloor’s implementation of *shadow tags*, instead we use the naive approach for simplicity (see section 3.2.1 of [30]). This causes the agents to aim towards the precise centre of the path at waypoints. While this loses some realism at waypoints, in practical terms it is hardly noticeable.

- Rather than pre-computing the environmental forces based on a set of vector-based objects, we calculate them on-the-fly using the walkability raster as a proxy for physical objects. This works well as the walkability raster implicitly contains information on how solid or impassable any given area is. As the forces in Gloor’s model decay quite quickly over a limited distance, we only need to sample a short radius (currently 2x the physical influence area described above) to achieve plausible results.

The resulting agent movement is, as expected, not as realistic visually as Gloor’s implementation. This is particularly true when one zooms in to examine agent behaviour at waypoints and destinations where many agents are waiting at the end of a run. However, this situation does not occur very often in a typical recreational simulation. And as these unrealistic artifacts are not visible when viewed at a typical resolution, this lack of realism has no impact on the results for a landscape-level simulation.

Indeed, in retrospect, the additional overhead of the environmental force calculation, even though it is only used when agents approach each other, is not even necessary for a landscape-level simulation of the kind exemplified by the pilot study. As there is no conditions where hiker congestion is so strong that hikers are forced to slow down in response to each other, a very simple model, which modulates speed in response to the underlying slope and that recognizes when agents are in close proximity to each other would have sufficed. However, for other recreational areas where congestion would be more of a concern, the added complexity would be useful, as it allows for realistic slowing down of agent speed in response to their immediate surroundings.
Figure 5.3: Example of the forces that are used to calculate the environmental forces on agents when they are in close proximity to each other. The direction of the forces is always perpendicular to the agents’ desired direction. Their magnitude is derived from the walkability raster. This causes the agents to avoid each other while also staying, as much as possible, on the trail. Figure from [30]

**Improvements to Future Pedestrian Simulations**

A key advantage of the modular nature of the framework and the fact that the various modules have been given relatively simple tasks, is that should be relatively simple to improve the capabilities of the physical simulation module. For simulations where very realistic movements at the sub-meter scale are important, another module could be developed that provides a higher degree of realism.

In the current implementation (mirroring Gloor’s implementation), the router inserts sub-waypoints into the agents’ plans. These define the curves in the path, and means that the physical simulation is always provided with start and end positions that are in a straight line along a path. This means that the Vizagent simulation shares with Gloor’s implementation the quirk that, while the pedestrian simulation algorithm is a continuous space algorithm, the overall framework resem-
bles a more typical path-based simulation. With a more refined router implementation, moving the entire framework to a continuous space model would be possible. For performance reasons, one could use two different pedestrian simulations, one for on path travel, and one for agents that leave the path. As our studies in the test site indicate that very few, if any, people ever left the standard paths and that the pathless simulation would require considerably more resources, splitting the two situations into two simulations would make a lot of sense.

5.2.2 Public Transit Simulation

The public transit simulator simulates agents’ movements in a public transportation system. It is designed to accommodate any kind of regularly scheduled transportation that follows a specific route. Compared with the pedestrian simulation module, this module’s implementation is simple. The module is able to model multiple transportation modes within a single simulation, simply by varying the scheduled departure times from transit stops, and by varying the transit vehicles’ speed.

Module Inputs

The transit network is represented internally as a series of routes, each representing the path that a transit vehicle follows between stops where passengers can board or disembark. The route is described as a series of points representing waypoints along the route. The transit vehicles are assumed to move in straight line between these points. Each route is assigned three parameters:

- vehicle speed: the speed that the vehicle moves along this particular segment.
- passenger capacity: number of people that can travel on this route per departure.
- a list of departure times that define when a transit vehicle leaves the beginning of the route.

Each route represents a single direction: for those routes that should be bidirectional, the operator needs to define two overlapping routes, one for each direction. This allows the model operator to specify different schedules for each direction.

The public transit module accepts agents sent by the Physical Simulation Controller using the accept Agent message. This causes the module to place the agent at the start of the nearest transit route. The module then receives direction from the agent Controller in the form of a move to message. The module determines, based on the current location of the agent and its desired destination, which transit route the agent should be assigned to.
During the simulation, the module updates the location of its agents on a set interval (set as a parameter- typically every 5 seconds of 'real’ time). At each route’s departure time, the module checks to see if there are agents waiting for a vehicle. If so, it will then move them along the route at the transit vehicle’s given speed. The module sends position events to the rest of the simulation framework, specifying where all of the agents are in the simulated landscape.

The public transit module is both extremely simple in its implementation and very flexible: for the Schönreid study site, it is used to simulate 3 different types of public transit by specifying different parameters. These modes include gondolas, trains and local buses. Chairlifts, for instance, have a very small capacity (10 agents), but a very high number of departures (every 30 seconds from 9am to 5pm.) Trains, on the other hand, have a very large capacity but infrequent departures.

Figure 5.4: The different kinds of public transit routes used on the Schönreid study site. The public transit module simulates all three modes by specifying different speeds, capacities and departure times for each route and transportation mode.
Visual Quality Modelling

Chapter 6

One of the primary goals of the Vizagent framework is to integrate visual quality concerns within an agent-based recreational modelling framework. In order to do this, there are two main tasks: determining what the agents can see as they move through the simulated landscape, and determining how what they see influences their satisfaction with a particular route.

6.1 Calculating what an agent can see

Determining what agents see is accomplished by the Visual analyzer module. This module, implemented as a separate C++ program from the rest of the Vizagent framework, is at its core a rendering engine that can generate detailed 3D visualizations from geographic data. It is based on the OpenSceneGraph rendering engine [16], a software framework for realtime 3D visualization. The visibility calculations rely on recent advances in hardware-based 3D rendering. Current video cards, found in every new PC, are designed to perform highly detailed rendering calculations in hardware and produce many 3D renderings every second (the precise number of renderings depend on the complexity of the scene and the resolution of the resulting image.) While not explicitly designed to perform visibility analyses, determining the visibility of objects in three dimensional space is a prerequisite of 3D graphics techniques.

Using these techniques for visibility calculations was first described by Gross [33], and further refined by Bishop [11].

Modern rendering techniques require that all 3D objects to be drawn need to be described as a series of geometric primitives (primarily triangles, but also rectangles and triangle strips.) Using matrix transformations, these triangles are projected mathematically onto a two dimensional view plane, based on supplied information on the viewer’s position and field of view. Each triangle is then rasterized (converted into pixels), and each pixel’s distance from the viewer is calculated. This depth is compared against an internal software buffer, the Z-buffer, which records the distance from the viewer of all previously drawn primitives. If the new pixel is closer to the
eye than the previously recorded pixel, the current primitive is used to overwrite both the main rendering buffer (which, at the end of the rendering process, contains the final perspective image of the objects) and the Z buffer. In addition to displaying the final rendered image on the screen, both buffers can be read by the software program allowing it to analyze what was visible from that particular viewpoint. For a good overview of the rendering process, see Akenine-Moller et al [1].

As described above, current PCs have video cards that have customized chips designed to accelerate the the primitive projection, rasterization and buffer comparisons. This, coupled with software rendering techniques [1] provided by the Open Scene Graph library means that the visual renderer can perform many view calculations per second (using current hardware, upwards of one hundred views per second, depending on the complexity of the objects being modelled.)

The OpenSceneGraph rendering engine requires that all objects be be specified as 3D geometry. As a result, the visual analyzer module reads standard 2 dimensional GIS data and translates it into 3D geometry. The module:

- generates a terrain model from a raster dataset of height values;
- places trees on this model based on polygons that outline forested areas - Associated data in the GIS coverage specify the height and density of the forest stands;
- creates geometry for paths and roads based on line coverages;

![Figure 6.1: Diagram of a modern Z buffer based rendering system [72].](image-url)
6.1. Calculating what an agent can see

- extrudes polygon coverages representing objects such as buildings - building heights are specified in the coverage data;
- places 3D models (modelled externally in modelling packages such as Sketchup [66] or 3D Studio [2]) at specified locations indicated by a point-based GIS coverage.

The modeller specifies the location and type of the various GIS files using an ASCII setup file.

The module uses the above information to generate a visual model of the landscape. It is important to point out that the module, given enough modeller effort, could generate highly realistic visual representations of the landscape. In addition to providing viewers with a more detailed scene of agents in the landscape, this could allow the modeller to test subtle issues such as the properties of different tree species with respect to screening, or the impact of a shrub layer on a recreationist's perception of visual quality. However, the amount of visual detail required is linked to the requirements of the visual quality model (see section 6.2) used to interpret what the agent sees. As the currently implemented visual quality model is quite simplistic, relying primarily on the average view distance and broad categories of landscape elements, a highly detailed model of the landscape is unnecessary.

In addition to generating a static landscape from GIS files, the module listens to positional events broadcast from the rest of the Vizagent framework. These events provide the positions of agents as they move through the simulated landscape. If any of the agents are visible from the specified viewpoint, the module includes them in the rendered image.

The module has two main modes: it functions as a 3D viewer that allows the end user to see the agents’ movements on a given model run, and a mode that generates images from the perspective of the agents themselves. This allows the module to determine and measure what each agent can see as it moves through the simulated landscape.

In the 3D viewer mode, the location of the camera, or viewpoint, is controlled by the end user using a typical mouse-based control. A submode of the Viewer mode can also attach the camera to the viewpoint of a specific agent (identified by its unique id), causing the camera to follow the agent as it moves through a model run. The viewer mode is primarily used to observe the progress of particular model runs, and for explaining the model results to others.

In the second mode, arguably the most important mode for the Vizagent framework, the camera position is determined by the agents’ movements. As the framework receives agent position messages, it changes the camera location to be at the agent’s eye position. The agent’s view direction is determined by subtracting its current location from its most recent position. The module generates a single image from this perspective, which, rather than being output to the screen, is read back by the module for analysis of what the agent can see from its current position.

Rather than outputting the resulting image to the video screen (the resulting video feed would be non-sensical, as it would be a rapidly flickering series of disconnected images each showing
the perspective of a different agent), the module reads the resulting buffers from the video card, and analyses the resulting pixel images.

While the module could create highly realistic images from the agents’ perspective, visually realistic images are of little use for classifying and identifying the contents of the resulting images. The buffers read back from the video cards at the end of the rendering process are pixel buffers—they contain no topological information which would allow software to easily analyze what objects or types of objects are present in the image. In order to facilitate a software based analysis of the images, the module assigns unique colour values to each object or type of object. During the rendering phase, the module turns off shading, shadows and textures, which means that objects are rendered in exactly the same colour value as specified originally. This allows the module to easily identify which objects are visible, and how much of the agents’ view they occupy.

One largely unanswered question for calculating what agents can see from a particular point is what angle of view to use? Although the typical human field of view is almost 180°, in practice human perception is much narrower than this, depending on the individual’s age, cognitive load and other factors [78]. Determining the appropriate field of view for a modelled agent in a recreational hiking environment is even more complex, as it is assumed that hikers turn their heads and stop for breaks to admire the scenery. As Ervin and Steinitz [26] point out, there is a large gap between determining what one is able to see, and what one actually sees, perceives, and judges. While it is trivial for the visual analyzer module to set an arbitrary field of view (it is specified by a single parameter for the module, ranging from 0 to 360°), determining an appropriate angle of view is beyond the scope of this particular research project. For the Schönried site, it is assumed that the agents see everything around them, as they are moving relatively slowly (at walking speed) through a scenic environment. Accordingly, the field of view parameter was set to 360°.

![Figure 6.2: Three images that demonstrate how the visibility calculations are performed. To the left is the “realistic” image of a rendered scene. The middle image is the false colour version, objects are coloured based on their type. The right is a stylized Z-buffer, which indicates how far away the various objects are from the viewer. Using the right two images, the visual analyzer module can determine how much of a given landscape type category is visible, and how far away from the viewer the various categories are.](image)
6.2. Visual Quality Modelling

This image-based approach to calculating visibility is very different from the GIS-based visibility calculations used in typical GIS analyses and other models such as some iterations of the RBSIM model [41]. The RBSim approach relies on standard GIS grid-based visibility calculations [32] [70] to calculate what can be seen from any given point. In RBSim's particular approach, objects in the landscape are assigned to a particular grid cell in the terrain's digital elevation model. Each cell's height is extruded based on the height of the objects contained with the cell. While this simplification of the landscape makes the calculations of which areas are visible from any given point relatively simple and efficient to make, it limits the resolution of any results, and limits the variety of visibility analyses that can be done. In particular, these kinds of GIS-based calculations only indicate which locations are theoretically visible from a given location. It is not easy to deduce from this information what a viewer actually can see from that location. This contrasts with the image-based method used in the vizagent framework, which can, depending on the resolution of the input data, indicate which objects are visible, in what proportion, and how far away they are from the viewer.

6.2 Visual Quality Modelling

As described by Ervin and Steinitz [26], there is a large leap to be made between determining what a person could see, and what that person actually sees as they move through a landscape. Another big leap is interpreting what the individuals saw, and translating it into a quantitative measurement usable in a computer model. This is the role of the visual quality evaluator function, which a key component of the Visual Quality Module.

This function reads the false colour and depth buffers and interprets their contents in order to provide a visual quality score and a summary of landscape types that can be used to calculate landscape variability.

6.2.1 Landscape Variability

Landscape variety was identified by the on-site survey (see section ??) as one of the most important factors influencing hikers' route choice. Landscape variability is something that must be measured over time. As the visual analyzer module is stateless (it maintains no information about the agents' prior movements, and only has knowledge about what the agent can currently see), it does not calculate a variability score. Instead, it calculates the percentage of the agent's field of view that is occupied by various landscape types. These percentages are sent to the rest of the framework as Experience events, where they are stored and interpreted as part of a larger route by the Decision-making modules (see section 4.2.2.) In the Schönrried example, the landscape type categories are:

- trees
Chapter 6. Visual Quality Modelling

- meadow
- structures
- sky

Each category (with the exception of sky), is further divided into 3 subcategories based on distance from the viewer: foreground, middleground, and background. The three threshold distances are somewhat arbitrarily set, and are defined as a startup parameter for the module. They are currently set according to Bishop et al. [11] at 200m and 500m (i.e. any objects nearer than 200m are classified as foreground, pixels between 200m and 500m are middleground, while any pixels more than 500m away are classified as background.

As with any diversity metric, care must be taken when selecting the categories for the landscape type. In this case, they are somewhat arbitrary, although they do represent a decent tradeoff between what is easily modelled and what intuitively makes sense (the three main experience types in the Schöried landscape (village, field, and forest) are easily distinguished with this particular selection of categories.)

6.2.2 Visual Quality Metric

The second responsibility of the visual quality evaluator function is to generate a score that indicates the visual quality of a given location. As described in Chapter 2, there are a number of approaches to doing this in the literature. The 3D rendering technique used by the Visual Quality Module lends itself particularly well to the pixel based approaches to visual quality modelling [62] [11]. These models use measurements of what an individual can see (from photographs or computer-generated images) as inputs to a regression model that calculates a relative score of visual quality (or, more precisely, scenic beauty.)

While it is clear that local calibration and validation of such a model is generally required (the two above cited models were based on studies in North America and Scotland), designing and implementing such a human subjects based study is outside of the scope of this work. As a placeholder, the module uses a variation of a particular regression model (Landscape Preference Model A) described by Bishop et al. [11]. This regression model, which the author’s claim achieves high level of visual quality prediction (adjusted $R^2$ values of 0.944), uses only a few measurable parameters from its rendered image to calculate scenic beauty (see figure 6.3.)

The model is primarily dependent on depth variables: scenic beauty is largely dependent on how far one can see in a given scene, and how much variation in depth there is within a scene. Even though the model has not been calibrated to local conditions, it performs surprisingly well, with areas identified heuristically as scenic receiving higher scores than areas in the lower valley, which are not perceived as being as scenic.
### 6.3. Speed Issues with Visual Analysis

#### Table 3.

<table>
<thead>
<tr>
<th>Unstandardised coefficients, ( B )</th>
<th>Standard error</th>
<th>Standardised coefficients, ( \beta )</th>
<th>( t )</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Constant)</td>
<td>-4.596</td>
<td>0.413</td>
<td>-11.132</td>
<td>0.000</td>
</tr>
<tr>
<td>Greatest depth difference</td>
<td>4.133E-04</td>
<td>0.000</td>
<td>6.926</td>
<td>0.000</td>
</tr>
<tr>
<td>Mean depth of vegetation</td>
<td>2.741E-03</td>
<td>0.001</td>
<td>5.219</td>
<td>0.001</td>
</tr>
<tr>
<td>Water pixels—midground</td>
<td>3.854E-05</td>
<td>0.000</td>
<td>1.422</td>
<td>0.193</td>
</tr>
<tr>
<td>Vegetation pixels—midground</td>
<td>-3.086E-03</td>
<td>0.000</td>
<td>-6.194</td>
<td>0.000</td>
</tr>
<tr>
<td>Vegetation pixels—background</td>
<td>3.390E-05</td>
<td>0.000</td>
<td>2.333</td>
<td>0.048</td>
</tr>
</tbody>
</table>

Sig., Significance. 

\( R = 0.984, R^2 = 0.968, \) adjusted \( R^2 = 0.944, \) standard error of the estimate = 0.1424.

Figure 6.3: The visual quality model used to calculate visual quality from the depth images. (Model and Figure from [11])

### 6.3 Speed Issues with Visual Analysis

While the image-based approach does provide for a much finer grained analysis of what the agents can see than a GIS-based analysis, the increased resolution does have implications for the overall speed of the simulation. Even with hardware acceleration in the video card, the system is limited in the number of visibility calculations it can perform per second. This limit is determined by the complexity of the model, the capabilities of the video card, and even by the structure of the visual model (using different techniques to model individual objects such as buildings can have significant impacts on the rendering speed.)

In the Schönreid example, using (for 2007) moderately powerful video cards (NVIDIA Geforce 6500 GT), the visual analyzer module can generate and analyze, on average, 55 images per second. With even a small agent population of 300 agents, this means that it requires roughly 5 seconds to analyze what all of the agents can see. If the framework is sampling this information every 5 seconds of simulated time, it means that the simulation runs in essentially real time—requiring 5 seconds to simulate 5 seconds of real time. As each simulation run represents an entire day (or, more accurately, about 12 hours of a day, as hikers are only present from 8am to roughly 8pm), and the simulation approach requires up to 30 runs to converge on (or learn) a solution of path choices, the amount of time spent calculating visibility is a serious liability for the overall Vizagent framework, causing it to require weeks to complete a simulation.

As a result, two alternate strategies were explored for speeding up the visibility calculations: distributing the visibility calculations across multiple computers; and pre-calculating visibility and caching the results.
6.3.1 Distributed Visibility Calculations

While the original reason for implementing the visual analyser module as a separate module was to take advantage of software frameworks that were only available using the C++ language, the resulting implementation means that the module is largely decoupled from the rest of the framework. From the perspective of the simulation, the module only receives information about agent locations, and only rebroadcasts information on the agents’ visual experiences. The decision to pass information between the framework and the module with TCP network messages, even if both programs are located on the same machine, means that where the module is physical located is irrelevant to the framework: it can be on the same computer, or located somewhere more distant. This means that it is relatively simple to distribute the visibility calculation across multiple instances of the module, each running on a separate machine (and having access to its own video card.) As the module sends its results back to the framework as discrete messages, the process of splitting up the computational load across multiple computers is invisible to the rest of the framework.

Each instance of the program is given a range of agent IDs that it is responsible for. Each instance receives position events for all of the agents as normal, and updates their location

Figure 6.4: Diagram of how the visibility calculations can be handled by a single instance of the visual analyzer module or distributed across multiple instances of it. Arrows indicate event flows. To the rest of the framework, this process is completely transparent.
within the landscape. Only when an instance receives a position event for one of its specific agents does it compute what that particular agent sees and sends an experience event back to the rest of the framework via a network message.

This approach works well: as the overhead of broadcasting messages to multiple instances is quite low (each instance of the visual analyzer module registers itself independently with the event broker module), the process should scale almost linearly with the number of computational nodes added. This is based on informal tests with a small test cluster of 3 machines.

However, setting up a visibility cluster of sufficient size to be effective enough to significantly bring down the calculation time is likely to be prohibitive for most recreational simulations. Typical general-purpose compute clusters available to researchers are “headless”, in that they do not have a discrete video card of their own. The test cluster of 3 machines was not sufficient to reduce the time spent on visibility calculations enough to be worthwhile. Given enough computational nodes, however, this approach could be very successful.

6.3.2 Pre-Calculating Visibility

The second strategy for speeding up the visibility calculation relies on the fact that for the two types of visual quality analysis described above, visual quality and landscape variety, the agents are evaluating a static landscape. The only thing that changes over the course of a simulation is the position of other agents. While in many recreational areas, in particular wilderness areas in North America, encounters with other recreationists is a big factor in determining visitor satisfaction, in other areas, such as the Schönried study area, it is less of a concern.

For those areas where precisely calculating whether an agent can see other agents at any given point is not necessary, pre-computing the visual quality of the landscape is a relatively easy way to reduce the amount of time needed to run a simulation. In the case of the pilot Schönried study, interactions with other hikers was not considered a major problem- as the study area is a cultural landscape, with pastures, barns, ski lifts and restaurants distributed across the hiking area, there is no expectation among visitors that one would not see other hikers.

With this strategy, the views are prerendered at predetermined sample points across the landscape. In the case of the pilot study, as agents do not leave the path network, these sample points are evenly spaced along the path network at 10 minute intervals. The precomputed images are calculated and summarized to an intermediate level (counting the number and distance of cells based on the categories used by the Landscape Variety calculation) by the Visual Analysis Module. The results are written to files. During a simulation run, these files are read by a simplified version of the Visual Analyzer Module. This simplified version, like the full version, listens to agent position events and responds with visual quality and landscape variety events. Instead of calculating the values one the fly, however, the module determines which of the pre-rendered points is closest to the agent’s current position. It uses the information associated with
that point to calculate a visual quality score and summary for visual variety.

As finding the nearest point from within a set of thousands of points can be an expensive operation computationally, the module uses a quad-tree structure [25] to spatially sort the points.

One issue that makes pre-computing visibility slightly more complex is if one assumes that agents have less than a 360° field of view. As the agents move in different directions, the pre-computed visibility calculations need to be able to reflect the agents’ direction of movement and field of view. In order to facilitate this functionality, the Visual Analyzer can be set up to divide the pre-computed views into distinct slices, each occupying a set number of degrees of the field of view from any given point. Given an agent’s position and view direction, these slices can be reassembled to encompass the desired view direction. Depending on the degree of accuracy required, the slices should be smaller than the desired field of view.

Although this approach to dividing and recreating the pre-computed views was implemented, as it was decided to use a full 360° field of view when calculating visual quality, it was not required for the pilot study.

Figure 6.5: Locations of pre-computed visibility calculations are arranged in a quad-tree structure in order to allow them to be loaded efficiently by the module in response to agent movements. (Diagram from [25].)
6.3. Speed Issues with Visual Analysis

Figure 6.6: At specified intervals along the path network, slices of views are precalculate (left.) In this example, each slice represents 45° of a field of view. Given an agent view direction (middle), the system selects the nearest slices that give it a field of view that meets its specified minimum angle. (In this example, a field of view of 105° was generated, even though a minimum 90° was requested.) Although it is not as accurate as calculating visibility on the fly, it is a good tradeoff between accuracy and speed.

The resulting speed increase is more than sufficient for the needs of the pilot study- instead of the visual quality calculations holding up the rest of the simulation, pre-computing the visual quality scores allowed the simulation to run on a single machine without slowing the simulation perceptively.
Chapter 7

Test Site: Schönenried, Switzerland

7.1 Introduction

As described in Section 1.1, the study site near Schönenried, Switzerland was used both to explore the scope of the problem of recreational modelling in non-constrained landscapes, and as a test to evaluate the usefulness and reliability of the model for real planning situations.

7.2 Calibration Studies

In order to better understand the usage patterns in the study area, two studies were conducted on site during the August high season in 2002 and 2004. In the first study, groups of hikers were interviewed at the end of the day and asked to describe their day’s activities. Further questions were asked to ascertain their motivations for choosing their particular route. The second study involved counting hikers as they passed certain key locations at different times of the day. Both studies were primarily exploratory: they were designed to collect information that could be used to calibrate the simulation, rather than to be statistically significant.

7.2.1 Hiker Interviews

In late August 2002, a survey was administered to hikers in the study area. Groups and individuals were approached at the main exit points of the area (including the two main parking lots at the base of the chairlift/gondola, and at the train station). Respondents who agreed to participate were asked the questions from a standardized survey. Depending on the language of the respondent (the region is near the border between French and German speaking Switzerland, and there are a number of international visitors), the survey was administered in German, English or French. The English version of the survey is included in Appendix ??.
The questions included basic demographic questions, whether or not visitors were regular visitors of the area, and whether or not the hike they had just completed had met their expectations. Most importantly, respondents were asked to recall their most recent hike in the area (this was almost exclusively the hike/walk they had just completed: in a couple of cases, they were from hikes completed the day before.) They were asked to draw, on a supplied map, the route they had planned to take, and where they had deviated from that plan, which route they had actually taken. An example of these maps, with a typical response, is included in figure 7.1. Respondents were asked to indicate which factors influenced their choice of routes and, if they deviated...
from their chosen route, which factors caused them to change their plans.

In total, 68 interviews were conducted over the course of a week. As many of the respondents can be considered representative of a larger hiking group, it is estimated that they represent over 120 hikers in total.

The trail maps were digitized into a GIS map (see Figure 7.2.) The spatial arrangement of the hikers reflects informal observations: visitors tend to ride up the ski-lifts to the top and either walk along relatively flat ridge-lines, enjoying the view, or hike back down to the parking lot. A significant minority (15%) take a short ride at on a public transit vehicle (bus or train) to reach their start point or to return to their point of departure.

Other key findings (see figure 7.3) include the fact that 43% of the hikers had hiked their indicated route previously, and that 79% indicated that they would use the same route again. Anecdotally, those who indicated that they would not visit the same route again did so for two main reasons: the route, while satisfying, was not sufficiently unique to warrant another visit; and because many did not want to revisit routes more than once on principle.

Figure 7.2: A GIS map showing the routes identified by the 63 survey respondents. Increasing thickness indicates increased numbers of users on that particular segment.
### Chapter 7. Test Site: Schönried, Switzerland

#### 7.2.2 Hiker Counts

In August 2004, a further study was conducted in order to refine the understanding of how hikers moved through the landscape. This study involved counting hikers as they passed key points in the landscape. As the recreational area is quite large, with many potential entrances and exits, it proved impossible to achieve a precise count of all hikers given available resources. Instead, key locations were chosen that were thought to be critical for understanding the dynamics of hiker movement in the area. Some spots chosen were particularly busy spots, or were key funnel points where many routes converged. Others were chosen because of their proximity to key infrastructure (such as parking lots, at the tops of chairlifts/gondolas.) At each of the key points (identified in figure 7.4), observers were asked to record the number of hikers who passed, the time, and, for those spots with multiple routes entering and leaving the spot, from which direction they arrived and on which route they left.

Similarly to the first study, this study was not intended to be completely representative or statistically valid: it was intended to provide more information to help the modeller understand how many agents were typically using certain routes at different times of day. Observers moved around over the course of a day and the week, which means they did not collect data at a single point for a significant amount of time. Nor did they have time to collect two samples at simi-
lar times on different days. While this sampling methodology might not have been exhaustive, it serves its purpose of providing more detail for the purposes of generating a synthetic population of agents.

7.3 Building a Synthetic Population

Based on the two studies outlined above, a synthetic population of agents was created. As described in Section 3.5.2, this involves creating a hypothetical collection of agents that reflects the kinds of hikers that were observed in the actual recreational landscape. This involves specifying the agents’ characteristics and expectations (including both the target and importance value for each of the expectations.) It also involves specifying an activity chain for each agent. The activity chain includes an ordered list of activities that the agent will complete, including suggestions of how long these activities should take. In addition, each agent needs to have its starting location and time specified by the modeller.

Figure 7.4: Locations where hikers were counted as they passed during the Hiker Count survey in August 2004. Each spot was observed for a minimum of two 4-hour periods over a week.
7.3.1 Number of Agents

Many, if not most, of the observed hikers were parts of small groups, ranging in size from 2 to 30 (the latter was a class group.) The majority of groups were pairs or triples. Rather than trying to model group behaviour, it was decided that, for this simulation, each agent would represent an agent or a group of agents. Each agent represents from one to six individuals (it was decided to ignore the school group, as this was a single observation that did not repeat itself over the two weeks of observation.) Although it was difficult to estimate precisely, it was determined that, on a typical sunny August day, there were approximately 450 hikers in the study area. Based on this, a population of 250 agents was created, with an emphasis on singles and pairs.

The agents were assigned random starting locations from a set of 4 possible locations: the parking lot at the base of the Rellerli gondola (30%), the parking lot at the base of the Horneggli chairlift (25%), arriving by train at the Schönried station (25%) and at the base of the Zweisimmen gondola (15%).

7.3.2 Agent Types

In order to make the synthetic population more manageable, the agents were grouped into a few categories based on observed behaviour and the first survey. These groups included:

- **Day hikers** (15% of the agents): those interested in a long hike (6 hours). This group was most interested in the length of the hike. While they have a tolerance for difficult terrain, it is not a requirement for them. This is expressed as a moderate target value for difficulty, with an low importance value for difficulty. They have a high isolation target value, but like other agents in this simulation, they are not particularly bothered by the presence of others in their vicinity, so isolation has a low importance.

- **1/2-day hikers** (40%): similar to the day hikers, but with a shorter desired hike length. This group, which is the largest by far, is further divided into those who seek a restaurant after their hike and those who do not (this is specified in their activity chain.)

- **Fitness oriented hikers** (15%): smaller groups, typically one or two people, who actively seek difficult terrain. As the study site does not have any particularly difficult terrain, this desire is typically expressed by them hiking both up and down the mountain during the hike. They are assigned a much higher desired difficulty level (0.6), and also have a much higher importance value (0.8). They are assigned hike lengths of 3 hours.

- **Casual visitors** (30%): this includes those who seek short strolls (30 minutes to 2 hours), with little to no difficulty.

All agents were given identical values for the visual quality and landscape variety expectations.
7.4. Calibrating the Model

7.3.3 Start Times

In order to avoid artificial congestion as agents are introduced to the simulation, agents are assigned entry times randomly within one of two typical entrance times (from 9am to 11am, and from 12:30 to 2pm.) The exception to this was for hikers arriving by train, who were all introduced at the time a train arrives.

7.4 Calibrating the Model

The description above describes the final version of the synthetic population. In reality, it was derived iteratively as part of the calibration process. Considerable time was spent adjusting both the agent properties (expectations and activity chains, primarily) and the magnitudes of the values being generated as experience events by the physical simulations and the Visual Analyzer module.

In the end, the model, when applied to the existing landscape, produced results that approximate the observed patterns (see Figure 7.5.)

![Figure 7.5: Traces of agent movements using the calibrated model on the existing landscape.](image)
While the results are roughly similar to the observed patterns (see Figure 7.1), they differ in some significant, albeit explainable, ways:

- While the simulation did discover popular routes such as the one between the Zweisimmen gondola and the Horneggli restaurant (50% of the agents using this route walked down to Schönried to catch the train back to Zweisimmen, while the others walked the return trip), the simulation was not able to find the significant route from [insert name of bus stop here] to Rellerli, which in reality was favoured by many day hikers (it is the only true 6 hour hike in the area.) It was not found due to the fact that the post bus stop in question was just slightly outside the study area boundaries, and that the bus route was not included in the public transit module’s model.

- Numerous agents chose routes that would not have been observed in reality. This mostly involved individuals back-tracking on routes that paralleled each other in close proximity. In reality, the routes would have been perceived as being too similar to each other. The simulation, however, was not able to ascertain that they were very similar routes that were visible from one another, so chose them as two different routes.

- While the overall pattern of movement, as shown in the trace map (Figure 7.5), there were many choices that were temporally unrealistic: i.e agents passed by a restaurant only to make an abrupt turn around shortly thereafter. This was largely due to the fact that the activity chain was supplied as fixed input to the model, which meant that the agents strove to meet the given time budgets as closely as possible. This flaw would be alleviated by either spending more time tweaking individual activity chains, or, preferably, by implementing the activity generator as the third decision-making module. This would allow the simulation to modify the activity chain dynamically to make it fit with the specifics of the scenario.

7.5 Testing the Model on New Scenarios

Once the model had been calibrated so that the outputs of the status quo scenario largely corresponded to observed behaviour, the model was applied to two scenarios that radically changed the landscape to explore how the model would react in a changed landscape.

In running the two scenarios, all agent and model parameters were kept constant. Aside from the main changes described below, the major elements of the modelled landscape (path network, topography, location of facilities, etc.) were also maintained across all three scenarios. This allows one to assess how well the model is calibrated, and how reacts to landscape change, which is the primary goal of the framework.

The two new scenarios were:
7.5. Testing the Model on New Scenarios

- **Reforestation**: All areas above a certain elevation (1350m) were reforested. This scenario was intended to represent an exaggerated consequence of reducing or eliminating farm subsidies in the area, which play a large role in maintaining the existing landscape structure of open meadows interspersed with small copses of trees.

- **Summer closure of Chairlift/Gondolas**: All three lifts are considered closed and removed from the model. This scenario is intended to reflect an existing challenge to the area: as the winter ski season becomes less viable due to climate changes, there will likely not be enough capital available to do periodic major maintenance without public support. This scenario is meant to explore what the implications could be for summer tourism if the lifts are not operated in the summer.

All three scenarios (including the status quo scenario) are illustrated graphically in Figures 7.6, 7.7 and 7.8.

![Figure 7.6: The status quo scenario, with existing forested areas and the three lifts that operate in the summer season.](image-url)
7.5.1 Scenario Results

Reforestation Scenario

As can be seen in figure 7.9, the resulting movement patterns are significantly different from the status quo scenario. Agents’ routes are more randomly distributed across the landscape. In particular, there is much more activity in the valley bottoms. This results from the agents searching for routes with a low difficulty value. The routes in the valleys are as easy as those on the ridgetops. In this scenario, however, rather than providing expansive views (and therefore high visual quality) the ridges are completely enclosed by forest. In fact, the valley bottoms perform better than the ridge lines in terms of visual quality, as there are more open areas and more buildings, which leads to a much higher landscape variety score.

Not only have the movement patterns changed, there has also been a marked difference in the satisfaction level of the agents. As Visual quality was given a high importance to all agents, the lack of areas with high visual quality means that agents’ satisfaction are considerably lower than in the status quo.
7.5. Testing the Model on New Scenarios

Figure 7.8: The lift closure scenario, where all three lifts have been removed as possibilities from the simulation.

Closure of Chairlift/Gondolas

As expected, Figure 7.10 demonstrates that the agents’ behaviour is significantly changed by the removal of the lifts in the third scenario. Most of the agents remain in the valley bottoms, as they are unable to reach the higher elevation areas without exceeding their desired difficulty score. Only those agents in the Fitness Seeking group, and a few of the day hikers, venture very far above a minimal elevation level.

One result of staying in the valley bottoms is that encounters between agents increase, which reduces the isolation value of their routes, which further decreases their satisfaction with the area. This is not a significant decrease however, because the importance of this value is quite low, in keeping with typical Swiss tolerances of other visitors in recreational areas. More detrimental to their satisfaction, however, is the steep reduction in their visual quality scores, as most do not reach areas with expansive views that are so critical to the visual quality model.
Chapter 7. Test Site: Schönried, Switzerland

Figure 7.9: Traces of the agents’ movements in the reforestation scenario.

7.5.2 Model Performance

All three scenarios were run for 45 iterations. Only the status quo scenario required this many runs to converge on a stable result: the other two converged more quickly. Each model run takes from 7 to 10 minutes to complete, which means that a complete scenario is run overnight.

It should be pointed out, however, that in the current implementation, very little effort was spent optimizing the software for speed. This is a contrast to Gloor’s work [30], where creating a framework that could accommodate large numbers of agents was a design goal.

In all situations, the desire to get a fully functioning framework was preferred over early optimization. While overnight model runs are not preferable, especially during the calibration phase, it is not unworkable. As this implementation of the Vizagent framework was intended as a proof of concept, if it proves useful in other situations, then a re-writing of some portions in order to increase performance would be warranted. Specific areas where performance improvements could be made include:

- switching from an interpreted scripting language (python) to a compiled language such as c++, or another interpreted language such as Java that has been optimized for speed;
7.5. Testing the Model on New Scenarios

Figure 7.10: Traces of the agents' movements in the chairlift/gondolas closure scenario.

- various algorithm improvements, especially for geometric calculations (often simple, brute force methods were used in the interest of getting a complete system working;)

- a better event passing method should be investigated. Currently, most messages are passed between modules as python objects, which has a high cost for instantiating and cleaning them up after use. While this is by far the simplest approach to program, it is very inefficient given the large number of events being passed around in the framework.
Chapter 8

Conclusions

As a proof of concept, the Vizagent framework as described in this work is a success. It demonstrates that agent-based modelling framework can be implemented in such a way that it is sensitive to landscape change and that it can be used to model how recreationists change their behaviour in response to his change. While others [5] have articulated the desire for a similar framework, and even describe limited implementations of parts of such frameworks, this is the only fully functioning agent based recreational model that is able to integrate visual quality concerns with behavioural modelling.

Even though it is a fully functioning model, it should be emphasized that the Vizagent framework at present is more akin to a basic shell that needs to be fleshed out with better submodels and algorithms. There are a number of areas where the chosen algorithms are suspected to have little justification behaviourally: they were simply introduced as ‘filler’ until better algorithms could be discovered or researched. A small selection of areas needing refinement include:

- the visual quality model
- link between slope and perceived difficulty
- how best to integrate all of the perceptual factors (difficulty, isolation, visual quality) into a single assessment of a route
- threshold for isolation: how many encounters with others is too much? How close does another person have to be to be considered an interaction?
- landscape variety model

The number of areas needing refinement is indicative of one of the most persuasive arguments against using this kind of model for recreational simulation: it is too complex to be practically useful. In order to achieve realistic movement patterns, the modeller has to adjust a large number of factors that have never been researched in the literature. While some of these knowledge
gaps could be addressed by more empirical studies, for others it is difficult to even conceive of an experimental design, even if cost were no concern (e.g. determining how individual visual experiences combine into a single visual experience over a specified time.) Even if one could conduct studies to fill some of these gaps, it is yet another large step to understand how all of these factors interact with each other to create a recreational experience.

This is particularly true when one considers the typical financial resources available to a recreational planner. Unlike transportation modelling (from which this work draws a lot of inspiration), there are very few resources available for collecting behavioural data. This is one reason where even a simple model like RBSim (which has a lot fewer variables to calibrate) has not been used outside of a very narrow academic community: even with its simple recreational areas, it requires more resources than are available.

While there are many elements of the implementation that are based on simplistic interpretations of the author’s experiences, it is interesting to note that the test study is, after considerable calibrating, able to produce plausible results. While it is not clear if the calibration for the Schöried area could be transferred to other locations and problem types, it does indicate that the approach is worth pursuing in order to understand if the amount of variables and submodels are actually required. This indicates that the approach is doable, but that some effort should be spent simplifying the number of variables required.

Another issue is whether or not this kind of modelling is appropriate for the kinds of landscapes exemplified by the Schönried study: landscapes where hiker interactions are not much of a concern. Agent modelling makes more sense where agent behaviour depends on the behaviour of other agents: otherwise, no emergent behaviour can be expected. Other recreational areas where capacity is a real concern, and where visual interactions between individuals are as important as physical interactions might be more appropriate: busy public parks are one such example.
List of Figures

1.1 Representational photograph of the study site. The site is a typical mix of forested areas mixed with grazing areas for cattle. Many of the grassy areas double as downhill skill trails in winter. .................................................... 6

2.1 Diagram of route choices from the Broken Arrow RBSim simulation. Path networks modelled using RBSim, like WUSM before it, are relatively small and have few path choices. .................................................... 11

2.2 Conceptual Diagram of the TRANSIMS framework [54]. Each of the main tasks in the simulation are implemented as independent modules that communicate with each other. The framework invokes the modules sequentially, moving from left to right. .................................................... 14

3.1 Simple example showing how agents start out randomly, but learn better routes over time. After one iteration (top), the agents follow the shortest route on the way to the peak. After multiple iterations (bottom), they ‘learn’ that other routes are better suited to their individual expectations. Figure from [30] ............................. 22

3.2 An overview of the Vizagent framework’s modules, divided by category. ........ 23

3.3 A comparison of the principal conceptual data flows (top) and how the data actually flows in the framework via events (bottom). While this appears to add an extra layer of complexity, in reality, it makes writing the software much more consistent as there is only a single interface for passing messages. It also means that new modules can be inserted very easily into the framework without significant software development effort. .................................................... 25

3.4 The types of GIS data that are required by the simulation. To the top is the underlying terrain data; in the middle are the key locations; at bottom is the path and public transit network. All are specified in standard GIS formats (ESRI Shapefile and ASCII Grid.) .................................................... 28
3.5 Example of all of the data that needs to be specified for each agent in the simulation. Agent 1 (on the left) represents a hiker who is interested in the physical activity of a hike, in addition to the landscape’s scenic qualities, while Agent 2 (on the right) is only interested in a short walk with a nice view and a lunch in a restaurant. 30

3.6 Graphical user interface of the python-based viewer program. It loads in the results of a simulation run and shows the agents movements over the course of the simulation. It can also display underlying GIS information, such as path networks and points of locations. The user can step through the simulation and examine each agent to determine its current plan and expectations. 33

4.1 Simplified XML Plan. The simulation system dynamically generates a new plan for each agent every day. 36

4.2 Schematic representation of each Decision-Making Module’s contribution to the plan generation process. As the Overall Controller queries the modules from top to bottom, the agent’s plan gains increasing resolution. In the current implementation, the Activity Generator’s output (a non-spatially specific activity chain) is specified manually by the modeller. 37

4.3 Comparison of the different types of networks maintained by the framework. In red (top) is the network of key locations, which is maintained by the location generator. In blue (middle) are the routes, managed by the router. Each link in the router’s network represents a path from one junction to another. The black routes (bottom) are the physical locations of the routes in geographic space. In order to more easily provide realistic movement choices, the router inserts waypoints from the actual physical routes between the nodes specified by its routing network. 40

4.4 Pseudo-code for the traversal part of the Route Selector Algorithm 43

5.1 A true continuous space simulation would allow an agent with a starting position of ‘A’ to cut the corner to reach its destination at ‘B’, if the intermediate area was passable. In Gloor’s implementation this is not possible, as it defines waypoints at every curve of the path. 48

5.2 The two input rasters for the physical simulation. The raster on the left represents the terrain’s elevation, and is used to calculate the slope of an agent’s route. The raster on the left represents walkability, and is scaled from 0.0 (not passable) to 1.0 (easy to walk over.) 50
5.3 Example of the forces that are used to calculate the environmental forces on agents when they are in close proximity to each other. The direction of the forces is always perpendicular to the agents’ desired direction. Their magnitude is derived from the walkability raster. This causes the agents to avoid each other while also staying, as much as possible, on the trail. Figure from [30].

5.4 The different kinds of public transit routes used on the Schönreid study site. The public transit module simulates all three modes by specifying different speeds, capacities and departure times for each route and transportation mode.

6.1 Diagram of a modern Z buffer based rendering system [72].

6.2 Three images that demonstrate how the visibility calculations are performed. To the left is the “realistic” image of a rendered scene. The middle image is the false colour version, objects are coloured based on their type. The right is a stylized Z-buffer, which indicates how far away the various objects are from the viewer. Using the right two images, the visual analyzer module can determine how much of a given landscape type category is visible, and how far away from the viewer the various categories are.

6.3 The visual quality model used to calculate visual quality from the depth images. (Model and Figure from [11]).

6.4 Diagram of how the visibility calculations can be handled by a single instance of the visual analyzer module or distributed across multiple instances of it. Arrows indicate event flows. To the rest of the framework, this process is completely transparent.

6.5 Locations of pre-computed visibility calculations are arranged in a quad-tree structure in order to allow them to be loaded efficiently by the module in response to agent movements. (Diagram from [25].)

6.6 At specified intervals along the path network, slices of views are precalculate. In this example, each slice represents 45° of a field of view. Given an agent view direction (middle), the system selects the nearest slices that give it a field of view that meets its specified minimum angle. (In this example, a field of view of 105° was generated, even though a minimum 90° was requested.) Although it is not as accurate as calculating visibility on the fly, it is a good tradeoff between accuracy and speed.

7.1 A typical respondent’s map of both their actual route taken.

7.2 A GIS map showing the routes identified by the 63 survey respondents. Increasing thickness indicates increased numbers of users on that particular segment.

7.3 Responses to the question “What factors influenced your choice of route?”
7.4 Locations where hikers were counted as they passed during the Hiker Count survey in August 2004. Each spot was observed for a minimum of two 4-hour periods over a week. ................................................................. 71

7.5 Traces of agent movements using the calibrated model on the existing landscape. 73

7.6 The status quo scenario, with existing forested areas and the three lifts that operate in the summer season. ................................................................. 75

7.7 The reforestation scenario, where all areas above 1350m have been reforested. 76

7.8 The lift closure scenario, where all three lifts have been removed as possibilities from the simulation. ................................................................. 77

7.9 Traces of the agents’ movements in the reforestation scenario. ........................ 78

7.10 Traces of the agents’ movements in the chairlift/gondolas closure scenario. . . . 79
Bibliography


Curriculum Vitae
Curriculum Vitae - Stuart Duncan Cavens

PO Box 344
Roberts Creek, BC
V0N 2W0
Canada

duncan@cavens.org

Citizenship: Canadian

Education

2002 – Present  Doctor of Science (in progress)
Institute for Spatial and Landscape Planning
Faculty of Civil Engineering
ETH Zürich, Switzerland

Dissertation Title: An Agent-based framework for modelling the impact of landscape change on recreational behaviour

2000 - 2002  Master of Science
Individual Interdisciplinary Graduate Studies Program
(Landscape Architecture, Computer Science, Forestry)
University of British Columbia

Thesis Title: A semi-immersive visualisation system for model-based participatory forest design and decision support

1995 – 1999  Bachelor of Landscape Architecture (Honours)
Landscape Architecture Program
University of British Columbia

Lester B. Pearson College of the Pacific
Victoria, B.C.

Related Employment

2007 – Present  Researcher
Design Centre for Sustainability / Collaborative for Landscape Planning
UBC Landscape Architecture

- Emerald Hills Sustainable Urban Village Project: Developed urban simulation software to measure sustainability performance at the site / neighbourhood scale.
- Centre for Interactive Research on Sustainability / World Urban
Forum: Developed an interactive 3D visualization kiosk for the proposed CIRS building “Canada’s most sustainable building.”

2002 – 2007  
**Researcher**  
Institute for Spatial and Landscape Planning, Faculty of Civil Engineering  
ETH Zürich, Switzerland  

*Key Responsibilities include:*  
- Project Coordinator and Researcher on the Swiss National Science Foundation funded project “Planning with Autonomous Agents and Virtual Alpine Landscapes”  
- GIS and software development support for numerous projects  
- Software design and development of web-based Environmental Monitoring Tools for Cantonal Government of Solothurn  
- Design, construction, software development and installation of an interactive landscape planning exhibit for ETH Zürich’s 150 Anniversary. Installation was a large 3 screen display with custom built control panel, allowing more than 200,000 visitors over 3 weeks to explore different landscape planning options for a valley near Lucerne.  
- Grant Writing and Editing

1998 – Present  
**Independent Contractor**  
SDC Design  
Vancouver, B.C.  

As a private contractor, worked on a variety of industry and university projects, including research projects. Representative projects include:  
- Development of an interactive forestry design tool as part of the NSERC/Industry funded “Local Landscape Ecosystem Management Simulator” project. (2002-2005)  
- GIS modeling of landscape ecology metrics for bird habitat on Vancouver Island. Funded by the University of Victoria (2001-2002).  

2000 – 2002  
**Research Associate**  
Collaborative for Advanced Landscape Planning  
Faculty of Forestry  
UBC  

*Key Responsibilities included:*  
- Designed and coordinated installation of the Landscape Immersion Lab (LIL): a CFI funded $100,000 large screen collaborative theatre  
- Designed and implemented the CALP Visualisation System, a software framework that integrates different stand- and landscape-level forestry models and generates realistic
visualizations of large-scale landscapes

- Developed CALP Forester, a model-based
- Developed (in both hardware and software) and tested user interface devices for use in the LIL
- Developing Project Proposals and acting as a liaison between CALP and other university projects and departments

1999

**Software Developer**

Waterfront Employers of B.C.

Vancouver, B.C.

- Designed and implemented business software as part of a large software engineering project. Responsible for training existing developers in Object-Oriented software development techniques.

1995 – 1997

**Software Developer**

Blue Heron Software

Sechelt, B.C.

**Publications**

**Refereed Journals**


In Preparation


**Refereed Conference Proceedings**


Invited Book Chapters


Conference Proceedings (Refereed Extended Abstracts)


Technical Reports


Teaching

Sessional Lecturer

2008  Advanced Graphics – LARC 535B
      University of British Columbia, Landscape Architecture Program

2007  Vertical Studio – LARC 504C
      University of British Columbia, Landscape Architecture Program

2006 -  Advanced Computer Aided Design – LARC 535D
2008              University of British Columbia, Landscape Architecture Program

2006       Introduction to Geographic Information Systems – LARC 535B
2007       University of British Columbia, Landscape Architecture Program

2005 -  Introduction to 3D Modeling – LARC 535B
2007       University of British Columbia, Landscape Architecture Program

Thesis Supervision

2003  Daniel Kistler, Mental maps for mobility simulations of agents
      Computer Science M. Sc. Thesis, ETH Zürich (co-supervised with Kai Nagel)

2003  Mathias Walker, Realistic real-time visualization of large-scale network-oriented multi-agent simulations
      Computer Science Thesis, ETH Zürich, (co-supervised with Kai Nagel)

Teaching Assistantships

1998 –  Open Space Planning Studio, Landscape Architecture Program, UBC
2002
1997 – 1999
Introduction to Computers in Landscape Architecture, Landscape Architecture Program, UBC

**Academic Awards**

<table>
<thead>
<tr>
<th>Year</th>
<th>Award Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002</td>
<td>UBC University Graduate Fellowship - $16,000 (declined)</td>
</tr>
<tr>
<td>2002</td>
<td>UBC Landscape Architecture Faculty Book Award</td>
</tr>
<tr>
<td>1999</td>
<td>Charles and Jane Banks Scholarship ($950)</td>
</tr>
<tr>
<td>1998</td>
<td>UBC Landscape Architecture Construction Prize</td>
</tr>
<tr>
<td>1996</td>
<td>UBC Scholarship ($1880)</td>
</tr>
<tr>
<td>1996</td>
<td>Charles and Jane Banks Scholarship ($950)</td>
</tr>
<tr>
<td>1995</td>
<td>UBC Dean’s Prize in Landscape Architecture</td>
</tr>
<tr>
<td>1992-4</td>
<td>B.C. Scholarship – Lester B. Pearson College of the Pacific ($20,000 / year)</td>
</tr>
</tbody>
</table>

**Selected Community Involvement**

<table>
<thead>
<tr>
<th>Year</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005 - Present</td>
<td>Founding Member, Civil Society Development Project (Vancouver-based Thinktank)</td>
</tr>
<tr>
<td>2005</td>
<td>Local Organising Committee, Our Shared Landscape Conference, Ascona, Switzerland</td>
</tr>
<tr>
<td>2001 – Present</td>
<td>Founding Member, Roberts Creek Cohousing</td>
</tr>
<tr>
<td>1998-1999</td>
<td>Invited Representative on UBC Board of Governor’s Committee “Principles for Physical Planning at UBC”</td>
</tr>
<tr>
<td>1995- 1999</td>
<td>Member, UBC Student Environment Committee (President 1996-1997)</td>
</tr>
</tbody>
</table>