Hybrid 3D Rendering of Large Map Data for Crisis Management

David Tully, Prof Abdennour El Rhalibi, Dr Christopher Carter, Dr Sud Sudirman

School of Computing and Mathematical Science
Liverpool John Moores University
Liverpool, UK
{ D.Tully@2008, A.Elrhalibi@, C.J.Carter@, S.Sudirman@ }.ljmu.ac.uk

Abstract: In this paper we investigate the use of games technologies for the research and development of 3D representations of real environments captured from GIS information and open source map data. Challenges involved in this area concerns the large data-sets to be dealt with. Some existing map data include errors and are not complete, which makes the generation of realistic and accurate 3D environments problematic. The domain of application of our work is crisis management which requires very accurate GIS or map information. We believe the use of creating a 3D virtual environment using real map data whilst correcting and completing the missing data, improves the quality and performance of crisis management decision support system to provide a more natural and intuitive interface for crisis managers. Consequently, we present a case study into issues related to combining multiple large data-sets to create an accurate representation of a novel, multi-layered, hybrid real-world maps. The hybrid map generation combines LiDAR, Ordnance Survey, and OpenStreetMap data to generate 3D cities spanning 1km^2. Evaluation of initial visualized scenes is presented. Initial tests consists of a 1km2 landscape map containing up to 16million vertices' and runs at an optimal 51.66 frames per-second.

Keywords: Crisis Management; Geovisualisation; Open Data and Volunteered Geographical Information (VGI); Location Technologies; Games Technology; LiDAR; Open Street Map; Ordnance Survey; Hybrid Map Generation; Map Error Reduction; Decision Support System

1. Introduction

Creating detailed and realistic 3D virtual environments for real-world maps for crisis management applications is increasingly difficult due to the variety, complexity, and large scale GIS data-sets. Modern GIS visualization applications focus on a limited number of data-sets [1]–[3]. Combining multiple data-sets can alleviate errors, produce unmapped areas, and increase accuracy of a map for terraforming of extruded models in runtime applications.

The creation of realistic virtual scenes at crisis time gives crisis managers the ability to plan specific emergency procedures: evacuations, fire extinguishing, flood prevention etc. If errors, or inaccurate data is visualized, the cascading effects could cause vast upset and potential loss of life.

Map data including critical infrastructures' are highly complex and include very large sets of data. For example, in the USA alone there are 560104 infrastructures. In 2005, there were: 2 million miles of pipeline, 2.800 power plants (with 300,000 production sites providing assets), 104 nuclear power plants, 80,000 dams, 60,000 chemical plants, 87,000 food-processing plants, 28.600 networked Federal

Deposit Insurance Corporation institutions, and 1,600 water-treatment plants [4]. If accidental, natural, or man-made upset happened to a single one of these infrastructures then this could have vast consequences and cascading effects to many other infrastructures of the USA. If crisis managers produce evacuation plans for specific infrastructures, the need for accurate, high resolution maps to create accurate recovery plans are needed. For instance, top-down 2D map representations do not give a fully immersive experience into the best locations for emergency vehicles to be placed. High resolution maps containing height points separated by small cell values give improved spatial awareness which is critical for the placing and maneuvering of emergency vehicles.

In addition to map complexity, the map data is not always kept up to date timely. For example, Googles' map system can be up to 3 years out of date for highly populated areas, and further out of date for rural areas. Although the updating of commercial map data can be helped by using crowd sourced data, as used with OpenStreetMap.org, this is not yet a common practice.

Current solutions provide limited user interaction, constricting a user's ability to fully understand a scene. The use of modern commercial game engines can provide cheaper, or free, alternatives to custom bespoke pieces of software. Research in user experience shows that immersion into simulations is broken, when inaccuracies, such as visible unsmooth level of detail (LOD) morphing occurs [5]. Robertson et al. state that a common criticism of immersion is the lack of peripheral vision due to limited screen size [5]. Game engines can allow advanced immersion into 3D scenes with use of: common rendering algorithms, culling techniques to produce large scale scenes running in real-time, animation systems which can be built upon, and advanced mathematics modules used for 3D world navigation and 3D camera creation. Advanced camera systems are capable of real-time changeable perspectives and projections to aid a user's interaction and understanding of a scene; perspective and orthographic projection. Perspective projection shows depth, orthographic [6] [7]. Within this paper, we discuss the procedures and techniques needed to create novel hybrid maps, for 3D terrain and scene generation, from the use of big data fusion from several GIS data sources. Light Detection and Ranging (LiDAR)[8]–[10], OpenStreetMap (OSM) [11], and Ordnance Survey (OS) [12] data are combined to reduce error and create increased accurate layered maps for use within a modern novel disaster support system using 3D environments. The layered hierarchy of maps allow a user to view the information they want to see; static or dynamic artefacts, highly detailed terrains, and procedurally generated content (PCG)[13]–[16], and specific infrastructure buildings. Use of filtering and interpolation algorithms are used to correct errors and generate missing map data. See [17] for early work into interpolation techniques for derivative of digital elevation models. For our research we will focus on the use and implementation of Catmull-rom interpolation [18][19].

The research focus is on map data generation, specifically the processes of combining maps in a variety of combinations, and the interpolation of data points.

The case study will test the improved high resolution map data within a game engine to decide the best resolution map density needed to run at a minimum 30 frames per-second.

The paper is structured as follows. Section 2 covers background work consisting of crisis management, related work and open data. Section 3 covers our visualization framework and the techniques and procedures for the generation and combination of multiple map data-sets. Section 4 discusses evaluations of the framework. Section 5 state the future work. Section 6 concludes our discussion and paper.

2. Background

Within this section we will cover background research into crisis management and current state of the art visualisation tools and techniques focusing on terrain and map generation.

2.1. Crisis Management

Interdependencies between infrastructures and unforeseen non-natural disasters can bring cities to a standstill [20]. A clear demand for proactive and reactive management strategies and systems for severe critical infrastructure (CI) disasters are needed [21]. Currently, limited response and effectiveness of plans and systems are hindered by: large number and diversity of stakeholders; conflicting demands and resources; and the lack of cooperation between different stakeholders, despite the best efforts of communities and governments [20] [21].
When a crisis incident occurs, response is a cyclic/iterative dynamic procedure. See Figure 1 for incident cycle.

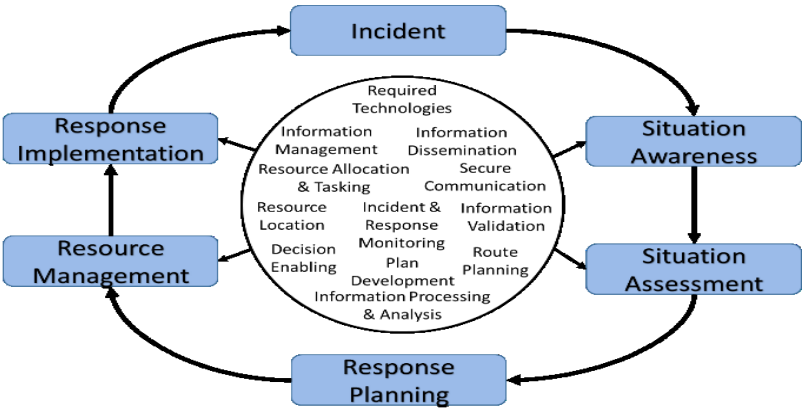

Figure 1 Incident and crisis management responce cycle

Minimizing decision time at crisis situation saves lives, resources, and costs. Fully understanding a large local area is time consuming. Many applications are available for the viewing of real-world scenes at crisis and non-crisis time. For example, Aqil et al. present a tool for real-time flood monitoring and simulation [22]. The application is limited to a top-down 2D representation of large areas giving little perspective to a scene/terrain. Simplistic polygon representations are used for flooded areas. Generation of realistic height terrains and 3D virtual cameras navigating scenes will produce a fuller understanding of potential flood plains, or potential flood paths; functionality Aqil et al. tool does not have.
We have observed from [23] that there are four types of interdependencies between infrastructures: Physical, Cyber, Geographical, and Logical. Within our research, we are focusing on the physical and geographical interdependencies due to their relevance to visualization, and their impact on the constraints they inhibit on potential crisis management plans. For example, the physical dependencies between a coal mine and a power plant connected by roads used for delivery are put at risk if flooding of said roads prevents delivery.
To prevent this situation, crisis managers and city planners can generate virtual scenes within our visualization framework, to simulate this situation. If map data is initially wrong, i.e. contains missing or inaccurate/outdated data, then user generated plans can be wrong from the beginning.

As highlighted, high complexity, inaccuracies, and incompleteness of map data; and lack of appropriate visualization solutions, are all the challenges we meet in dealing with 3D rendering of large map data for crisis management.
To solve some of these problems and provide a more fitting solution, we put forward procedures which will alleviate these issues by use of hybrid GIS data combination techniques and game technology visualization techniques.

2.2. Related Work

In this section, we discuss the current state of the art in visualization tools and techniques for geographical information.
Sketchaworld [24] is a tool which converts simplified sketches of landscapes, and generates full 2D and 3D procedural environments able to tackle rivers and bridges. The tool can generalize conflicting user inputs and create satisfactory scenes. This work does not include real world data, thusly the maps in question are user generated and subject to unrealistic terrain generation. Using real-world data with their procedural techniques will provide greater realism for future city planners. We would like to use this software as a benchmark for ideas into the interactivity and design procedures which may be relevant to users in a decision support system for crisis management. For example, a user could input a shorter path between destinations for use in pathfinding algorithms [25]–[29], such as through parks or open spaces which vehicles are normally prohibited to travel, and see the effects it might have on the planning algorithms used for automated routing.
Tanagra [30] is an editing tool which uses a mixed initiative paradigm to create a tool which helps the creation of 2D platform levels. Objects are placed within the world, such as obstacles and platforms which the player must navigate to get to the end of the level. They state current techniques in procedural level generation tend to offer little author guidance and suggest other techniques allowing them to tweak parameters and variables, which can be "*unintuitive, with small shifts leading to radical changes in the produced content*". Mixed initiative is the term where user and algorithm work together to create content; the algorithm suggests the next position of objects in the world, and in some cases overrides the user's decision because of unreachable desires. Using a mixed initiative approach provides access to large numbers of varying map types. A procedural autonomous map generator can ask a user to resolve conflicts of data within a scene. For example, if one data-set states land height should be above sea level, and another states below sea-level, the user can be queried.
Falconer et al. [31] created a prototype simulation and visualization tool, named '*SAVE'*, which allows stakeholders to interact with a virtual environment composed of real-world data combing computer game technology with computer modelling techniques. Their work has been used to model effect and cause of future concepts and plans for the sustainability of assets within the environment e.g. buildings, population, economy, etc. Their application colors assets which the stakeholder is interested in with distinctive colors and patterns representing high or low results for the effect they want to observe; sustainability, cost and safety comparisons, among others. Their tool needs a large number of user generated variables representing their personal interest of assets. We want to use similar visualization techniques to view scenes with multiple overlays, highlighting crisis situations, infrastructure critical buildings, and potential crisis zones.
A real-time procedurally generated, GPU only, application for the visualization and interaction of real-time accurate fluid simulator has been created by Kellomäki [32]. We believe we can implement this

purely GPU driven fluid simulator in our simulation and accurately model many scenarios involving water, with tremendous realism. The simulation can also interact with a physics engine to move objects within in a scene, simulating the damage a flood can create. The algorithm can also take into account rainfall data which is specifically interesting to our work. Modelling the rainfall predicted by the METoffice, can simulate flash floods or even dam water levels and predict how long it will be before flooding commences. To observe accurate simulation of fluids within a scene, accurate terraforming of real-world map data is needed, otherwise, the fluid simulator will guide the resource in the wrong direction, with potential of huge consequences.

Autonomous Learning Agents for Decentralized Data and Information Network (*ALADDIN*), a research project to investigate multi-agent systems and data fusion, decision making, resource allocation, and other domains. The project consists of simple visualizations but complex planning procedures. An issue with this work is the limited visualization of 2D scenes.

The *SAVE* project and *ALADDIN* project are great examples which we can be built upon with the use of 3D game technologies.

Visualization is not the only problem with creating a coherent modern decision support system for crisis management. Correct and accurate data is key for successful planning for possible future issues.

2.3. Digital Map Open Data

Open data is freely available, often through internet portals, provided by companies and government sectors. Careful consideration must be given when working with open-data due to inconsistencies and quality assurance and amount of multiple data types and file formats.

Many projects have incorporated open data for improved visualization and engagement: *Open Data Monopoly*; *Bar Chart Ball* [33]; *Open Trumps*; *Flight Leader*; *Urbanopoly*, *MuseumVILLE*, and *Open Street Racer*. For example, the *Open Street Racer* uses OSM data to generate virtual racing tracks through procedurally generated urban environments, translating buildings which interrupt the track playability [34]. Figure 2 shows *Open Street Racer*.

Combination of open data and commercial data can alleviate errors within single maps by querying each map to detect anomalies.

We propose to use a mixed initiative approach to content generation: procedurally generating landscapes and building models at multiple LOD at run-time will allow generalized map generation of many selected areas, and premade models of assets, which can be tailored offline and loaded when necessary. The use of LiDAR and OSM data will be the most relevant and accurate data sets for this type of content creation.

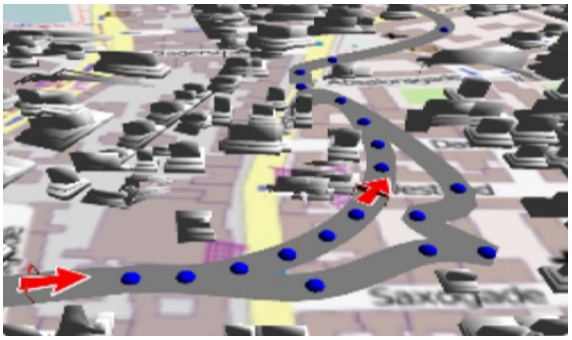

Figure 2 OpenStreetRacer procedural city and race track generated from OSM data [34]

3. Visualization Framework

This section we present our framework *Project Vision Support* (PVS), a multi-agent system which combines an advanced game engine and planning algorithms to manipulate complex intertwining sets of big-data extracted from real-world map instances, to aid the action planning of emergency services and response management during crisis events.

PVS is a C# framework built on top of a modern game engine [35]. Microsoft XNA has been used on many project to varying success [6], [36]–[39]. We have changed from Microsoft XNA rendering engine to Monogame rendering engine [40]. The Monogame engine builds on top of XNA and uses the same namespaces so change over introduces a minimal number of problems. Monogame removes restrictions of XNA and allows users to program much lower level code allowing custom memory management which is key for high impact visualization techniques for high density model meshes. The framework includes many separate components as depicted in Figure 3. The main '*engine*' consists of: animation classes, custom 2D and 3D camera classes, debugging classes, primitive geometry classes, input systems, advanced shader classes used for advanced rendering, and advanced screen system. The main project extends these basic objects for custom assets such as buildings or road networks.

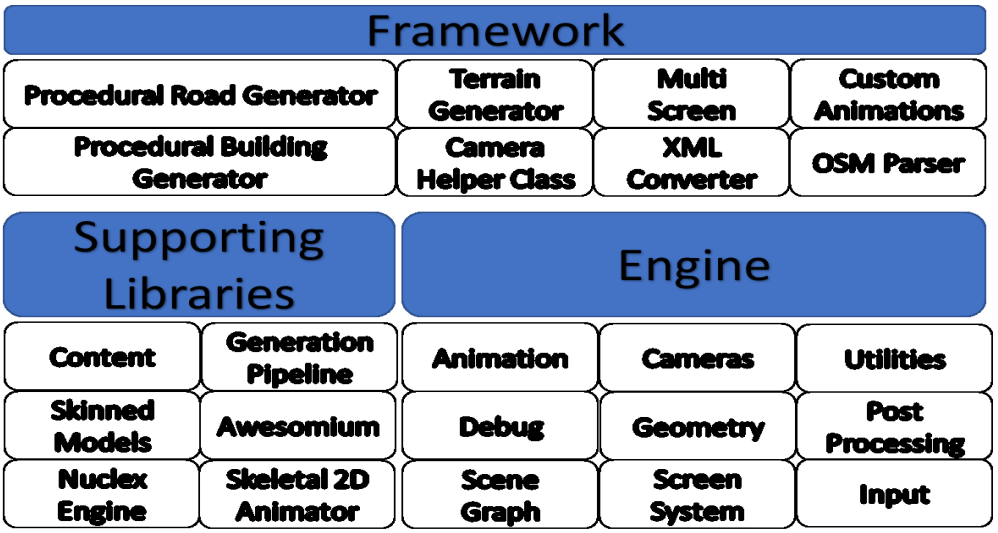

Figure 3 Project Vision Support Framework Components

PVS has the ability to convert the OSM data and generate 3D environments including minimal procedural buildings (walls and roof tops), road networks, amenities (user generated assets such as pharmacy's and parking areas), as well as boundary locations used for depicting emergency locations. The power of PVS is the ability to generate procedural scenes from any OSM map selected from the OSM webpage. See Figure 4 to see an 800m^2 visualization of an area of New York Manhattan. Figure 2 shows *Open Street Racer* [34]. *Open Street Racer* generates procedural tracks but relocates generated buildings which impact the track, producing inaccurate visualizations. Generated OSM buildings and road and rail networks are visualized to the unaltered OSM specifications in Figure 4. Object pooling allows the visualization of large cities within the UK currently tested up to 115km^2 running above 50 frames per second (FPS).

Figure 4 PVS procedural city generation of New York Manhattan containing buildings, roads, and rail networks

In the following sections, we will focus on map data generation procedures and algorithms, specifically the data complexity, the generation of missing data, errors in data and related issues.

3.1. Map Data Generator

Generation of 3D terrains within virtual environments is a common technique in modern computer games. Generating real-world, accurate, terrains representing 1 kilometer square at multiple resolutions adds additional problems. Increased computation and memory consumption at runtime, and storage limits are issues which need to be overcome.

Terrain meshes generated in games are traditionally created in square grids, with vertex points separated in equal segments, called *cells*. A map of 500^2 points at 2metre cells representing $1km^2$, will use 250,000 vertex points, containing data specific for each application: position in a 3D world, 1 or more texture coordinates, and quite often a facing vector direction for use with advanced lighting systems. Terrain generated at this resolution using real-world recordings, with cell sizes representing 2 meters apart, has the potential to miss vital information such as walls, trees, bollards, small buildings, and telephone boxes. Producing higher resolution maps by combining multiple GIS data-sets can reduce this error.

Ordnance Survey (OS) Data is terrain height maps represented by points separated by 50 meters covering the whole UK. This distance of $50m^2$ is a generalization of the terrain heights above sea level recorded using Ordnance Datum Newlyn. Each map file covers $10km^2$ containing 200^2 points. A subset of $1km^2$ map will contain only 20^2 points from the larger OS map. The potential for missed artefacts within a scene is vast. Although the data is only a generalization, improvements can be made with the combination of alternative GIS data-sets such as OSM and LiDAR data.

LiDAR is a mapping technique capable of recording highly accurate spatial positions on a surface. The LiDAR data obtained was captured by flying a small aircraft over selected areas, commonly at 800 - 1000 feet. Because of the high cost of flying an aircraft, much of the UK is unmapped. Where mapping has taken place, multiple resolutions are normally available. Figure 5 shows the areas of UK which has been mapped. Image A represents the areas of the UK which have been mapped to 2m resolution, image B - 1m resolution, image C - 0.5m resolution, and image D - 0.25m resolution. See Table 1 to compare LiDAR resolution with area coverage and point density. Light pulses are shot towards the surface of the Earth, and sensors detect how long the light signal takes to bounce of the surface, back to the aircraft. This particular data is accurate to within 5-15cm's. The data provided is split into two types of map: Digital Terrain Model (DTM) and Digital Surface Model (DSM).

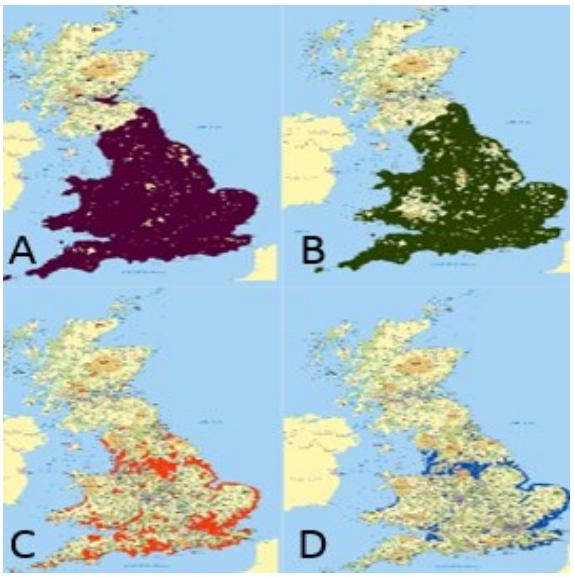

Figure 5 LiDAR coverage of the UK

Table 1 LiDAR data comparison

LiDAR Map	Area	Points Contained
2m	$1km^2$	$500^2 = 250,000$
1m	$1km^2$	$1000^2 = 1,000,000$
0.5m	$500m^2$	$1000^2 = 1,000,000$
0.25m	$500m^2$	$2000^2 = 4,000,000$
Proposed 0.25m	$1km^2$	$4000^2 = 16,000,000$

A DTM map is pre-processed interpolated data-set representing only the terrain height, removing all objects from a scene: buildings, trees, cars etc. interpolation only occurs when small areas of data is missing and is done by the company from whom we obtained the data from *Geomatics-Group* [8]. Figure 6 shows the difference between DSM and DTM maps. White space represents missing data. The top images are DSMs, the bottom images are DTMs. The two right bottom images show large areas of missing data after processing.

The DSM map is un-altered surface heights of a scene containing buildings, walls, trees, power-lines, and also flocks of birds which can input huge errors within a map. When visualized, the huge spike of height data can be interpreted as a skyscraper or monument, which can hinder potential routes for emergency vehicles. This issue can be removed by combining the multiple maps of DTM, DSM, and OSM map data. Another issue which can be resolved by hybrid map combination techniques is the error of parked cars represeted by increaed height data within the LiDAR maps. This can be done by simply querying every point within a map to see if it lies on a road, if so, then remove this data point and interpolate using the surrounding data points.

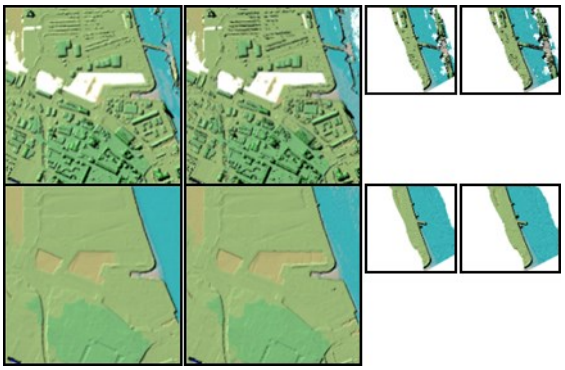

Figure 6 LiDAR DSM (top) and DTM (bottom) comparison, left to right is 2m, 1m, 0.5m, 0.25m resolution map images

OSM is a worldwide open data mapping project using over 1.7 million volunteers to map local environments. The Topological Integrated Geographic Encoding and Referencing system (TIGER) which includes road data for almost all of the USA has been donated to OSM. OSM has four main elements: *Node* (a latitude and longitude location on the Earth's surface); *Way* (a group of ordered *Nodes*. If the first *Node* is the same as the last *Node* in the ordered list then this denotes a *boundary*, else it is a *Highway*.

The direction of the *Highway* is the order of the *Nodes* in the list); *Relations*; and *Tag* (extra information describing the *Node* or *Way*, for example *Tag=Building*) which is a key/value pair. Currently OSM map data contains: 52,971 different key items. The complexity of deciphering the massive data sets and intricate naming's for seemingly similar objects in the world is demanding and needs vast attention to detail to compare names and split corresponding data sets into more manageable groups of data. For example, the *Key:Building* has 7512 *Tags* to describe it, many of which are repeated but with slightly different string representations. Many fuzzy text matching algorithms have been created to fix this problem [41]. Another option would be manually selection of the most commonly used *Key* and *Tag* data. Within the database there are: 2.8 billion objects; 1.1 billion tags; 2.5 billion nodes of which 3.73% have at least one *Tag* description; 255 million *Ways*, half of which are closed (boundaries); and 2.8 million *Relations*. A planets worth of data is 498GB. Pre-processing and separating this big-data set can greatly reduce storage space needed, i.e., pre-process OSM maps for specific locations, and scenarios. An example of a specific scenario could be to generate a map containing water-ways and water boundaries to monitor potential flood alerts.

Generating files containing specific data can greatly improve processing time. Having separate files containing single objects: buildings, roads networks, path networks etc. can improve processing of this data, and convert complex highly dense data-sets into manageable components for use to build up layers of a scene. Layers for which a crisis manager can pick to build up their own representation of a real-world scene.

Parsing unprocessed files can take a considerable amount of time, which can be better spent else-where. Generating run-time objects and 3D meshes takes much longer, see Table 2.

The process of converting the node locations to game world locations is a brute force process, and does not exclude nodes which are not used. Obviously this can be improved as stated above.

Other errors contained in the LiDAR data is dynamic objects; mobile vehicles which are picked up at the point of recording. This has potential to create spikes in height data on surfaces which should be flat such as roads. These errors can also dictate the direction of potential flood paths.

Using a combination of OS, DSM, DTM, and OSM can create custom layered accurate maps, with reduced errors and removed dynamic objects within scenes.

Table 2 OSM file parsing and generation for buildings within Manchester UK. Area coverage 115.6km^2

Process	Time
Serialisation of OSM map file	5977 milliseconds
Converting node longitude and latitude to X,Y,Z position	3 minutes, 38 seconds, 287 milliseconds
Extracting and generating 3D building meshes	1 minute, 48 seconds, 228 milliseconds.

3.2. High Resolution Map Generation

To create high resolution maps, we propose to interpolate missing map data from the lowest resolution, to the highest resolution: OS 50m to LiDAR 0.25m using Catmull-Rom interpolation. Catmull-Rom has the benefit of interpolating through control points, and accurate starting tangents by the use of additional control points either side of the interpolation control points. This is important for accurate map generation, especially when interpolating from the 50m resolution points of OS data to the 2m LiDAR data due to the large difference in distance between them.

Before we discuss the process of minimizing errors by the combination of map data, an issue became apparent at the beginning of the terrain generation process.

However, when generating virtual terrain from map files, the polygons are generated from the outside vertex's inwards, which is not a problem when only a single map is visualized. When multiple neighboring maps are needed, a visible skirt of missing polygons appears between neighboring maps. See Figure 7. A technique proposed by [42] allows smoother LOD transitioning between levels of resolution, and removes the need for stitching meshes. LOD popping is a problem when simulating accurate realism to the user. An alternative to this is to duplicate data from surrounding maps, as represented by the red points and single purple point from Figure 7. This duplication of data can dramatically increase the need for large storage devices, as well as memory at runtime simulation.

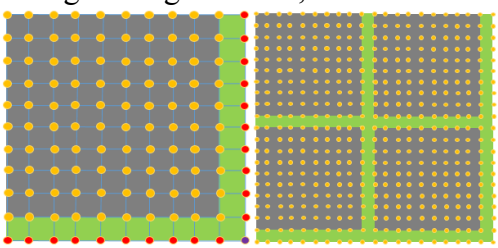

Figure 7 Skirt of missing data between maps

With using an optimized binary file format for the generated maps for the complete UK, we estimate a saving of 29.23% which is equal to 13.22 Terabytes. With added duplicated data, this an extra 16.384 Gigabytes of storage space which is needed. This is only for the highest resolution of 4000^2 points at 0.25m cells covering the same coverage of as OS covers and not taking into account other map resolutions. The generation of the binary format file is out of scope for this article.

Another issue with the generation of complete maps is the differences in the LiDAR data available. Figure 8 shows the differences between the maps. White space represents missing data. Inserting data

from the DTM map into the missing data points of the DSM will not generate complete maps due to DTM maps having large areas of missing data. Figure 8 also shows maps generated by combining DSM and DTM data sets.

Process 1 is the combination of DSM and DTM map to exchange data from the DTM to the DSM. If data is still missing, process 2 interpolates lowers resolution map points and combines it with the map from process 1. Process 3 uses map data from OSM to improve height levels by querying OSM boundaries such as water boundaries. Process 4 uses OSM data for the removal of dynamic object heights such as cars on roads. An issue with process 1, working with DTM maps is the pre-removal of information. The DSM map in process 1 shows a ship and slipway. The DTM map in process 1 has this data removed. The issue is not the removal of the ship but the slipway. This is an issue for creating accurate paths for water vehicles but can be applied to other dynamic artefacts within an urban environment.

Each of these processes have been simulated individually, but have not been combined into a full automated or partial automated programmatic procedure. This is a potential issue with the limit of testing into all possible combinations of maps and OSM data boundaries. Future work will improve this issue.

To generate the missing data, interpolation from lower resolution maps is needed. The Catmull-rom interpolation algorithm uses 4 points; points 1 and 4 are tangent points used to get the tangent angle entering and leaving points 2 and 3 respectively. It is between points 2 and 3 where interpolation occurs.

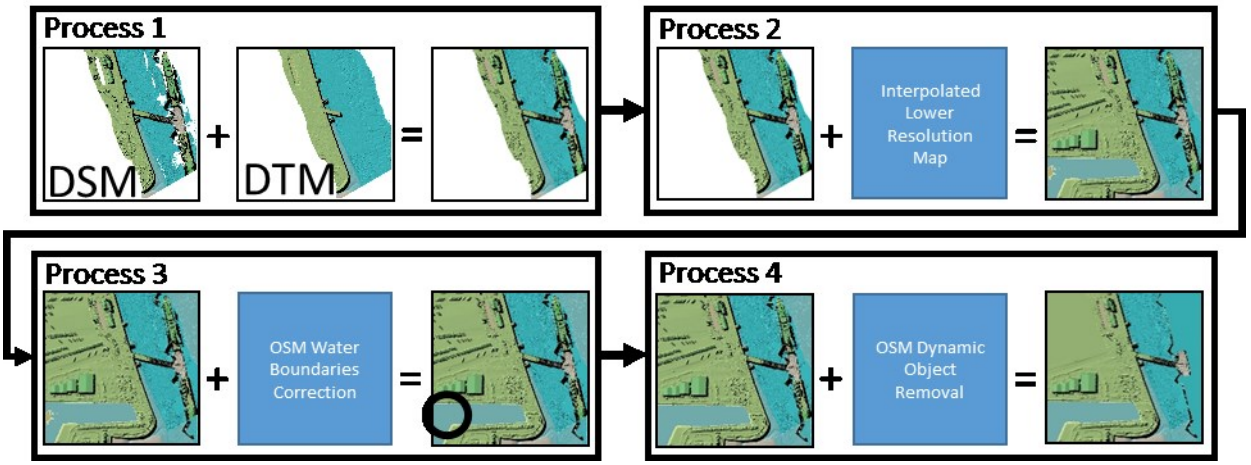

Figure 8 Hybrid map combination for improved map data.

To generate missing map data, the left, top, right and bottom tangent interpolation points are needed. The algorithm procedure runs as: combine 2m DSM and DTM map together, if the map is not complete then generate missing data points from surrounding maps of the same resolution or lower. If a map is lower resolution then multiple interpolation stages are processed.

To generate the left tangent interpolation points, there is a possible 10 combinations of maps, top interpolation tangent points have 5 possible combinations, right interpolation tangent points has 5 possible combinations, and bottom interpolation tangent points has 10 possible combinations. The combination consists of the current resolution map we are generating the tangent points at, and the lower resolution map, i.e. the map we want to generate has 8 surrounding maps. Think of it as a 3 by 3 matrix with the center index being the map we want generate. To generate the left tangent points, the combination consists of top left map, top map, left map, bottom left map, and the bottom map. Figure 9

shows the combinations needed for the generation of the left tangent points. Red represents lower resolution maps, blue represents the next higher resolution map, and green represents the missing skirt of data between the maps. Notice there are larger skirts on the lower resolution maps.

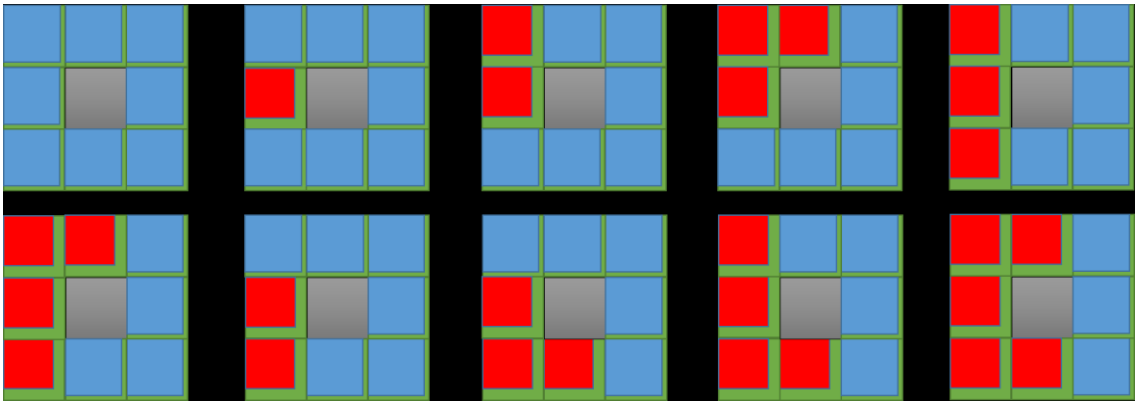

Figure 9 Map combination for the left tangent points extraction.

Figure 10 depicts the iterative interpolation technique used to generate values between control points. TL can be thought of as the left tangent point, then being interpolated horizontally through a single row of data from the 2D map to the final right tangent points.

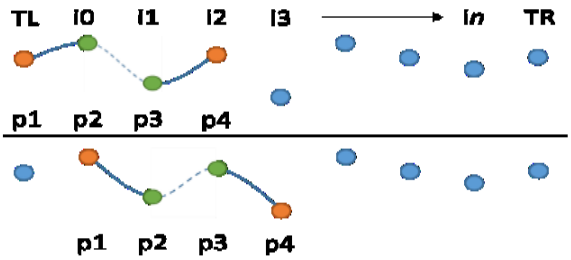

Figure 10 Catmull-rom interpolation. 'TL' and 'TR' represent the tangent points generated from surrounding maps. The 'i' represents index position.

If a neighbor map does not exist then a lower resolution map will be used. If a lower resolution map does not exist, i.e. the map is an edge map of the UK, then tangent interpolation points will be taken from the lower resolution map we are trying to create, i.e. the green map in Figure 11.

Figure 11 shows the procedure needed to generate the left tangent interpolation points for the interpolation of the missing map, depicted as green in Figure 11. As shown, the surrounding maps are all lower resolution. In this case, multiple interpolations using adjacent data points between maps are needed. Points 1, 2, 3, and 4 are needed to create point 5. Each point is generated through interpolation of adjacent points from multiple maps.

The first data value loaded from the file is the top left point of the map. This is contradictive to how OS data is explained. OS map naming and referencing are taken from the bottom left, noted as easting and northing, but the OS data within the map can be thought of as upside down, the first value loaded corresponds to the top left point of the map. This is a confusing state of affairs, especially when visualising this data with polygon creation within a game engine to view on screen.

To generate *point 1*, the lower left map supplies the index (*width.max – 1, height.index 1*) data point, and index (*width.max – 1, height.index 0*) data point depicted as *green points*, and the index (width.max – 1, *height.index* 0) data point, and index (*width.max - 1*, *height.index* 1) depicted as purple points. *Width.max* represents the total width of the map, and *height.max* represents the total height of the map. *Width.index* and *height.index* represent the *column* and *row* index respectively. These four

points are used as control points and interpolated to create point 1. This process is repeated to create points 2, 3, 4, 7 and 8. Point 5 is generated by points 1, 2, 3, 4 being the control points for interpolation horizontally.

The next process is to interpolate between the left map and the green map horizontally, the same as was done with points 1, 2, 3, and 4, to make point 5 to generate the majority of tangent points. This is depicted by the curve gold arrows and the black dashed arrow line.

With point data generated between the left and center map, vertical interpolation of the tangent points produces the final array of tangents used in a later process. Point 5 is the tangent point, and point 6 is the control point for the first iteration of the interpolation process. Figure 10 depicts this situation in the top of the image.

Edge and corner maps of the UK are special cases because the tangent points needed cannot be generated due to lack of neighboring maps. These specific maps are often in the middle of the surrounding oceans of the UK, and need custom procedures. Because of the locations we will not focus map generation techniques for these areas. See Table 3 which shows the number of maps with 1 or more neighboring maps missing for the 2m and 1m LiDAR maps, and also the OS maps after being split into 1km map segments. The 0.5m and 0.25 m LiDAR maps cover 500m^2, so the left, right, top and bottom missing maps are doubled.

With all surrounding interpolation points generated, interpolation vertically occurs, then horizontally. If a map has data, the interpolation algorithm uses these as control points.

With missing data being created, removing objects from the DSM can be completed by utilizing OSM.

The novelty of this paper is the use of OSM building boundary data for the extraction of building meshes from the DSM data. This data can be placed within the DTM data to create a map with reduced dynamic artefacts; cars, flocks of birds, other movable objects etc. OSM also has data locations of trees, bollards, walls and many others which can be used for the extraction of assets needed for accurate planning. This hybrid combination has the ability to get accurate heights of static artefact: buildings, trees, pylons, which can be used for reduced mesh models for visualization at runtime.

LiDAR has trouble penetrating waters. OSM brings improvements to LiDAR map generation by the use of water boundaries. Figure 8 shows that combining DSM and DTM maps may remove missing data, but that data may be false positive. Using water boundaries of OSM, these errors in heights can be lowered a specified height value as shown in Figure 8 procedure 3: *OSM water boundaries correction*. Deciding the height of water levels is complicated due to water ways and rivers may be higher or lower than the sea level state by Ordnance Datum Newlyn.

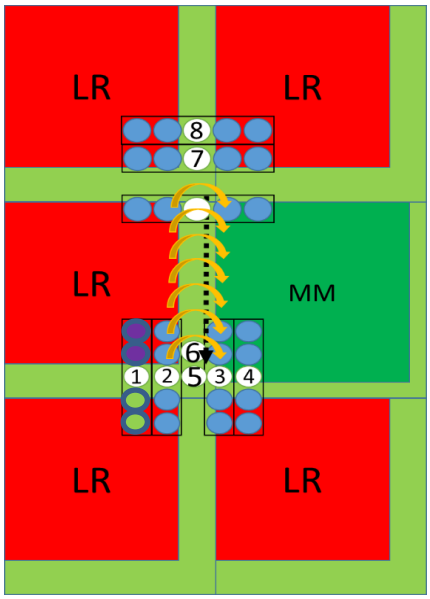

Figure 11 Surrounding maps needed for missing map interpolation for higher resolution maps. Left tangent line depicted.

Within this section we have explained the technique and benefits that a combination of multiple GIS data-sets can bring to hybrid map generation, by reducing errors and removing dynamic objects, while keeping needed data for accurate crisis management planning decisions including for example, planning evacuation routes for crisis events, and proposing emergency developments and resources.

Table 3 Showing the number of maps which don't have certain neighbor maps, for 2m and 1m LiDAR maps covering 1km2 and 1km2 subsections of OS maps.

Maps with missing neighbors to the:	**Count**
Left	1300
Right	1300
Top	1100
Bottom	1100
Top-left only	5
Top-right only	4
Bottom-left only	2
Bottom-right only	3
Top Left, Left, Bottom Left, Bottom, Bottom Right	3
Top Right, Right, Bottom Right, Bottom, Bottom Left	4
Bottom Left, Left, Top Left, Top, Top Right	6
Bottom Right, Right, Top Right, Top, Top Left	5

4. Evaluation

In this section we report our experiments to evaluate the performance of our system and techniques to manage complex map data, correct errors and missing data, and identity the best trade-off between maps/terrains resolution and performance. We have used the map densities stated in Table 4 on a desktop computer: Intel Core i7-3770 @ 3.40GHz, 16.0 GB RAM, 64-bit Operating System, x64-

based processor, Windows 8.1 Pro, NVIDIA GeForce GTX 650 Ti graphics card. As stated, the framework is built on top of a game engine. We have used Monogame for this experiment due to the limitless size of vertex buffers, unlike the previous framework which utilized Microsoft XNA rendering engine. C# does not allow arrays to be over 2GB, thusly, additional configuration files must be added to allow very large objects. We have tested a LiDAR map file covering $1km^2$ containing 1000^2 height points of Liverpool UK dock lands, producing accurate and realist results running at 751 frames per-second. We have extrapolated and tested the densities directly corresponding to measurements for the testing of maximum potential frame rate to area coverage trade off.

Initial test of the $4*4000^2$ representing $2km^2$ area map at 0.25m resolution within the framework with, no LOD techniques applied, uses 3.42676 GB of memory.

We can see that the technique for generating large and complex maps has a good performance and is able to process with an acceptable frame rate and rendering time, up to 16 million polygons for a map dimension of 4000^2 at 0.25m.

The experiments results looks promising, but further analysis has to be carried out into the best situation for the splitting of maps. The splitting of a 4000^2 map at 0.25m resolution to a 4 neighboring 2000^2 maps at 0.25m resolution contains 4 vertex buffers to cover $1km^2$, which increases the framerate by 0.33fps. The GPU draw time is reduced from 0.23ms, to 0.19ms. A question is raised whether the speed increase is due to bus-speed transfer from CPU to GPU, or is it the reduction of polygons the GPU has to rasterize. The splitting of a map adds the issues of missing skirt information as stated earlier, removing 47,982 polygons from the scene.

Table 4 Runtime evaluation of visualized terrains at multiple resolutions. The data is captured while the 3D camera is pointing to the center of the scene, containing all terrain polygons.

Map Dimensions: Points within map at meter cell size	**Vertex Count**	**Index Count**	**Polygon Count**	**FPS**	**Draw MS**	**GPU MS**	**Vertex Buffer Count**
500^2 points at 2m $1km^2$	1,494,006	498,002	250,000	2,153.98	0.08	0.33	1
1000^2 points at 1m $1km^2$	5,988,006	1,996,002	1,000,000	751.15	0.18	0.24	1
1000^2 points at 0.5m $500m^2$	5,988,006	1,996,002	1,000,000	764.99	0.08	1.76	1
2000^2 points at 0.25m $500m^2$	23,976,006	7,992,002	4,000,000	203.73	0.17	4.56	1
$4*2000^2$ points at 0.25m $1km^2$	95,904,024	31,968,008	16,000,000	52.04	0.19	18.76	4
4000^2 points at 0.25m $1km^2$	95,952,006	319,834,002	16,000,000	51.66	0.23	20.38	1
$4*4000^2$ points at 0.25m $4km^2$	383,808,024	127,936,008	64,000,000	4.68	0.27	211.99	4

5. Conclusion

In this paper we have discussed the use of Games Technologies for the research and development of 3D virtual environments representing real environments captured from GIS information and Open source map data. The challenges involved in these research areas concern the large amount of data to be dealt with, as well as some map data which include errors and are not complete. The domain of application of our work is crisis management which requires very accurate GIS or map information, as well as data that is manageable and user friendly. We believe the use of creating 3D virtual environments using real map data whilst correcting and completing the missing data will improve the quality and performance of crisis management decision support system and provide a more natural and intuitive interface for crisis managers.

We have proposed and tested procedures for improved map generation techniques combining real-world maps using LiDAR, OSM, and OS data for terrain generation. We have also stated improvements in storage space by saving maps in an optimized file format. Evaluations of initial tests stress testing a C# game engines rendering capabilities and concluded that with additional advanced rendering techniques, large, highly dense, and accurate map visualizations over multiple kilometers is achievable at a comfortable frame-rate.

The differentiation of multiple GIS data-sets and programming of map processors has been more complex than first thought. To complete our framework, we must complete future work. The generation algorithm for hybrid map generation needs to be tested and completed to allow additional data to be combined, as well as increased OSM integration. For example, using the road system of OSM to smooth out dynamic objects on the LiDAR maps where roads will be placed.

LOD techniques need to be added to the framework to allow larger terrain coverage. Once larger terrains are visible on screen, and map data is complete, i.e. contains no missing data but can still be error prone, then automated planning algorithms can be tested within the framework while viewing the results in real-time.

6. Future Work

Our future work consists of completing and programming a semi-automatic procedure which can combine the multiple map data sets into a hybrid layered representation capable of querying a user to supersede its own decisions.

Our future work also consists the comparison of different interpolation techniques for the interpolation of lower resolution maps to a higher resolution to fix error prone maps such as the LiDAR data.

The need to implement flooding visualization techniques other than using a single plane representing water level is needed to combine the enriched hybrid maps to monitor flood pooling to create future artificial intelligent path finding world representations custom for individual agents.

Acknowledgments

We would like to thank Geomatics-Group [8] for supplying the LiDAR data. We would also like to thank the UK government for making the OS data freely available through easy web portals [12], and finally we would like to thank OSM [11] for the data provided through their web portal and supporting partners.

References

[1] P. Folger, “Geospatial information and geographic information systems (GIS): Current issues and future challenges,” *Congr. Res. Serv.*, pp. 1–34, 2009.

[2] “What is GIS.” [Online]. Available: http://www.esri.com/what-is-gis. [Accessed: 30-Mar-2015].

[3] P. Folger, “Geospatial Information and Geographic Information Systems (GIS): An Overview for Congress,” 2011.

[4] A. Miller, “Trends in process control systems security,” *Distrib. Syst. Online, Secur. Privacy, IEEE*, pp. 57–60, 2005.

[5] M. Robertson, George; Czerwinski, Mary; Van Dantzich, “Immersion in desktop virtual reality,” *Proc. 10th Annu. ACM Symp. User interface Softw. Technol.*, pp. 11 – 19, 1997.

[6] R. P. Mihail, J. Goldsmith, N. Jacobs, and J. W. Jaromczyk, “Teaching graphics for games using Microsoft XNA,” *Proc. CGAMES 2013 USA - 18th Int. Conf. Comput. Games AI, Animat. Mobile, Interact. Multimedia, Educ. Serious Games*, pp. 36–40, 2013.

[7] P. Bourke, “Low cost projection environment for immersive gaming,” *J. Multimed.*, vol. 3, no. 1, pp. 41–46, 2008.

[8] “Geomatics Group: aerial LIDAR data, aerial photography and spatial data.” [Online]. Available: https://www.geomatics-group.co.uk/geocms/. [Accessed: 30-Mar-2015].

[9] L. Cheng, J. Gong, M. Li, and Y. Liu, “3D Building Model Reconstruction from Multi-view Aerial Imagery and Lidar Data,” *Photogramm. Eng. Remote Sens.*, vol. 77, no. 2, pp. 125–139, Feb. 2011.

[10] F. Leberl, A. Irschara, T. Pock, P. Meixner, M. Gruber, S. Scholz, and A. Wiechert, “Point Clouds Lidar versus 3D Vision,” *Photogramm. Eng. Remote Sens.*, vol. 76, no. 10, pp. 1123–1134, 2010.

[11] “OpenStreetMap.Org,” 2014. [Online]. Available: http://www.openstreetmap.org/#map=15/53.4312/-2.8737.

[12] “Britain’s mapping agency | Ordnance Survey.” [Online]. Available: http://www.ordnancesurvey.co.uk/. [Accessed: 30-Mar-2015].

[13] Y. I. H. Parish and P. Müller, “Procedural Modeling of Cities,” no. August, pp. 12–17, 2001.

[14] B. Bradley, “Towards the Procedural Generation of Urban Building Interiors Table of Contents,” no. August, pp. 1–42, 2005.

[15] P. Wonka, D. Aliaga, P. Müller, and C. Vanegas, “Modeling 3D Urban Spaces using Procedural and Simulation - based Techniques SIGGRAPH 2011 Course Proposal Organizers : About the Lecturers,” 2011.

[16] G. N. Yannakakis and J. Togelius, “Experience-Driven Procedural Content Generation,” *IEEE Trans. Affect. Comput.*, vol. 2, no. 3, pp. 147–161, Jul. 2011.

[17] V. Chaplot, F. Darboux, H. Bourennane, S. Leguédois, N. Silvera, and K. Phachomphon, “Accuracy of interpolation techniques for the derivation of digital elevation models in relation to landform types and data density,” *Geomorphology*, vol. 77, no. 1–2, pp. 126–141, 2006.

[18] C. Twigg, “Catmull-Rom splines,” *Computer (Long. Beach. Calif).*, pp. 4–6, 2003.

[19] S. R. Marschner and R. J. Lobb, “An evaluation of reconstruction filters for volume rendering,” *Proc. Vis. ’94*, 1994.

[20] D. Dudenhoeffer, S. Hartley, and M. Permann, “Critical Infrastructure Interdependency Modeling : A Survey of Critical Infrastructure Interdependency Modeling : A,” *INL is a U.S. Dep. Energy Natl. Lab. Oper. by Battelle Energy Alliance*, no. August, 2006.

[21] J. Hernantes, E. Rich, A. Laugé, L. Labaka, and J. M. Sarriegi, “Learning before the storm: Modeling multiple stakeholder activities in support of crisis management, a practical case,” *Technol. Forecast. Soc. Change*, vol. 80, no. 9, pp. 1742–1755, Nov. 2013.

[22] M. Aqil, I. Kita, A. Yano, and N. Soichi, “Decision Support System for Flood Crisis Management using Artificial Neural Network.,” *Int. J. Intell. Technol.*, vol. 1, no. 1, pp. 70–76, 2006.

[23] S. Rinaldi, “Modeling and simulating critical infrastructures and their interdependencies,” *Syst. Sci. 2004. Proc. 37th Annu. Hawaii Int. Conf.*, vol. 00, no. C, pp. 1–8, 2004.

[24] R. Smelik and T. Tutenel, “Interactive creation of virtual worlds using procedural sketching,” *Proc. Eurographics 2010*, pp. 1–4, 2010.

[25] X. Cui and H. Shi, “A * -based Pathfinding in Modern Computer Games,” *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 11, no. 1, pp. 125–130, 2011.

[26] P. Kumar, L. Bottaci, Q. Mehdi, N. Gough, and S. Natkin, “Efficient path finding for 2D games,” no. 1, 2004.

[27] Y. Björnsson and K. Halldórsson, “Improved Heuristics for Optimal Path-finding on Game Maps.,” *AIIDE*, 2006.

[28] R. Graham, H. McCabe, and S. Sheridan, “Pathfinding in computer games,” *ITB J.*, 2003.

[29] S. Kumari and N. Geethanjali, “A survey on shortest path routing algorithms for public transport travel,” *Glob. J. Comput. Sci. ...*, vol. 9, no. 5, pp. 73–76, 2010.

[30] G. Smith, J. Whitehead, and M. Mateas, “Tanagra: Reactive planning and constraint solving for mixed-initiative level design,” *Comput. Intell. AI Games*, vol. 3, no. 3, pp. 201–215, 2011.

[31] R. a. Falconer, J. Isaacs, D. J. Blackwood, and D. Gilmour, “Enhancing urban sustainability using 3D visualisation,” *Proc. ICE - Urban Des. Plan.*, vol. 164, no. 2002, Jun. 2011.

[32] T. Kellomäki, “Interaction with Dynamic Large Bodies in Efficient, Real-Time Water Simulation,” *J. WSCG*, vol. 21, no. 2, pp. 117–126, 2013.

[33] J. Togelius and M. G. Friberger, “Bar Chart Ball, a Data Game,” *Proc. 8th Int. Conf. Found. Digit. Games (FDG 2013) fdg2013.org*, pp. 451–452, 2013.

[34] M. G. Friberger, J. Togelius, A. B. Cardona, M. Ermacora, A. Mousten, and M. M. Jensen, “Data Games,” *Proc. Proced. Content Gener. Work. FDG, Found. Digit. Games*, 2013.

[35] D. Tully, A. El Rhalibi, M. Merabti, Y. Shen, and C. Carter, “Game Based Decision Support System and Visualisation for Crisis Management and Response,” in *The 15th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, 2014.

[36] T. B. Kvamme, “Evaluation and Extension of an XNA Game Library used in Software Architecture Projects,” no. June, 2008.

[37] R. Ouch and B. Rouse, “Developing a driving training game on Windows mobile phone using C# and XNA,” *Proc. CGAMES’2011 USA - 16th Int. Conf. Comput. Games AI, Animat. Mobile, Interact. Multimedia, Educ. Serious Games*, pp. 254–256, 2011.

[38] O. Denninger, “Game Programming and XNA in Software Engineering Education,” *Eng. Educ.*, no. April, 2008.

[39] E. W. Clua, P. a Pagliosa, A. Montenegro, and L. Murta, “A Fast and Safe Framework to Prototyping Physical Worlds Using XNA and GPU,” *Physics (College. Park. Md).*, pp. 8–11, 2009.

[40] “MonoGame | Write Once, Play Everywhere.” [Online]. Available: http://www.monogame.net/. [Accessed: 30-Mar-2015].

[41] P. Jokinen, J. Tarhio, and E. Ukkonen, “A comparison of approximate string matching algorithms,” *Software—Practice Exp.*, vol. 26, no. 12, pp. 1–4, 1996.

[42] F. Strugar, “Continuous distance-dependent level of detail for rendering heightmaps,” *J. Graph. GPU Game Tools*, no. July, pp. 1–15, 2009.