Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 13 (2014) 56 – 65

Procedia Technology

SLACTIONS 2013: Research conference on virtual worlds - Learning with simulations

Simplifying crowd automation in the virtual laboratory of archaeology

Luís Miguel Sequeira[a,]*, Leonel Morgado[b], E. J. Solteiro Pires[c]

[a] PhD Student at University of Trás-os-Montes Alto Douro, Rua do Monte Leite, 468 – 4º Dtº, 2765-496 Estoril, Portugal
[b] INESC TEC (formerly INESC Porto) – Universidade Aberta, Lisbon, Portugal
[c] INESC TEC (formerly INESC Porto) – University of Trás-os-Montes e Alto Douro (UTAD) Pole, Quinta de Prados, Vila Real, Portugal

Abstract

Virtual archaeology projects have been evolving to go beyond a mere reconstruction of architecture and artefacts of heritage sites: human interaction with the environment is also an object of research for historians and archaeologists. Methodologies like the London Charter propose that historians and archaeologists, in close collaboration with technical teams, lead virtual archaeology projects to guarantee the credibility and scientific validation of the result. The question is how to allow historians to model crowds on their own, if lacking the required skills to programme complex artificial intelligent-driven autonomous agents. In this article a method is proposed, currently under development, to allow non-programmers will be able to successfully model crowds using very simple tools that do not require formal programming knowledge but can still provide convincing results. The underlying idea is to employ concepts borrowed from computer games, whose interfaces are targeted to non-experts and adapt them to the specificities of virtual world platforms like Second Life® and OpenSimulator. Moreover, some limitations and ideas for further extension are discussed.

Keywords: Second Life; OpenSimulator; Artifircial intelligence; Virtual archaeology; Genetic algorithms; Bots

* Corresponding author: *Email address*: lms@esoterica.pt

doi:10.1016/j.protcy.2014.02.009

1. Introduction

In the past decade, historians and archaeologists have been experimenting with computer models to recreate three-dimensional representations not only of heritage sites, but also of the human population that inhabited or used those sites. This is an extension of the concept of **virtual archaeology** [1] — a blend of techniques and methods employed by historians and archaeologists using computer models for visualising cultural artefacts and heritage sites, and which lately has also often included crowd simulation. Bogdanovych & Rodriguez [2] argue for the need of including the simulation of virtual agents as part of any virtual heritage project.

Ulicny & Thalmann [3], for example, represent in their CHARISMA project '(...) a virtual population of worshippers performing morning Namaz prayer inside a virtual mosque', with detailed technical implementations explained by the same authors in a different paper [4]. The difficulties of simulating crowds are also addressed by the Ancient Malacca Virtual Walkthrough, 'a project that focuses on the modelling and visualization of Malacca city in 15th century' [5].

To allow virtual agents to navigate in a virtual heritage site, some authors propose the inclusion of semantic data embedded in the 3D models, for subsequent use by a rule-based engine to help crowds navigate in the virtual environment. A typical example of such an approach is demonstrated by an EPOCH-sponsored project recreating the crowds of Romans populating the ancient city of Pompeii [6].

Similar techniques are also used in augmented reality (AR) where the public, walking physically through the ruins of a heritage site, expects the AR hardware not only to show them the rebuilt architecture, but also a representation of the human inhabitants [7].

Finally, this approach is also used for Serious Games, depicting realistic crowd behaviour inside authentic historical recreations of several scenarios, as can be found on a review by Anderson et al. [8].

2. Early limitations of the virtual archaeology projects

Early approaches to virtual archaeology projects in the 1990s, according to Frischer et al. [9], resulted in models that were created mostly by computer programmers and 3D modellers. Historians and archaeologists were just employed as consultants, but often did not make the crucial decisions, nor validated the final models: they were merely invited to 'comment' upon them.

For this reason, methodologies like the London Charter [10] propose that virtual archaeology projects add a degree of validation and establish rules for documentation about the choices made, to allow for rigorous testing of hypothesis, public discussion, and critical analysis. Under this model, historians and archaeologists lead the project, formulate hypothesis, test theories, provide written feedback across all stages, and do the final validation. Historians and archaeologists supervise the technical teams (3D modellers and programmers). This model has shown to create more credible projects, and further reasoning about this kind of methodology virtues were proposed earlier by Sequeira & Morgado [11,12].

One constraint of this scenario is that, since historians and archaeologists usually have little or no computer programming skills, and programmers and modellers might not be sufficiently versed in history, there can arise some communication difficulties among the team.

However, in traditional 3D virtual archaeology projects, models are rendered in a graphics pipeline that might take weeks or months to produce the final models. Any changes might require the process to start from scratch.

An alternative is to use **virtual worlds**, which enable end-user content generation. In such virtual worlds, content created is instantly rendered for all participants in the scene, and can be built collaboratively in real time. Thus, historians can log into a virtual world like Linden Lab's Second Life® (SL), into one of several created using an open-source counterpart platform, OpenSimulator, or yet some other similar platform, and discuss with the development team as they build the models. As soon as an element is finished, historians are able to move it around, position it correctly, and visually align it so as to validate some hypothesis, while simultaneously opening the project to public visitors. This concept of virtual laboratory of archaeology has been successfully deployed by many projects and critically analysed by Morgan [13] (a survey of various projects in Second Life and OpenSimulator was done by Sequeira & Morgado [12]).

Projects like *Rome Reborn* [14,15] have met enormous success, but what immediately becomes noticeable is that those projects are 'empty' — no inhabitants crowd the streets of Imperial Rome. However, it is argued that historians should also capture a degree of understanding on how the inhabitants of the heritage site would have interacted with each other. This has been done in projects like the reconstruction of the City of Uruk [16,17], where intelligent agents follow certain rules to depict common activities of local inhabitants, and allow human visitors to interact with them to ask them what they are doing. Not only does this allow historians a better understanding of why the architecture looks like it does, but also because it provides human visitors with a higher degree of educational value.

Morgan [13], however, warns that non-playing characters (NPCs) should not be used as merely decorative elements or 'props'. In her critical analysis, she discovered that there was often the temptation of limiting NPCs to chatting about their own environment, but be presented decontextualized from the heritage site; this, she argues, not only lowers the credibility of the project, but *'[...] turning people of the past into mere mouthpieces for their architecture diminishes the rich potential of reconstructions to impart information about complex lifeways'*. The Uruk project avoids this trap, and so do the reconstructions done by Kennedy et al. [18]. In the Crystal Palace project, NPCs which chat with each other and with visitors are employed to provide an understanding of what Victorian visitors of the Crystal Palace exhibit of Pompeii thought about it, based on actual records of interviews done at the time [19]. Thus, NPCs, if employed at all, have to be meaningful for the project.

The problem here, of course, is how to programme them, because historians and archaeologists might not have the necessary skills. The above-mentioned projects all require specialised teams of programmers to do the complex artificial intelligence (AI) programming, and leave the validation details to the historians.

In this article, a suggestion is proposed for a model where historians, instead of being passive towards the AI simulation of NPC, become the 'programmers' themselves, but without requiring writing a single line of traditional computer programming code. In other words, just like historians can move objects around in a virtual world (and change environment settings) to formulate hypothesis and test theories, without requiring any assistance from the technical teams (once the models have been built, historians can simply use them and place them on the required spots), this article suggests that they should be given an easy-to-use tool to simulate crowd behaviour.

3. The City & Spectacle Project

City & Spectacle: A vision of pre-1755 earthquake Lisbon [20] is a project promoted by CHAIA/Universidade de Évora with Beta Technologies as its computational development partner. It is a virtual archaeology project to recreate a memory of a heritage site that does not exist any longer: the Baroque city of Lisbon, utterly destroyed by a devastating earthquake, tsunami, and subsequent fire on November 1st, 1755.

Besides merely modelling architecture, the project also includes the simulation of specific events, namely, a night at the Opera House, attending the coronation ceremony of King Joseph I, a bullfight at the Royal Court (Terreiro do Paço) and following a processional across the streets of Lisbon — as well as the earthquake itself. Ideally, NPCs would populate the streets of Lisbon, walking from home to work and back, and interacting with other NPCs they would find on their way. Once one of the 'special' events is flagged as starting, some NPCs are expected to break from their routine and attend the events.

4. Algorithms for AI-driven NPCs applied to Second Life/OpenSimulator

A discussion of the typical algorithms employed to appropriately model such a framework is beyond the scope of this article. It is an area of active research for at least the past two decades [21–23], especially because of computer games [24–27], which have similar specifications, but similar techniques are also employed in military simulations [28,29]. Each approach makes certain assumptions about the environment in order to present a 'best' Artificial Intelligence (AI) algorithm for developing autonomous agents. Early work by Wooldridge & Jennings [30] divided the then existing solutions in two categories: planning and non-planning. In the first case, the scenario is deemed to be reasonably static, and some sort of rule-based engine is used to determine in advance the 'best' solution for movement/interaction within the environment. While individual agents will have exceptions to deal with sudden changes, the simulation is expected to have a certain degree of stability. Non-planning methods, by contrast, do not

make such assumptions. They are best employed in highly dynamic environments with constant changes (which is the case of user-generated virtual worlds like the ones already mentioned above), where there is little metadata in the environment (also typical of these worlds), and where sensors and movement commands are not precise. The latter is specifically the case of robotics [31–33], but, perhaps surprisingly, it also applies to SL/OpenSimulator worlds, because unlike other virtual world platforms the programmers do not have direct access to the simulation itself, which creates lag, delays, and unpredictable extraction of content from the scene (even using a new module for OpenSimulator, in the server itself, there would still be delays and unpredictable results).

A full description of the proposed approach, and its critical analysis is, again, not the purpose of this article, but only the concern about how historians can actually use it for their crowd simulations. It should suffice to say that this article expresses the belief that a hybrid approach between swarms [34–36] (to establish constraints about collisions and setting goals as attractors) and genetic algorithms (GA) [27,31,37–40] (to select the 'best' rule for a certain behaviour) seems to be favoured by the reviewed literature as providing the best results.

The proposed framework departs slightly from established algorithms due to the characteristics of SL/OpenSimulator. Second Life does not provide a direct application programming interface (API) to control NPCs, mostly because Linden Lab felt that this would be exploited for abuse. OpenSimulator, by contrast, has no such issues, and NPC control can be done in the built-in language, Linden Scripting Language (LSL)[†], which has severe performance constraints, due to running on a virtual machine. There are alternative extension modules for OpenSimulator (but not for SL), which work well but require OpenSimulator grid operators or independent server managers to add those modules. Further, module developers may not ensure compatibility with future OpenSimulator versions. In the case of the *City & Spectacle* project, some of the early work was done in Second Life, then ported to an internal OpenSimulator-based grid, sometimes exporting the full content to a public-accessible grid, Kitely[‡], for public visits Thus, full access to the simulator code may not be always available, and, as such, a decision was made to avoid tying the solution to a specific infrastructure.

In view of the above, the communications module with SL/OpenSimulator uses libopenmetaverse, a popular C# third-party library which supports the communications protocol used by both SL and OpenSimulator, and acts as an end-user viewer — but exposing to the programmer the full range of functionality as an API. From a computational development perspective, we emphasize that this library only allows acting as a user with a viewer— it does not have special access to routines that are deployed at the simulator level.

To extract environment information from the simulated scene, traditional methods on other platforms simply query the object database. Under Second Life the database is not directly accessible at the viewer level (in OpenSimulator it is, but that involves tying approaches both to OpenSimulator and to a specific version of it). Instead, through the underlying physics engine (Havok under SL; Open Dynamics Engine or Bullet Physics under OpenSimulator), LSL function calls allow information to be extracted of the environment. This is similar to the approaches used by physical robots, but has also been successfully used in virtual worlds and games [27].

To send the extracted information to the reasoning engine, we use HTTP requests. Because these are very limited in LSL, a terse, non-standard format is used (as opposed to XML or JSON [§]). One advantage is that libopenmetaverse is able to retrieve object data based on their universally unique identifiers (UUID), which can be stored on the reasoning engine, so that additional information from sensor/collision/ray-casting data coming from LSL can usually be limited to sending UUID and little else, thus saving precious lookup calls in LSL.

Conversely, to move avatars around, the reasoning engine sends information to the libopenmetaverse module to move the NPCs. Movement, however, just like with physical robots, is not precise. A curious characteristic of remotely manipulating intelligent agents in SL/OpenSimulator is a principle of uncertainty: the more precise the information that is required, the more the system has to be polled in shorter time frames, and the more lags and delays these requests create, thus actually retrieving less precise information. This paradox requires adapting the sensing and movement algorithms so that they deal with partial, unreliable, and imprecise data — again, not unlike

[†] Linden Scripting Language is an event-driven language developed by Second Life's creators, which is used inside SL and OpenSimulator.
[‡] Readers may wish to visit it at http://bit.ly/lisbon1755.
[§] XML and JSON are two popular human-readable formats to pass data objects between client and server Web applications.

physical robots — which imply a certain amount of compromising. Based on the first author's previous experience in manipulating avatars in this way [41,42], the 'best' approach seems to be to give commands to a NPC to move in a certain direction and just check when it collides with an obstacle (which might not have been retrieved by the perception module), and then signal the reasoning engine to engage in new rule selections based on the new input. In fact, this closely resembles the way humans navigate in the virtual world environment, especially when it is very "laggy", and objects might not have all been downloaded. Similar difficulties have been reported in other projects [43,44], which mention that they successfully dealt with the issue, but don't detail the approach that was employed.

However, detecting collisions in LSL works under Second Life, but this functionality is as yet unavailable under OpenSimulator. Therefore, the prototype polls sensor data and retrieves the NPCs' velocity (if it is zero, it might have collided against an object) instead of polling the underlying physics engine for a collision. This, unfortunately, renders collision detection unreliable.

5. The reasoning engine

The reasoning engine follows the work done by Silva et al. [45] at the University of Trás-os-Montes e Alto Douro (UTAD) and Instituto Politécnico do Porto (IPP) to define not only an ontology for describing the implementation of intelligent agents in virtual worlds, but a conceptual framework for such an implementation, which has been successfully deployed by UTAD/IPP in some projects [48,49].

The engine includes an auxiliary communications module, as explained before, developed on top of libopenmetaverse to provide communications with SL/OpenSimulator, and parts of the module are directly implemented in LSL and running as scripts inside objects in the virtual world itself; also, NPCs have attached to them an 'invisible bubble' which allows them to use sensors and the ray-casting function to extract information about its environment, and which also communicates with the rest of the application via HTTP request calls.

Currently, the libopenmetaverse module is being temporarily implemented using RESTbot[**], a software library originally developed by the Canadian company Pleiades and which has subsequently been kept up-to-date by some volunteers and released as open source software. It allows a web-based application to communicate with it via RESTful calls[††], which, in turn, get converted to the Linden Lab viewer protocol to extract information from the virtual world environment and to move NPC avatars around. In the future, however, the module will be ported to Silva et al.'s own module providing similar capabilities. It requires a server running Microsoft's .NET framework over Windows or Novell's Mono framework over Linux/Mac OS X.

The remaining modules are a web-based application developed in PHP[‡‡] and can run on any webserver that supports PHP (currently the application uses Nginx[§§]/PHP-FPM[***]). The backend database is currently SQLite3[†††] (accessed via an abstraction layer) and might be ported to MySQL[‡‡‡].

The core is composed of a classical subdivision of perception module, selection module, and behaviour module. Perception deals with extracting information from the surroundings and arranges it in an internal representation. In our case, the perception module gathers information via RESTbot, which gets periodically polled; and directly from LSL scripts inside OpenSimulator, which call it directly every time they have sensed an extraordinary event (proximity to an obstacle or another avatar).

[**] Code is available at http://code.google.com/p/restbot/. The word 'REST' in RESTbot refers to *Representational State Transfer*, defined by Wikipedia as 'a [software] architectural style that abstracts the architectural elements within a distributed hypermedia system.'

[††] That is, application calls that use the REST philosophy. See footnote **.

[‡‡] PHP, which stands for 'PHP: Hypertext Preprocessor' is a widely used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML (cited from http://www.php.net).

[§§] nginx (pronounced: 'engine x') is a lightweight HTTP/reverse proxy server, written by Igor Sysoev, popularly deployed on systems with a small memory/CPU footprint (source: http://nginx.org)

[***] PHP-FPM (FastCGI Process Manager) is one form of implementing PHP application control between a web server (in this case, Nginx) and actual execution of the PHP-based application. See also http://php-fpm.org/

[†††] SQLite is a software library that implements a self-contained, server-less, zero-configuration, transactional SQL database engine (cited from http://www.sqlite.org/). It is popular in embedded systems (e. g. mobile phone applications).

[‡‡‡] MySQL is, according to Wikipedia, the world's second most used open source database engine.

The selection module runs the genetic algorithm to figure out the 'best' action for each run; actions are typically 'move forward/left/right' or just 'rotate by some angle'. The fitness function depends on the length of the longest, unimpeded line of sight (as calculated by the ray-casting functions in LSL) and the nearness of the current goal (which we will see shortly), as well as some additional weighting coming from the two last modules. Certain objects in the virtual world environment also act as goals or attractors (in the sense of swarm attractors) and they flag their position to this module, which uses them to calculate and order rules according to a GA. Currently, a classical GA is being implemented, but other non-classical algorithms might also be employed to compare results.

The behaviour module, once a rule has found by the GA, will flag RESTbot to move the avatar around, if a movement is the best choice encoded by the rule, or another behaviour might be triggered. In terms of SL/OpenSimulator, this mostly means playing a gesture or animation (for example, if the NPC waves or greets another NPC or human) or eventually sitting down on some object (which will be explained in a later section).

With the core modules, NPCs are able to navigate in an environment of which they are only partially aware, and deal with collisions and simple interactions (avoiding other NPCs, for example, like in swarms). Notice that, due to the constant dynamic changes of the environment, is was considered wasteful to implement a 'memory' of past interactions or movement sequences.

6. Borrowing ideas from games

This project follows the methodology of Design Science [46,47] — the search for an artefact that fulfils specifications — in our case, to allow historians and archaeologists to create crowd simulations based on intelligent agents without the need to programme anything in traditional computer code.

While the section above permits movement and collision management without any additional rules, goals and interactions are not contemplated. The next module deals with **motivation**, and it works together with the selection module to feed the GA with appropriate weighting for the rule selection.

Here we consider the following analogy, typical of strategy simulation games: Each NPC is assigned a certain **energy level**. As they move around, they lose energy, and, at some point, they will have to change their strategy or goal in order to search for a way to recharge their energy. This is very similar to the approach used in robotics, where robots will also track the energy level of their batteries and need to move to a recharge station periodically [32], and virtual simulations of robots, where similar concepts are employed [48,49]. Both approaches use GA to 'train' robots, and the expected emergent behaviour was observed: robots, real and virtual, will maximize their 'life' by appropriately returning to the recharge stations as their energy levels drop.

To simulate the motivation of going to work every day, we define a goal for earning **money**. These are special places in the environment where the NPC is able to increase its money level until it reaches a certain threshold (the 'daily wage') and loses some energy in the process, after which it will return 'home' for an energy recharge. This is done by having historians visually dropping special cubes on the ground — yellow for energy, green for money — and assign a NPC to them. The energy/money boxes include a simple slider, which allow them to define the rate of energy replenishment or amount of money earned, which can be adjusted dynamically— both the position, and the rate. Every time the cubes are changed, the motivation module is flagged, changes the appropriate information on the database (location and rate), and the GA on the selection module will re-calculate the fitness for its rules.

Historians can also model a behaviour where NPCs stay for some time at work, while energy gets depleted, but instead of returning home for a full recharge, they can go to the nearest tavern or market stall for refreshment. To allow for this behaviour, a special **home** box was added, which is the 'main' goal for a NPC to return to, but, while at work, if energy levels are not enough, NPCs might be able to search for nearby yellow energy cubes for a quick recharge. It is expected that this allows NPCs to walk longer paths while 'eating' on their way to and from work.

How to deal with the special events — leaving work and/or home to attend a processional, for instance? A possible approach is to add a fourth cube for **happiness**. While at work, NPCs earn money and lose some energy, but their happiness levels drop even quicker. To recover happiness, historians can drop red cubes inside of churches or other entertainment areas. Thus, it is expected that NPCs will sometimes feel 'bored' at work, leave their place (even if their 'daily wage' is not fully earned yet!) and wander around in search of some entertainment — while losing energy that way, and, perhaps, they might select a goal to get some quick energy at a tavern before returning to work. Or they might 'stray' away from the workplace so much that the GA selects the 'home' goal instead.

To simulate a large event, the historian just needs to drop a new 'happiness' cube that acts as a global attractor for all NPC. More interestingly, this cube can be attached to a moving NPC — say, the priest leading a processional — and then the other NPC will follow that priest, in the hope of getting some happiness that way.

The interactions between NPC belonging to different classes are handled by the **interaction module**. NPC belong to a certain 'class' — nobleman, priest, peasant, merchant, foreigner, beggar and so forth — and a webpage, tied into the interaction module (and which can also be accessed while the historian is logged in to the virtual world), will present a matrix of behaviour between classes. Initially, simple friend/foe/neutral relationships will be available. For example, noblemen and peasants are 'foes' — peasants will avoid to cross their paths with noblemen, but if they collide for some reason (a poor action selection by the GA, or due to some unpredictable 'glitch' in the communications, which did not capture the movement of the nobleman towards the peasant in time to avoid a collision), the nobleman hits the peasant, and it loses energy. By contrast, priests are neutral towards peasants; if a peasant collides with them, priests will grant them a blessing (raising happiness for a slight amount).

While the profiles are fixed and programmed in advance, the interaction matrix is not, and is open for the historians to fine-tune. For instance, further development might include an analogue range, from -1.0 ('hate/contempt') to +1.0 ('love/awe'). It is also possible that the interaction matrix acts as a 'template' for a specific class, but gets instantiated for each individual NPC, and adjusted according to the interactions they have. In that case, behaviour could subtly change among individuals, like having a nobleman that ordinarily does not strike peasants even if they cross their paths. Over time, repeated behaviour like this could make peasants stop and greet that specific nobleman when they are nearby, instead of running away, but still fear other noblemen and continue to avoid all others. Such improvements are left for future work.

7. Expanding the concept

Further refinement of the engine can lead to more interesting behaviour modelling. A first change would be to make each cube work only for a single NPC — when the first NPC collides with the cube, it sits down on it, activating it, and preventing other NPCs to 'use' this cube. Sitting is a special function in SL/OpenSimulator which also can trigger animations, so it could be used to give visual clues to visitors: one NPC at a money cube will perform some kind of work by running the proper animation. A yellow cube at a tavern or market stall will animate the avatar with eating gestures. The home cube will put the NPC to sleep.

But the point is to allow historians to create more complex behaviours. For example, imagine a NPC is working at the docks, and the GA triggers the need to walk around in search of some energy. There might be a nearby yellow cube, and the NPC move towards it, expecting it to be 'free'. But in the meantime, another NPC reaches that same spot first. The NPC now has two options: wait until the cube is free again, or search for a different one — which will result in a loss of further energy — or return to work, or return home. Whatever the choice, it is clear that the 'daily routine' of a NPC can quickly become anything but 'routine' — it becomes much more complex and unpredictable, but, and this is worth noticing, it does not become random or chaotic. By identifying the current active goals from a web backend, historians can see what is currently been triggered, and see if the expected behaviour is consistent with the simulation parameters — but changing conditions will often lead to surprising and unpredictable results, which will make the simulation much more convincing.

Contrast this to typical rule-based models. Most of them add a degree of 'randomness' to allow for diversity. But human observers will quickly understand the patterns: NPCs will attempt to repeat their usual tasks over and over again, but, at some point, they might not (because of the random factor). Bogdanovych et al. [44] use this approach, by breaking up specific goals in sections, which NPCs are expected to follow, but allow for exceptions to be simulated using a random number generator, forcing NPCs to temporarily follow a different set of rules instead.

In the scenario described in this article there is nothing that tells the NPC 'you have to go to work first, then wait a bit, move to the nearest tavern, get back to work, and if you have energy left and feel unhappy, seek a place of entertainment before returning home' — all this would require lots of rules to be written beforehand, and, no matter how 'simple' these rules are (or how easy to assemble them together using a programming language), it still means that historians would have to painfully assemble them and model each interaction for all cases and exceptions.

Instead, the proposed framework offers an alternative way to accomplish similar simulation detail without the pain of complex programming. Historians only need to drop cubes around, move them into place, set a slider, and

watch what happens. By changing the location of the cubes and the settings on the slider, they can refine their simulation — dynamically. They can fine-tune a specific area of the virtual heritage site, just addressing a few NPCs, while the rest of the NPCs are happily moving around and taking the existing environment and current layout of cubes into account. Indeed, historians can see NPCs approaching a specific cube, clearly with the intention of sitting on top of it, and just delete the cube — or place another one near the NPC — just to see what happens. Or they can delete a building, block a street, demolish a wall, and see how NPCs immediately take these changes into account. The simulation adapts dynamically to everything that happens in the environment.

A further development is to have 'combined boxes', which can make the simulation even more interesting. Consider the following scenario: a church provides happiness, but it provides *more* happiness if a priest is present. To model this, historians drop a money cube for the priest (meaning that priests earn their 'daily wages' by holding mass, and that acts as their motivation for being inside the church), as well as a few red cubes for happiness. When a priest NPC sits on its money cube, the red cubes get 'activated' — for instance, their rate of bestowing happiness increases dramatically. The expected behaviour of the rest of the NPCs is that they will try to prefer entering a church with a priest holding mass, because the rate at which happiness is increased will weigh more on the fitness function of the GA. But there might be a limit of just a few cubes available in the church; what will the GA select? It could wait until the cubes are free — but by then, the priest might have reached its 'daily wage' and leave its place, and thus 'deactivating' the happiness cubes. Similar approaches can be used to simulate a tavern or market stall: when the innkeeper is at work, the energy cubes are active, but when he's away, they are not.

'Combining' cubes that way requires a bit more work for the historians. Once they are dropped, they could be assigned to a certain class (for instance, only for priests), which is accomplished via the SL/OpenSimulator mechanism of sending a dialogue box with options to the 3D viewer. A further option would be to 'combine' nearby cubes, which can either be accomplished automatically (all cubes in a certain radius) or by inviting the historians to click on each cube that they wish to 'link' with the one currently be confirmed. This can even be done visually, by using particle beams linking the combined cubes, thus giving historians some feedback on what has been configured so far.

8. Conclusions and avenues for further work

Turning the visual programming of complex crowd simulation by historians into something very much like a game has some interesting possibilities. While the prototype, at the time of writing, is still far from functional, and has not underwent user acceptance testing, it should allow easy configuration of the whole simulation, and thus fulfil the initial requirements of the project. This, however, has not yet been validated.

Unlike strict rule-based systems, this model is expected to be much less 'predictable', while still maintaining 'realism': NPCs navigate the virtual world according to goals and motivations, and not randomly, but their behaviour, due to the ever-changing, dynamic nature of past interactions, will not be deterministic. The way that the crowd simulation is configured, by dropping cubes and setting sliders, is quite similar to many strategy/simulation games (like the popular *The Sims*). This can be used to develop educational games.

Further research could also focus on the 'cubes' themselves. Right now, these are supposed to be relatively static, and remain at the place where the historian has dropped them. But as was explained before, an interesting case is when the cube is picked up by an NPC and carried around. Thus, future development might have moving cubes (not 'carried' but autonomously moving); or cubes might interact by being attracted/repelled to each other.

One possible complication with this approach is the requirement of having many coloured cubes spread all over the scene of the simulation, being potentially difficult for historians to track the overall model. In fact, it can be argued that such a method is not practical for large-scale simulations with hundreds or thousands of intelligent agents. However, in the case of SL/OpenSimulator, system constraints limit the number of agents in a scene to a few dozens; thus, the proposed framework is expected to be adequate for small-scale crowd simulation in SL/OpenSimulator without being too confusing to follow for the historians setting it up, but might require a different approach for other platforms able to model large-scale crowds.

Also, this model is assuming somewhat 'modern' perspectives, due to its urban 18th century application scenario: concepts such as 'work' and 'money' may not be adequate motivators in earlier settings or, indeed, in rural settings of the same century. Defining sets of motivational/control cubes is a probable later requirement of historians and

possibly might lead to the ability of sharing and discussing different models of behavioural determinants as a source of research.

Left out of this model was the ability for humans to interact directly with the NPC and ask questions about their internal state, in a way similar to the work done by Bogdanovych et al. [17].The interaction matrix could also be extended by allowing NPCs to 'learn' behaviour from watching how other NPCs interact.

The Uruk project is also able to make much more complex interactions possible, for instance, by allowing NPCs to use tools to accomplish some tasks. In our model, tools were not employed.

Although SL/OpenSimulator imposes several constraints, most of the engine runs on a common Web development language, PHP, and could be adapted to work on different virtual world platforms. However, on platforms offering precise movements and perfect sensors that retrieve all data flawlessly and immediately, many of the approaches presented in this document would not make much sense. The approach described here is best suited for environments where data retrieval from sensors and movement commands are not guaranteed to have any precision, and, as such, the same approach might be more suitable for controlling physical robots than to simulate crowds in 'perfect' virtual worlds not subject to lag, delays, or partial data retrieval. Our intent is to later connect this approach to ongoing work that aims to render choreographies independent of the underlying virtual worlds platform [45].

References

[1] P. Reilly, Towards a Virtual Archaeology, in: K. Lockyear, S.P.Q. Rahtz (Eds.), Comput. Appl. Quant. Methods Archaeol. 1990, Archaeopress, Oxford, UK, 1991: pp. 133–139.
[2] A. Bogdanovych, J. Rodriguez, Virtual agents and 3D virtual worlds for preserving and simulating cultures, in: Z. Ruttkay, M. Kipp, A. Nijholt, H.H. Vilhjálmsson (Eds.), Proc. 9th Int. Conf. Intell. Virtual Agents (IVA '09), Springer-Verlag Berlin, Heidelberg, Germany, 2009: pp. 257–271.
[3] B. Ulicny, D. Thalmann, Crowd simulation for virtual heritage, in: Proc. First Int. Work. 3D Virtual Herit., 2002: pp. 28–32.
[4] B. Ulicny, D. Thalmann, Towards Interactive Real-Time Crowd Behavior Simulation, Comput. Graph. Forum. 21 (2002) 767–775.
[5] M.S. Sunar, M. `Adi M. Azahar, M.K. Mokhtar, D. Daman, Crowd Rendering Optimization for Virtual Heritage System, Int. J. Virtual Real. 8 (2009) 57–62.
[6] J. Maïm, S. Haegler, B. Yersin, P. Mueller, D. Thalmann, L. Van Gool, Populating Ancient Pompeii with Crowds of Virtual Romans, Eurographics. 0 (2007) 109–116.
[7] G. Papagiannakis, S. Schertenleib, B. O'Kennedy, M. Arevalo-Poizat, N. Magnenat-Thalmann, A. Stoddart, et al., Mixing virtual and real scenes in the site of ancient Pompeii, Comput. Animat. Virtual Worlds. 16 (2005) 11–24.
[8] E.F. Anderson, L. McLoughlin, F. Liarokapis, C. Peters, P. Petridis, S. Freitas, Developing serious games for cultural heritage: a state-of-the-art review, Virtual Real. 14 (2010) 255–275.
[9] B. Frischer, F. Niccolucci, N.S. Ryan, J.A. Barceló, From CVR to CVRO: the Past, Present and Future of Cultural Virtual Reality, in: F. Niccolucci (Ed.), VAST 2000, Archaeopress, Oxford, 2002: pp. 7–18.
[10] R. Beacham, The London Charter, (2006).
[11] L.M. Sequeira, L. Morgado, Virtual Archaeology in Second Life and OpenSimulator, in: L. Morgado, Y. Sivan, A.M. Maia, G.C. Matos, R.R. Nunes, D. Pedrosa, et al. (Eds.), SLACTIONS2012 — 4th Int. Res. Conf. Virtual Worlds — Life, Imagination, Work Using Metaverse Platforms, UTAD, Vila Real, 2012: pp. 59–66.
[12] L.M. Sequeira, L. Morgado, Virtual Archaeology in Second Life and OpenSimulator, J. Virtual Worlds Res. 6 (2013).
[13] C. Morgan, (Re)Building Çatalhöyük: Changing Virtual Reality in Archaeology, Archaeol. - J. World Archaeol. Congr. 5 (2009) 468–487.
[14] K. Dylla, B. Frisher, P. Mueller, A. Ulmer, S. Haegler, Rome Reborn 2.0: A Case Study of Virtual City Reconstruction Using Procedural Modeling Techniques, in: CAA 2009. Mak. Hist. Interactive., Archaeopress, Oxford, 2009: pp. 62–66.
[15] B. Frisher, Rome Reborn: A Case Study in the Digital Reconstruction of Historic Cities, in: Virtual Hist. Cities Reinventing Urban Res., Lisbon, 2010.
[16] A. Bogdanovych, S. Simoff, Establishing Social Order in 3D Virtual Worlds with Virtual Institutions, in: A. Rea (Ed.), Secur. Virtual Worlds, 3D Webs, Immersive Environ. Model. Dev. Interact. Manag., IGI Global, Western Michigan University, USA, 2010: p. 30.
[17] A. Bogdanovych, K. Ijaz, S. Simoff, The City of Uruk: Teaching Ancient History in a Virtual World, in: Y. Nakano, M. Neff, A. Paiva, M. Walker (Eds.), Intell. Virtual Agents - 12th Int. Conf., Springer Berlin Heidelberg, Santa Cruz, CA, USA, 2012: pp. 28–35.
[18] S. Kennedy, L. Dow, I.A. Oliver, R.J. Sweetman, A.H.D. Miller, A. Campbell, et al., Living history with Open Virtual Worlds: Reconstructing St Andrews Cathedral as a stage for historic narrative, in: M. Gardner, F. Garnier, C.D. Kloos (Eds.), Proc. 2nd Eur. Immersive Educ. Summit EiED 2012, Universidad Carlos III de Madrid, Departamento de Ingeniería Telemática, Madrid, Spain, Paris, France, 2012: pp. 146–160.
[19] N. Earle, S. Hales, Pompeii in the Crystal Palace: Comparing Victorian and Modern Virtual, Immersive Environments, in: EVA 2009 Electron. Vis. Arts Conf. Br. Comput. Centre, 6 - 8 July 2009, London, 2009: pp. 37–46.
[20] A.G. da Câmara, A. Pimentel, H. Murteira, P. Rodrigues, City and Spectacle: A Vision of Pre-earthquake Lisbon, in: 15th Int. Conf. Virtual Syst. Multimed. (VSMM '09), IEEE, Vienna, Austria, 2009: pp. 239–243.
[21] M. Cavazza, R. Earnshaw, N. Magnenat-Thalmann, D. Thalmann, Motion control of virtual humans, IEEE Comput. Graph. Appl. 18 (1998) 24–31.
[22] M. Funcke, A comparison of crowd simulation techniques, Thesis, Rhodes University, 2012.

[23] S. Zhou, D. Chen, W. Cai, L. Luo, M.Y.H. Low, F. Tian, et al., Crowd modeling and simulation technologies, in: ACM Trans. Model. Comput. Simul., New York; NY; USA, 2010: pp. 1–35.
[24] K. Getchell, A. Miller, R. Nicoll, R. Sweetman, C. Allison, Games Methodologies and Immersive Environments for Virtual Fieldwork, IEEE Trans. Learn. Technol. 3 (2010) 281–293.
[25] J. Snape, S.J. Guy, D. Vembar, A. Lake, M.C. Lin, Reciprocal Collision Avoidance and Navigation for Video Games, Intel Softw. Netw. (2012).
[26] S. Kim, S.J. Guy, D. Manocha, M.C. Lin, Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory, in: Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games - I3D '12, ACM Press, New York, New York, USA, 2012: p. 55.
[27] R. Graham, S. Sheridan, H. McCabe, Realistic Agent Movement in Dynamic Game Environments, in: DiGRA 2005 Conf. Chang. Views – Worlds Play, Digital Games Research Association DiGRA, 2005.
[28] F.D. McKenzie, M.D. Petty, P.A. Kruszewski, R.C. Gaskins, Q.-A.H. Nguyen, J. Seevinck, et al., Integrating crowd-behavior modeling into military simulation using game technology, Simul. Gaming. 39 (2007) 10–38.
[29] B. Zhou, K. Xu, M. Gerla, Group and swarm mobility models for ad hoc network scenarios using virtual tracks, in: IEEE MILCOM 2004. Mil. Commun. Conf. 2004., IEEE, 2004: pp. 289–294.
[30] M. Wooldridge, N.R. Jennings, Intelligent agents: theory and practice, Knowl. Eng. Rev. 10 (1995) 115–152.
[31] D. Floreano, F. Mondada, Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot, in: D.C.P. Husbands, J.-A. Meyer, S.W. Wilson (Eds.), From Anim. to Animat. — Third Int. Conf. Simul. Adapt. Behav., MIT Press, Brighton, UK, 1994: pp. 421–430.
[32] F. Mondada, D. Floreano, Evolution of neural control structures: some experiments on mobile robots, Rob. Auton. Syst. 16 (1995) 183–195.
[33] S. Nolfi, D. Floreano, O. Miglino, F. Mondada, How to evolve autonomous robots: Different approaches in evolutionary robotics, in: R.A. Brooks, P. Maes (Eds.), Artif. Life IV Proc. Fourth Int. Work. Synth. Simul. Living Syst., MIT Press, 1994: pp. 190–197.
[34] C.W. Reynolds, Flocks, Herds, and Schools: A Distributed Behavioral Model, in Computer Graphics, in: M.C. Stone (Ed.), SIGGRAPH '87, ACM Press, New York, New York, USA, 1987: pp. 25–34.
[35] C.W. Reynolds, Boids, (2001).
[36] C.W. Reynolds, Interaction with Groups of Autonomous Characters, in: Game Dev. Conf. 2000, CMP Game Media Group, San Francisco, California, 2000: pp. 449–460.
[37] R. Kovacina, D. Palmer, Swarm rule-base development using genetic programming techniques, in: Work. Auton. Swarm Program., John Carroll University, University Heights, OH, 2003.
[38] C.-F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design., IEEE Trans. Syst. Man. Cybern. B. Cybern. 34 (2004) 997–1006.
[39] E. Nonas, A. Poulovassilis, Optimisation of Active Rule Agents using a Genetic Algorithm approach, in: 9th Int. Conf. Database Expert Syst. Appl., Springer-Verlag, London, UK, 1998: pp. 332–341.
[40] N. Cole, S.J. Louis, Ch. Miles, Using a genetic algorithm to tune first-person shooter bots, in: Proc. 2004 Congr. Evol. Comput. (IEEE Cat. No.04TH8753), IEEE, 2004: pp. 139–145.
[41] L.M. Sequeira, L. Morgado, Mechanisms of three-dimensional content transfer between the OpenSimulator and the Second Life Grid® platforms, J. Gaming Virtual Worlds. 5 (2013) 41–57.
[42] L.M. Sequeira, Mechanisms of Three-Dimensional Content Transfer Between the Opensimulator and Second Life Grid Platforms, Mastership Thesis, LAP Publishing, 2012.
[43] S. Slater, D. Burden, Emotionally Responsive Robotic Avatars as Characters in Virtual Worlds, in: 2009 Conf. Games Virtual Worlds Serious Appl., IEEE, Coventry, UK, 2009: pp. 12–19.
[44] A. Bogdanovych, J.A. Rodriguez-Aguilar, S. Simoff, A. Cohen, Authentic interactive reenactment of cultural heritage with 3D virtual worlds and artificial intelligence, Proc. 9th Int. Conf. Intell. Virtual Agents (IVA '09). 24 (2010) 617–647.
[45] E. Silva, N. Silva, H. Paredes, P. Martins, B. Fonseca, L. Morgado, Development of platform-independent multi-user choreographies for virtual worlds based on ontology combination and mapping, in: 2012 IEEE Symp. Vis. Lang. Human-Centric Comput., Ieee, 2012: pp. 149–152.
[46] K. Peffers, T. Tuunanen, M. a. Rothenberger, S. Chatterjee, A Design Science Research Methodology for Information Systems Research, J. Manag. Inf. Syst. 24 (2007) 45–77.
[47] A. Hevner, S. March, J. Park, S. Ram, Design Science in Information Systems Research, MIS Q. 28 No. 1 (2004) 75–105.
[48] F.R. Miranda, J.E. Kögler, E. Del Moral Hernandez, M. Lobo Netto, An artificial life approach for the animation of cognitive characters, Comput. Graph. 25 (2001) 955–964.
[49] M.L. Netto, J.E. Kögler Jr, E.D.M. Hernandez, F.R. Miranda, WOXBOT: a Wide Open Extensible Robot for Virtual World Simulations, Comput. Graph. Geom. - Internet J. 3 (2001) 20–39.