(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 September 2005 (29.09.2005)

PCT

(10) International Publication Number
WO 2005/091198 A1

(51) International Patent Classification[7]: G06F 19/00, 15/18

(21) International Application Number: PCT/CA2005/000426

(22) International Filing Date: 18 March 2005 (18.03.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/554,357 19 March 2004 (19.03.2004) US

(71) Applicant *(for all designated States except US)*: **BGT BIOGRAPHIC TECHNOLOGIES INC.** [CA/CA]; 3981 St-Laurent Blvd., Suite M1, Montréal, Québec H2W 1Y5 (CA).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **KRUSZEWSKI, Paul** [CA/CA]; 234 Redfern, Westmount, Québec H3Z 2G3 (CA).

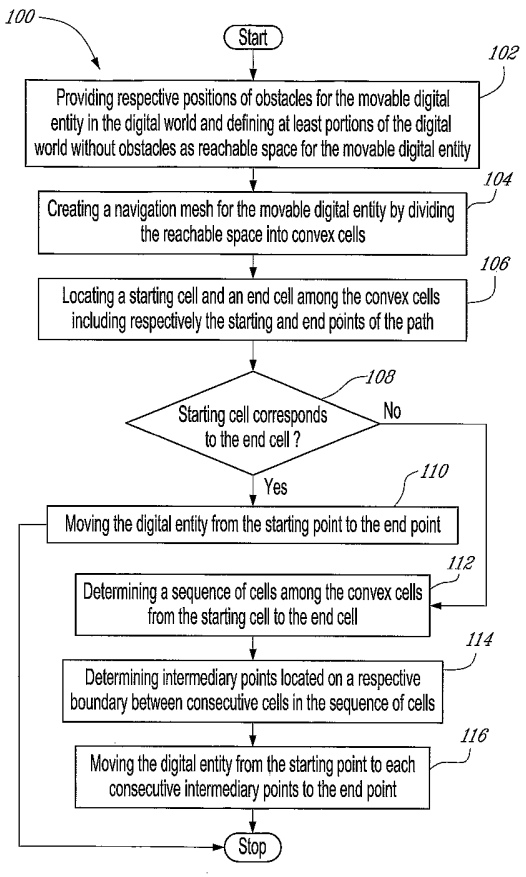
(74) Agents: **FOURNIER, Claude** et al.; BCF LLP, 1100 René-Lévesque Blvd. West, 25th Floor, Montréal, Québec H3B 5C9 (CA).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO,

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR ON-SCREEN NAVIGATION OF DIGITAL CHARACTERS OR THE LIKES



(57) Abstract: A method for moving a digital entity such as a character or an object on-screen from a starting point to an end point in a digital world by: providing the position of the obstacles for the digital entity and defining the portion of the digital world without obstacles as reachable space; creating a navigation mesh for digital entity by dividing the reachable space into convex cells; locating the starting and ending cells among the convex cells; if the starting cell corresponds to the ending cell then the digital entity is moved from the starting to the ending point. If the starting cell does not correspond to the end cell determining intermediary points located on the boundary between consecutive cells in a sequence of cells among the convex cells from the starting to the end point and moving the digital entity from the starting point to each consecutive intermediary point to the end point.

WO 2005/091198 A1

SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

TITLE OF THE INVENTION

METHOD AND SYSTEM FOR ON-SCREEN NAVIGATION OF DIGITAL CHARACTERS OR THE LIKES

FIELD OF THE INVENTION

The present invention relates to the digital entertainment industry and to computer simulation. More specifically, the present invention concerns a method and system for on-screen navigation of digital objects or characters.

BACKGROUND OF THE INVENTION

Since many years three-dimensional (3D) computer graphics is a well established field having many applications including movie animation and digital effects, gaming and simulations.

Figure 1 of the appended drawings illustrates a generic 3D application from the prior art which can be in the form of an animation package, a video/computer game, a trainer or a simulator for example. The 3D application shares the basic high-level architecture of objects, which can be more generally referred to as digital entity, being manipulated by controllers via input devices or physics and artificial intelligence (AI) systems and made real to the user by synthesizers, including visual rendering and audio.

One level deeper, 3D applications are typically broken down into two components: the simulator and the image generator. As illustrated in Figures 2 and 3, the simulator takes in many inputs from both human operators and CGE the simulator then modifies the world database accordingly and outputs the changes to the image generator for visualization.

A typical simulation/animation loop structure is illustrated in Figure 4. The world state manager first loads up the initialisation data from the world database. For each frame/tick of the simulation, the world state manager updates the controllers, the controller acts accordingly and sends back object updates to the manager. The world state manager then resolves all the object updates into a new world state in the world database (WDB) and passes this to the image generator (IG). The IG updates the characters' body limb positions and other objects and renders them out to the screen.

More recently, procedural animation, which is driven by physics and artificial intelligence (AI) techniques, has been introduced in the fields of 3D animation, visual effects and gaming. AI animation allows augmenting the abilities of digital entertainers and simulators across disciplines. It gives game designers and simulators the breadth, independence and tactics of film actors. Film-makers get the temporal performance of game characters and behavioural realism of simulation entities.

For over twenty years, the visual effects departments of film studios have increasingly relied on computer graphics for whenever a visual effect is too expensive, too dangerous or just impossible to create any other way than via a computer. Unsurprisingly, the demands on an animator's artistic talent to produce even more stunning and realistic visual effects have also increased. It is currently not uncommon that the computer animation team is just as important to the success of a film as the lead actors.

Large crowd scenes, in particular battle scenes, are ideal candidates for computer graphics techniques since the sheer number of extras required make them extremely expensive, their violent nature make them very dangerous, and the use of fantastic elements such as beast warriors make them impractical, if not impossible, to film with human extras. Given the complexity, expense, and danger of such scenes, it is clear that an effective artificial intelligence (AI) animation solution is preferable to actually staging and filming such battles with real human actors. However, despite the clear need for a practical commercial method to generate digital crowd scenes, a satisfactory solution has been a long time in coming.

Commercial animation packages such as Maya™ by Alias Systems have made great progress in the last twenty years to the point that virtually all 3D production studios rely on them to form the basis of their production pipelines. These packages are excellent for producing special effects and individual characters. However, automatic/crowd animation remains a significant problem.

In the computer game field, game AI has been in existence since the dawn of video games in the 1970s. However, it has come a long way since the creation of Pong™ and Pac-Man™. Nowadays, game AI is increasingly becoming a critical factor to a game's success and game developers are demanding more and more from their AI. Today's AI need to be able to seemingly think for themselves and act according to their environment and their experience giving the impression of intelligent behaviour, i.e. they need to be autonomous.

Game AI makes games more immersive. Typically game AI is used in the following situations:

- to create intelligent non-player characters (NPCs), which could be friend or foe to the player-control characters;
- to add realism to the world. Simply adding some non-game play related game AI that reacts to the changing game world can increase realism and enhance the game experience. For example, AI can be used to fill sporting arenas with animated spectators or to add a flock of bats to a dungeon scene;
- to create opponents when there are none. Many games are designed for two or more players however, if there is no one to play against intelligent AI opponents are needed; or

- to create team members when there are not enough. Some games require team play, and game AI can fill the gap when there are not enough players.

Another application for real-time intelligent 3D agents (such as vehicles and people that can interact in 3D cities for example) are military simulation and training (MS&T) and more specifically in the context of Military Operations Urban Terrain (MOUT). Indeed, all conflicts within the last 15 years involving western forces have been urban in nature (e.g., Mogadishu, Bosnia, and Iraq). Nonetheless, real-time crowd simulation in complex urban terrain has been neglected for a multitude of reasons:

- Up until recently the hardware simply was not powerful enough;
- The military establishment moves very slowly to acknowledge change; so they have been correspondingly slow to emphasize it as a need; and
- Application developers have avoided human simulation as it is much more difficult than machine simulation.

Artificial Intelligence's role is to simulate the people (not only the ground forces but also the drivers of vehicles) in the battle, the combatants (blue and red forces) and increasingly the civilians (green forces). In the context of MS&T, these are often called Computer Generated Forces (CGFs) or Semi-Automated Forces (SAFs). The types of possible simulated agents are now described.

Vehicle drivers and pilots: although vehicles have very complex models for physics (e.g., helicopters will wobble realistically as they bank into turns and tanks will bounce as they jump ditches) and weapon / communication systems (e.g., line of sight radios will not work through hills), they tend to have simplistic line of sight navigation systems that fail in the 3D concrete canyons of MOUT (e.g., helicopters fly straight through skyscrapers rather than around them and tanks get confused and stuck in the twisty garbage filled streets of the third world). AI can be used to simulate of the brain of the human driver in the vehicle (with or without the actual body being simulated).

Groups of individual doctrinal combatants: United States government-created SAFs of groups of individual doctrinal combatants limited in their usefulness to MS&T applications since they are limited to US installations only and are unable to navigate properly in urban environments. While aggregate SAFs operate on a strategic and often abstract level, individual combatant simulators operate on a tactical and often 3D physical level.

Groups of individual irregular combatants: by definition irregular combatants such as militiamen and terrorists are non-doctrinal and hence it is difficult to express their personalities and tactics with a traditional SAF architecture.

Crowds of individual non-combatants (clutter): one of the most difficult restrictions of MOUT is how to conduct military operations in an environment that populated with non-combatants. These large civilian

populations can affect a mission by acting as only operational "clutter" to actually affecting the outcome of the battle.

One of the more specific aspects of game AI and MS&T is real-time intelligent navigation of agent per se and, for example, in the context of crowd simulation.

Computer graphics crowd simulation has been pioneered by C.W. Reynolds in "Flocks, Herds, and Schools: A Distributed Behavioural Model", In Computer Graphics, 21 (4) (SIGGRAPH' 87 Conference Proceedings), 25-34, 1987. Reynolds established the core idea that movement is a fundamental goal of intelligence and as such intelligent characters can be represented by a simple vehicle model. This system was extremely effective at producing simple "flocking" behaviours and has been used with great success in the simulation of animals such as bats and birds in non-real-time applications such as film special effects.

Musse and Thalmann in "Hierarchical model for real time simulation of virtual human crowds", in IEEE Transactions on Visualization and Computer Graphics 7, (No. 2 Apr.-Jun. 2001), 152-164, developed a hierarchical model for real-time crowd simulation called ViCrowd in which humans are controlled at not only at the individual but also group level.

By representing crowds as a particle system and using clever pre-computed alpha-channel rendering techniques, Tecchia et al. (Tecchia F. Loscos C., Chrysanthou Y. "Visualizing crowds in real-time", In Computer Graphics Forum 21:1, Eurographics, 2002.) succeed in

simulating and rendering large numbers of very simple milling crowds in pseudo-real-time on a PC. However, the crowds could not react to stimuli.

Reece develops a system for modelling crowds within the Dismounted Infantry Semi-Automated Forces (DISAF) system (Reece, D.A. "Crowd Modeling in DISAF.", In Proceedings of the Eleventh Conference on Computer-Generated Forces and Behavior Representation (Orlando, FL, May 7-9, 2002), 87-95.).

Sung et al. expand on the notion of crowd members as intelligent particles by allowing the author to paint information for the crowd's behaviour directly to the world geometry. This authoring optimization limits the crowd members to dynamically change between varied behaviours such as transitioning from milling to panicking (Sung, M., Gleicher, M., and Chenney, S. "Scalable behaviors for Crowd Simulation", In Computer Graphics Forum 23:3, Eurographics, 2004.).

Movement is fundamental to all entities in a simulation whether bipedal, wheeled, tracked or aerial. Hence, an AI system should allow an entity to navigate in a realistic fashion from point X to point Y. For example, for a vehicle this means in normal operations driving on the road and obeying traffic rules such as staying in lane and stopping a traffic lights; and in military operations, avoiding roadblocks and trying not to run over civilian bystanders. For a militiaman, this means in normal operations walking down the sidewalk in an inconspicuous fashion but in military operations quickly running into a building, up its staircase and onto the rooftop to launch an RPG. Intelligent navigation can be broken down into

two basic levels: dynamically finding a path from X to Y and avoiding dynamic obstacles along that path.

An example of agent navigation is following a pre-determined path (e.g., guards on a patrol path). Such a predetermined path is an ordered set of waypoints that digital characters may be instructed to follow. For example, a path around a racetrack would consist of waypoints at the turns of the track. Path following consists of locating the nearest waypoint on the path, navigating to it by direct line of sight and then navigating to the next point on the path. The character looks for the next waypoint when it has arrived the current waypoint (i.e., within the waypoint's bounding sphere). For example, Figure 5 shows how a path can be used to instruct a character to patrol an area in a certain way.

It has been found that path following as a navigation mechanism works well when not only both the start and destination are known but also the path is explicitly known.

However, this is the exception rather than the rule as normally we want to tell the character the destination and let it find its own path around barriers based on its knowledge of itself and its world.

A method for automatically moving a digital entity on-screen from starting to end points in a digital world is however desirable.

OBJECTS OF THE INVENTION

An object of the present invention is therefore to provide an improved navigation method for a digital entity in a digital world.

Another object of the invention is to provide a method for automatically moving a digital entity on-screen from start to end points.

SUMMARY OF THE INVENTION

More specifically, in accordance with a first aspect of the present invention, there is provided a method in a computer system for moving at least one digital entity on-screen from starting to end points in a digital world, comprising:

i) providing respective positions of obstacles for the at least one movable digital entity in the digital world; defining at least portions of the digital world without obstacles as reachable space for the at least one movable digital entity;

ii) creating a navigation mesh for the at least one movable digital entity by dividing the reachable space into at least one convex cell;

iii) locating a start cell and an end cell among the at least one convex cell including respectively the start and end points; and

iv) verifying whether the starting cell corresponds to the end cell; if the starting cell corresponds to the end cell, then: iv)a) moving the at least one movable digital entity from the starting point to the end point; if the starting cell does not correspond to the end cell, then iv)b) i)

determining a sequence of cells among the at least one convex cell from the starting cell to the end cell, and iv)b)ii)determining at least one intermediary point located on a respective boundary between consecutive cells in the sequence of cells, and iv)b)iii) moving the at least one movable digital entity from the starting point to each consecutive the at least one intermediary point to the end point.

In accordance to a second aspect of the present invention, there is provided a system for moving a digital entity on-screen from starting to end points in a digital world, comprising:

a world database for storing information about the digital world and for providing respective positions of obstacles for the movable digital entity in the digital world;

a navigation module

i) for defining at least portions of the digital world without obstacles as reachable space for the movable digital entity;

ii) for creating a navigation mesh for the movable digital entity by dividing the reachable space into at least one convex cell;

iii) for locating a start cell and an end cell among the at least one convex cell including respectively the start and end points; and

iv) for verifying whether the starting cell corresponds to the end cell; and

if the starting cell does not correspond to the end cell, for further

v) determining a sequence of cells among the at least one convex cell from the starting cell to the end cell, and vi) determining at least one intermediary point

located on a respective boundary between consecutive cells in the sequence of cells;

and

a simulator coupled to the navigation module and to the world database for moving the digital entity on-screen via an image generator coupled to the simulator from the starting point to the end point if the starting cell corresponds to the end cell as verified in iv); or for moving the digital entity from the starting point to each consecutive the at least one intermediary point to the end point; if the starting cell does not correspond to the end cell.

It is to be noted that the expression "character" should be construed herein as broadly as "entity".

Other objects, advantages and features of the present invention will become more apparent upon reading the following non restrictive description of preferred embodiments thereof, given by way of example only with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

In the appended drawings:

Figure 1, which is labeled "prior art", is a block diagram illustrating the first level of a generic three-dimensional (3D) application;

Figure 2, which is labeled "prior art", is a block diagram

illustrating the second level of the generic 3D application from Figure 1;

Figure 3, which is labeled "prior art", is an expanded view of the block diagram from Figure 2;

Figure 4, which is labeled "prior art", is a flowchart illustrating the flow of data from the generic 3D application from Figure 1 to the image generator part of the 3D application from Figure 1;

Figure 5, which is labeled "prior art", is a schematic view illustrating the use of waypoint to navigate a digital entity in a digital world;

Figure 6 is a block diagram illustrating a system for on-screen animation of digital entities including a navigation module embodying a system for moving a digital entity on-screen from starting to end points in a digital world according to an illustrative embodiment of the present invention;

Figure 7 is a flowchart illustrating a method for moving a digital entity on-screen from starting to end points in a digital world according to an illustrative embodiment of the present invention;

Figure 8 is a schematic view illustrating a two-dimensional barrier used with the method of Figure 7;

Figure 9 is a schematic view illustrating a three-dimensional barrier used with the method of Figure 7;

Figure 10 is a schematic view illustrating a co-ordinate system used with the method of Figure 7;

Figure 11 is a top plan schematic view of a two-dimensional world in the form of a one-floor building according to a first example of the reachable space for a specific movable digital entity according to the method from Figure 7;

Figure 12 is a top plan schematic view of the one-floor building from Figure 11 illustrating a navigation mesh created through the method from Figure 7;

Figure 13 is a schematic view of a connectivity graph obtained from the navigation mesh from Figure 12;

Figure 14 is a top plan schematic view of a two-dimensional world according to a second example of the reachable space for a specific movable digital entity according to the method from Figure 7;

Figure 15 is a top plan schematic view of the world from Figure 11 illustrating a navigation mesh created through the method from Figure 7;

Figure 16 is a top plan schematic view similar to Figure 15, illustrating steps from the method from Figure 7;

Figure 17 is a top plan view schematic view similar to Figure 15, illustrating a path resulting from the method from Figure 7;

Figure 18 is a top plan view schematic view similar to Figure 17, illustrating a first alternative path to the path illustrated in Figure 17 resulting from the blocking of a first passage;

Figure 19 is a top plan view schematic view similar to Figure 18, illustrating a second alternative path to the path illustrated in Figures 17 and 18 further resulting from the blocking of a second passage;

Figure 20 is a top plan schematic view similar to Figure 17, illustrating a third alternative path to the path illustrated in Figure 17, resulting from new starting and end points;

Figure 21 is a top plan view schematic view similar to Figure 20, illustrating an alternative path to the path illustrated in Figure 20, resulting from doubling the width of the movable digital entity;

Figure 22 is a perspective view of a floor plan generator on a small city part of a simulator;

Figure 23 is a perspective view illustrating the navigation mesh created from the floor plan generator from Figure 22 using the method from Figure 7;

Figure 24 is a perspective view of a digital world in the form of a city street;

Figure 25 is a perspective view of the city street from Figure 24, illustrating the navigation mesh creating step according to the method from Figure 7, including the used of blind data to characterize the resulting cells;

Figure 26 is a cut out perspective view of a digital world in the form of a building;

Figure 27 is a perspective view of the navigation mesh resulting from the building from Figure 26 using the method from Figure 7;

Figure 28 is flowchart of a collision avoidance method for a digital entity moving on-screen from starting to end points in a digital world according to a specific illustrative embodiment of the present invention;

Figure 29 is a perspective view of an entity in a 3D application, in the form of a character, illustrating character's sensor according to the present invention;

Figure 30 is a top plan view of an entity in a digital world illustrating the entity's vision sensor according to the present invention, and more specifically illustrating the field of view provided by the sensor;

Figure 31 is a perspective view of an entity in a digital world similar to Figure 30 illustrating the selection of a sub path to avoid obstacles according to a specific embodiment of the method from the Figure 7;

Figures 32A-32C are schematic views illustrating avoidance strategies (Figures 32B-32C) that can be used by a movable digital entity according to a specific embodiment of the method from Figure 7 in the case of a stationary obstacle (Figure 32A);

Figures 33A-33C are schematic views illustrating avoidance strategies (Figures 33B-33C) that can be used by a movable digital entity according to a specific embodiment of the method from Figure 7 in the case of an incoming obstacle (Figure 33A);

Figures 34A-34C are schematic views illustrating avoidance strategies (Figures 34B-34C) that can be used by a movable digital entity according to a specific embodiment of the method from Figure 7 in the case of an outgoing obstacle (Figure 34A);

Figures 35A-35C are schematic views illustrating avoidance strategies (Figures 35B-35C) that can be used by a movable digital entity according to a specific embodiment of the method from Figure 7 in the case of an sideswiping obstacle (Figure 35A); and

Figures 36A-36E are schematic views illustrating paths for simultaneously moving five movable digital entities using the method from

Figure 7 and applying a group-based movement modifier.

DETAILED DESCRIPTION

A system 10 for on-screen animation of digital image entities (IE) including a navigation module 12 embodying a method for moving a digital entity on-screen from starting to end points in a digital world according to an illustrative embodiment of the present invention will now be described with reference to Figure 6.

Since the system 10 shares similarity with conventional 3D applications, such as the one illustrated in Figure 2, and for concision purposes, only the differences between the system 10 and conventional 3D application systems will be described herein in more detail.

The system 10 comprises a simulator 14 a world database (WDB) 16 coupled to the simulator 14, a plurality of image generators (IG) 18 (three shown) coupled to both the world DB and to the simulator 14, a navigation module 12 according to an illustrative embodiment of the present invention, coupled to the simulator 14, a decision-making module, also coupled to the simulator 14, and a plurality (three shown) of animation control module, each coupled to a respective IG 18.

The number of IG 18 may of course vary depending on the application. For example, in the case wherein the system 10 is embodied in a 3D animation application, the number of IG 18 may effect the rendering time.

The simulator 14 and IG 18 may be in the form of a single computer. The world database 16 is stored on any suitable memory means, such as, but not limited to, a hard drive, a dvd or cd-rom disk to be read on a corresponding drive, or a random-access memory part of the computer 14.

The simulator 14 and IG 18 are in the form of computers or of any processing machines provided with processing units, which are programmed with instructions for animating, simulating or gaming as will be explained hereinbelow in more detail.

The simulator 14, IG 18 and world DB 16 can be remotely coupled via a computer network (not shown) such as Internet.

Of course, depending on the application of the system 10, the simulator 14 can take another form such as a game engine or a 3D animation system.

The modules 12, 20 and 22 are in the form of sub-routines or dedicated instructions programmed in the simulator 14 for example. The characteristics and functions of the modules 20, 22 and more specifically of module 12 will become more apparent upon reading the following non-restrictive description of a method 100 for moving a digital entity on-screen from a starting point to an end point in a digital world according to an illustrative embodiment of the present invention.

The method 100, which is illustrated in Figure 7, comprises the following steps:

102 – providing respective positions of obstacles for the movable digital entity in the digital world and defining at least portions of the digital world without obstacles as reachable space for the movable digital entity;

104 - creating a navigation mesh for the movable digital entity by dividing the reachable space into convex cells;

106 - locating a starting cell and an end cell among the convex cells including respectively the starting and end points;

108 - verifying whether the starting cell corresponds to the end cell;

if the starting cell corresponds to the end cell then :

110 - moving the digital entity from the starting point to the end point and stopping the method;

if the starting cell does not correspond to the end cell then :

112 - determining a sequence of cells among the convex cells from the starting cell to the end cell;

114 - determining intermediary points located on a respective boundary between consecutive cells in the sequence of cells; and

116 - moving the digital entity from the starting point to each consecutive intermediary points to the end point.

Each of these steps will now be described in more details.

In step 102, the respective position of obstacles for the movable digital entity in the digital world are defined yielding the portion of the digital world without obstacles as reachable space for the movable digital entity. The reachable space can be defined as regions of the digital world enclosed by barriers.

It is to be noted that what the simulator 14 will consider an obstacle for a specific movable digital entity may not be an obstacle for another one as further examples given hereinbelow will allow to enlighten.

Of course, the digital world may have been previously defined including any autonomous or non-autonomous entity with which the movable digital entity may interact. The concept of digital world and of digital entity will now be described according to an illustrative embodiment of the present invention.

The digital world model includes image object elements. The image object elements include two or three-dimensional (2D or 3D) graphical representations of objects, autonomous and non-autonomous characters, building, animals, trees, etc. It also includes barriers, terrains, and surfaces. As will become more apparent upon reading the following description, the movable entity that is to be moved using the method 100 can be either autonomous or non-autonomous. The concepts of autonomous and non-autonomous characters and objects will be described hereinbelow in more detail.

As it is believed to be commonly known in the art, the graphical representation of objects and characters can be displayed, animated or not, on a computer screen or on another display device, but can also inhabit and interact in the virtual world without being displayed on the display device.

Barriers are triangular planes that can be used to build walls, moving doors, tunnels, etc., or any obstacles for any movable entity in the digital world. Terrains are 2D height-fields to which entities can be automatically bound (e.g. keep soldier characters marching over a hill). Surfaces are triangular planes that may be combined to form fully 3D shapes to which autonomous characters can also be constrained.

In combination, these elements are to be used to describe the world in which the characters inhabit. They are stored in the world DB 16.

In addition to the image object elements, the digital world model includes a solver, which allows managing entities, including autonomous characters, and other objects in the digital world.

The solver can have a 3D configuration, to provide the entities with complete freedom of movement, or a 2D configuration, which is more computationally efficient, and allows an operator to insert a greater number of movable entities in a scene without affecting performance of the animation system.

A 2D solver is computationally more efficient than a 3D solver since the solver does not consider the vertical (y) co-ordinate of an image object element or of an entity. The choice between the 2D and 3D configuration depends on the movements that are allowed in the virtual world by the movable entities and other objects. If they do not move in the vertical plane then there is no requirement to solve for in 3D and a 2D solver can be used. However, if any entity requires complete freedom of movement, a 3D solver is used. It is to be noted that the choice of a 2D solver does not limit the dimensions of the virtual world, which may be 2D or 3D.

Non-autonomous characters are objects in the digital world that, even though they may potentially interact with the digital world, are not driven by the solver. These can range from traditionally animated characters (e.g. the leader of a group) to player characters to objects (e.g. flying debris) driven by other components of the simulator.

Barriers are used to represents obstacles for movable entities, and are equivalent to one-way walls, i.e. an object or a digital entity inhabiting the digital world can pass through them in one direction but not in the other. When a barrier is created, spikes (forward orientation vectors) are used to indicate the side of the wall that can be detected by an object or an entity. Therefore, an object or an entity can pass from the non-spiked side to the spiked side, but not vice-versa. It is to be noted that, in some application, a specific avoidance constraint can be defined and activated for a digital entity to attempt to avoid the barriers in the

digital world. The concept of behaviours and constraints will be described hereinbelow in more detail.

As illustrated in Figures 8 and 9 respectively, a barrier is represented in a 2D solver by a line and by a triangle in a 3D solver. The direction of the spike for 2D and 3D barriers is also shown in Figures 8-9 (see arrows 24 and 26 respectively) where P1-P3 refers to the order in which the points of the barrier are drawn. Since barriers are unidirectional, two-sided barriers are made by superimposing two barriers and by setting their spikes opposite to each other.

Each barrier can be defined by the following parameters:

Parameter	Description
Exists	This parameter allows the system to determine whether or not the barrier exists in the solver world. If this is set to *off* the solver ignores the barrier.
Collidable	This parameter allows the system to determine whether or not collisions with other collidable objects will be resolved. If this parameter is set to *off* entities can pass through the barrier from either side.
Opaque	This parameter allows setting whether or not objects can see through the barrier using a sensor as will be explained hereinbelow.
Surface	This parameter allows setting whether or not the barrier will be considered as a surface. A barrier that is a surface is considered for surface hugging by the solver.
Use Bounding Box	This parameter allows the system to determine whether or not to create barriers based on the bounding boxes for the selected objects. If the currently active solver has a 2D configuration then the barriers created with this option will only be created around the bounding-perimeter. If the

Parameter	Description
	solver is 3D, then barriers will be created and positioned the same way as the bounding box for the object.
Use Bounding Box Per Object	If the "Use Bounding Box "parameter is enabled and this option is also enabled a barrier-bounding box per selected object will be created. If it is disabled, a barrier-bounding box will be created at the bounding box for the group of selected items.
Reverse Barrier Normal	This parameter reverses the normals for the selected barriers.
Group Barriers	When this parameter is activated, all barriers are grouped under a group node.

As it is commonly known, a bounding box is a rectilinear box that encapsulates and bounds a 3D object.

The solver of the digital world model may include subsolvers, which are the various engines of the solver that are used to run the simulation. Each subsolver manages a particular aspect of object and simulation in order to optimize computations.

As it is commonly known in the art, each animated digital entity is associated to animation clips allowing representing the entity in movement in the digital world. According to a specific embodiment of the present invention, virtual sensors are assigned to and used by some entities to allow them gathering data information about image object elements or other entities within the digital world. Decision trees can also

be used for processing the data information resulting in selecting and triggering one of the animation cycle or selecting a new behaviour.

As it is believed to be well known in the art, an animation cycle, which will also be referred to herein as "animation clip" is a unit of animation that typically can be repeated. For example, in order to get a character to walk, the animator creates a "walk cycle". This walk cycle makes the character walks one iteration. In order to have the character walk more, more iterations of the cycle are played. If the character speeds up or slows down during time, the cycle is "scaled" accordingly so that the cycle speed matches the character displacement so that there is no slippage (e.g., it looks like the character is slipping on the ground).

The autonomous image entities are tied to transform nodes of the animating engine (or platform). The nodes can be in the form of locators, cubes or models of animals, vehicles, etc. Since animation clips and transform nodes are believed to be well known in the art, they will not be described herein in more detail.

Figure 10 shows a co-ordinate system for moving the IE and used by the solver.

Examples of attributes that can be associated to a movable digital image entity are briefly described in the following tables.

Attribute	Description
Exists	This attribute allows the solver whether or not to consider the character in the world. If this attribute it is

Attribute	Description
	set to *off*, the solver ignores the IE and does not update it. This attribute allows dynamically creating and killing characters (IEs).
Hug Terrain	This attribute allows setting whether or not the IE will hug the terrain. If this is set to *on* the IE will remain on the terrain. It is to be noted that terrains are activated only when the solver is in 2D mode.
Align Terrain Normal	This attribute allows setting whether or not the IE will align with the terrain's surface normal. This parameter is taken into account when the IE is hugging the terrain.
Terrain Offset	This attribute specifies an extra height that will be given to a character when it is on a terrain. The offset is only taken into account when the IE is hugging the terrain. A positive value causes the IE to float above the terrain, and a negative value causes the IE to be sunken into the terrain.
Hug Surface	This attribute specifies whether or not the IE will hug a surface. A surface is a barrier with the *Surface* attribute set to true. Surface hugging applies in a 3D solver. The IE hugs the nearest surface below it.
Align Surface Normal	This attribute specifies whether or not the IE's up orientation aligns to the surface normal. This parameter is taken into account when the IE is on a surface. An IE with both hug surface and align surface enabled will follow a 3D surface defined by barriers, while aligning the up of the IE according to the surface.
Surface Offset	This attribute specifies an extra height that will be given to an IE when it is on a surface. The offset is taken into account only when the IE is hugging a surface. A positive value will cause the IE to float above the surface, and a negative value will cause the IE to be sunken into the surface.
Collidable	This attribute specifies whether or not collisions with other collidable objects will be resolved. If this parameter is set to false then nothing will prevent the IE from occupying the same space as other objects, as would (for instance) a ghost.
Radius	This attribute specifies the radius of the IE's bounding sphere. Since the concept of bounding sphere is

Attribute	Description
	believed to be well known in the art, it will not be described herein in more detail.
Right Turning Radius	This attribute specifies the maximum right turning angle (clockwise yaw) per frame measured in degrees. The angle can range from 0-180 degrees.
Left Turning Radius	This attribute specifies the maximum left turning angle (anticlockwise yaw) per frame measured in degrees. The angle can range from 0-180 degrees.
Up Turning Radius	This attribute specifies the maximum up turning angle (positive pitch) per frame measured in degrees. The angle can range from 0-180 degrees.
Down Turning Radius	This attribute specifies the maximum down turning angle (negative pitch) per frame measured in degrees. The angle can range from 0-180 degrees.
Maximum Angular Acceleration	This attribute specifies the maximum positive change in angular speed of the IE, measured in degrees/frame^2. If this variable is larger than the turning radii, it will have no effect. If set smaller than the turning radii, it will increase the IE's resistance to angular change. In general, the maximum angular acceleration should be set smaller than the maximum angular deceleration to avoid overshoot and oscillation effects.
Maximum Angular Deceleration	This attribute specifies the maximum negative change in angular speed of the character, measured in degrees/frame^2. If this variable is larger than the turning radii, it will have no effect. If set smaller than the turning radii, it will increase the IE's resistance to angular change. In general, the maximum angular acceleration should be set smaller than the maximum angular deceleration to avoid overshoot and oscillation effects.
Maximum Pitch (a.k.a. Max Stability Angle)	This attribute specifies the maximum angle of deviation from the z-axis that the object's top vector may have, measured in degrees. The maximum pitch can range from -180 to 180 degrees. This attribute can be used to limit how steep a hill the IE can climb or descend to prevent objects from incorrectly turning upside down.
Maximum Roll	This attribute specifies the maximum angle of deviation from the x-axis that the object's top vector may have, measured in degrees. The maximum can roll range from

Attribute	Description
	-180 to 180 degrees. This attribute can be used to limit the side-to-side tilting of the IE to prevent objects from incorrectly turning upside down.
Min Speed	This attribute specifies the minimum speed (distance units/frame) of the IE.
Max Speed	This attribute specifies the maximum speed (distance units/frame) of the IE.
Max Acceleration	This attribute specifies the maximum positive change in speed (distance units/frame^2) of the IE.
Max Deceleration	This attribute specifies the maximum negative change in speed (distance units/frame^2) of the IE.
Brake Padding and *Braking Softness*	Braking is only applied when an IE tries to turn at an angle greater than one of its turning radii. When this occurs, the *Brake Padding* and *Braking Softness* parameters work together to slow the IE down so that it doesn't overshoot the turn. *Brake Padding* controls when braking is applied. It can be set to 0, which means that braking will be applied as soon as the object tries to turn beyond one of its maximum turning radii, or 1 which means that braking is never applied. Values between 0 and 1 interpolate those extremes. The default value can be set to 0. *Braking softness* controls the gentleness of braking and can be set to any positive number, including zero. A value of 0 corresponds to maximum braking strength and the IE will come to a complete stop as soon as the brakes are applied. A value of 1 corresponds to normal strength, and values greater than 1 result in progressively gentler braking. The default value can be set to 1. Setting a very large *Braking Softness* (effectively $+\infty$) is equivalent to setting the *Brake Padding* to 1, which is equivalent to turning braking off.
Forward Motion Only	This attribute is set to *on* to limit the movement of the IE such that it may only move in the direction it is facing. *Off* will allow the IE to move and face in different directions, provided that its behaviours are set up to produce such motion. The default value can be set to *on*.

Attribute	Description
Initial Speed	This attribute specifies the initial speed of the IE (distance units/frame) at start time.
Initial Position X, Y, Z	This attribute specifies the initial position of the IE at start time. The default is the position where the object was created.
Initial Orientation X, Y, Z	This attribute specifies the initial orientation of the IE at start time. The default is the orientation of the object when created.
Display Radius	This attribute specifies whether or not the radius and heading of the IE will be displayed.
Current Speed	This attribute specifies the current speed (distance units/frame) of the IE. The solver controls this variable.
Translate	This attribute specifies the current position of the IE. The AI solver controls this variable.
Rotate	This attribute specifies the current orientation of the IE. The solver controls this variable.

Of course, other attributes can also be used to characterize an IE.

The concept of IE behaviour will now be described hereinbelow in more detail.

In addition to attributes, IE from the present invention can also be characterized by behaviours. Along with the decision trees, the behaviours are the low-level thinking apparatus of an IE. They take raw input from the digital world using virtual sensors, process it, and change the IE's condition accordingly.

Behaviours can be categorized, for example, as Locomotive behaviours allowing an IE to move. These locomotive behaviours generate

steering forces that can affect any or all of an IE's direction of motion, speed, and orientation (i.e. which way the IE is facing) for example.

The following table includes examples of behaviours:

Simple behaviours:
- Avoid Barriers
- Avoid Obstacles
- Accelerate At
- Maintain Speed At
- Wander Around
- Orient To

Targeted behaviours:
- Seek To
- Flee From
- Look At
- Follow Path
- Seek To Via Network

Group behaviours:
- Align With
- Join With
- Separate From
- Flock With

A locomotive behaviour can be seen as a force that acts on the IE. This force is a behavioural force, and is analogous to a physical force (such as gravity), with a difference that the force seems to come from within the IE itself.

It is to be noted that behavioural forces can be additive. For example, an autonomous character may simultaneously have more then one active behaviours. The solver calculates the resulting motion of the character by combining the component behavioural forces, in accordance with behaviour's priority and intensity. The resultant behavioural force is then applied to the character, which may impose its own limits and

constraints (specified by the character's turning radius attributes, etc) on the final motion.

The behaviours allow creating a wide variety of actions for IEs. Behaviours can be divided into four subgroups: simple behaviours, targeted behaviours, and group behaviours.

Simple behaviours are behaviours that only involve a single IE.

Targeted behaviours apply to an IE and a target object, which can be any other object in the digital world (including groups of objects).

Group behaviours allow IEs to act and move as a group where the individual IEs included in the group will maintain approximately the same speed and orientation as each other.

Examples of behaviours will now be provided in each of the four categories. Of course, it is believed to be within a person skilled in the art to provide an IE with other behaviours.

Simple Behaviours

Avoid Barriers

The Avoid Barriers behaviour allows a character to avoid colliding with barriers.

Parameters specific to this behaviour may include, for example:

Parameter	Description
Avoid Distance	The distance from a barrier at which the IE will attempt to avoid it. This is effectively the distance at which barriers are visible to the IE.
Avoid Distance Is Speed Adjusted	Whether or not the avoidance distance is adjusted according to the IE's speed. If this is set to *on,* the faster the IE moves, the greater the avoidance distance.
Avoid Width Factor	The avoidance width factor defines how wide the "avoidance capsule" is (the length of the "avoidance capsule" is equal to the Avoid Distance). If a barrier lies within the avoidance capsule, the IE will take evasive action. The value of the avoidance width factor is multiplied by the IE's width in order to determine the true width (and height in a 3D solver) of the capsule. A value of 1 sets the capsule to the same width as the IE's diameter.
Barrier Repulsion Force	Allows controlling how much the IE is pushed away from barriers. A value of 0 indicates no repulsion and the IE will tend to move parallel to nearby barriers. Larger values will add a component of repulsion based on the IE's incident angle.
Avoidance Queuing	Allows controlling the IE's barrier avoidance strategy. If set to on the IE will slow down when approaching a barrier, if set to *off* the IE will dodge the barrier. The default value can be set to *off*.

The Avoid Obstacles behaviour allows an IE to avoid colliding with obstacles, which can be other autonomous and non-

autonomous image entities. Similar parameters than those detailed for the Avoid Barriers behaviour can also be used to define this behaviour.

Accelerate At

The Accelerate At behaviour attempts to accelerate the IE by the specified amount. For example, if the amount is a negative value, the IE will decelerate by the specified amount. The actual acceleration/deceleration may be limited by max acceleration and max deceleration attributes of the IE.

A parameter specific to this behaviour is the Acceleration, which represents the change in speed (distance units/frame2) that the IE will attempt to maintain.

Maintain Speed At

The Maintain Speed At behaviour attempts to set the target IE's speed to a specified value. This can be used to keep a character at rest or moving at a constant speed. If the desired speed is greater than the character's maximum speed attribute, then this behaviour will only attempt to maintain the character's speed equal to its maximum speed. Similarly, if the desired speed is less than the character's minimum speed attribute, this behaviour will attempt to maintain the character's speed equal to its minimum speed.

A parameter allowing defining this behaviour is the desired speed (distance units/frame) that the character will attempt to maintain.

Wander Around

The Wander Around behaviour applies random steering forces to the IE to ensure that it moves in a random fashion within the solver area.

Parameters allowing defining this behaviour may be for example:

Parameter	Description
Is Persistent	This parameter allows defining whether or not the desired motion calculated by this behaviour is applied continuously (at every frame) or only when the desired motion changes (see the *Probability* attribute). A persistent Wander Around behaviour produces the effect of following random waypoints. A non-persistent Wander Around behaviour causes the IE to slightly change its direction and/or speed when the desired motion changes.
Probability	This parameter allows defining the probability that the direction and/or speed of wandering will change at any time frame. For example, a value of 1 means that it will change each time frame, a value of 0 means that it will never change. On average, the desired motion produced by this behaviour will change once every 1/probability frames (*i.e.* average frequency = 1/probability)
Max Left Turn	This parameter allows defining the maximum left wandering turn angle in degrees at any time frame.
Max Right Turn	This parameter allows defining the maximum right

Parameter	Description
	wandering turn angle in degrees at any time frame.
Left Right Turn Radii Noise Frequency	This parameter affects the value of the pseudo-random left and right turn radii generated by this behaviour. A valid range can be between 0 and 1. The higher the frequency the more frequent an IE will change direction. The lower the frequency the less often an IE will change direction.
Max Up Turn	This parameter allows defining the maximum up wandering turn angle in degrees at any time frame.
Max Down Turn	This parameter allows defining the maximum down wandering turn angle in degrees at any time frame.
Up Down Turn Radii Noise Frequency	This parameter affects the value of the pseudo-random up and down turn radii generated by this behaviour. The valid range is between 0 and 1. The higher the frequency the more frequent an IE will change direction. The lower the frequency the less often an IE will change direction.
Max Deceleration	This parameter allows defining the maximum wander deceleration (distance units/frame^2) at any time frame.
Max Acceleration	This parameter allows defining the maximum wander acceleration (distance units/frame^2) at any time frame.
Speed Noise Frequency	This parameter affects the value of the pseudo-random speed generated by this behaviour. The valid range is between 0 and 1. The higher the frequency the more frequent an IE will change direction. The lower the frequency the less often an IE will change direction.
Min Speed	This parameter allows defining the minimum speed (distance units/frame) that the behaviour will attempt to maintain.
Use Min Speed	This parameter allows defining whether or not the *Min Speed* attribute will be used.
Max Speed	This parameter allows defining the maximum speed (distance units/frame) that the behaviour will attempt to maintain.
Use Max Speed	This parameter allows defining whether or not the *Max Speed* attribute will be used.

Orient To

The Orient To behaviour allows an IE to attempt to face a specific direction.

Parameters allowing defining this behaviour are:

Parameter	Description
Desired Forward Orientation	This parameter allows defining the direction this IE will attempt to face. For example, a desired forward orientation of (1,0,0) will make an IE attempt to align itself with the x-axis. When a 2D solver is used, the *y* component of the desired forward orientation is ignored.
Relative	If true, then the desired forward orientation attribute is interpreted to be relative to the current character forward. If false, then the desired forward is in absolute world coordinates. By default, this value is set to false.

Targeted Behaviours

The following behaviours apply to an IE (the source) and another object in the world (the target). Target objects can be any object in the world such as autonomous or non-autonomous image entities, paths, groups and data. If the target is a group, then the behaviour applies only to the nearest member of the group at any one time. If the target is a datum, then it is assumed that this datum is of type ID and points to the true target of the behaviour. An ID is a value used to uniquely identify objects in the world. The concept of datum will be described in more detail hereinbelow.

The following parameters, shared by all targeted behaviours, are:

Parameter	Description
Activation Radius	The *Activation Radius* determines at what point the behaviour is triggered. The behaviour will only be activated and the IE will only actively seek a target if the IE is within the activation radius distance from the target. A negative value for the activation radius indicates that there is no activation radius, or that the feature is not being used. This means that the behaviour will always be *on* regardless of the distance between the IE and the target.
Use Activation Radius	This parameter allows defining whether or not the *Activation Radius* feature will be used. If this is *off*, the behaviour will always be activated regardless of the location of the IE.

Seek To

The Seek To behaviour allows an IE to move towards another IE or towards a group of IEs. If an IE seeks a group, it will seek the nearest member of the group at any time. Of course, a Seek To behaviour may be programmed according to the navigation method 100.

Parameters allowing defining this behaviour are for example:

Attribute	Description
Look Ahead Time	This parameter instructs the IE to move towards a projected future point of the object being sought. Increasing the amount of look-ahead time does not necessarily make the Seek To behaviour any "smarter" since it simply makes a linear interpolation based on the target's current speed and position. Using this parameter gives the behaviour

Attribute	Description
	sometimes referred to as "Pursuit".
Offset radius	This parameter allows specifying an offset from the target's centre point that the IE will actually seek towards.
Offset Yaw Angle	This parameter allows defining the angle in degrees about the front of the target in the yaw direction that the offset is calculated. The angle describes the amount of counter-clockwise rotation about the front of the target. For example, to make a soldier follow a leader, the soldier seek the leader with a positive offset radius and an offset yaw angle of 180°. This attribute is ignored if the *Strafing* parameter is turned on. Strafing automatically sets an appropriate value for the offset angle.
Offset Pitch Angle	This parameter is the similar to *Offset Yaw Angle* but for the offset angle in the pitch direction relative to the target object's orientation. This applies only in the case of a 3D solver and will be ignored in a 2D solver.
Contact Radius	This parameter allows specifying a proximity radius at which point the behaviour is triggered. In other words, it defines the point at which the IE has reached the target and has no reason to continue seeking it. If the parameter is set to -1, this feature is turned *off* and the IE will always attempt to seek the target regardless of their relative positions. Since the contact radius extends the target's radius, a value of 0 means that the IE will stop seeking when it touches (or intersects with) the target.
Use Contact Radius	This parameter allows defining whether or not the *Contact Radius* feature is used. If this is *off*, the IE will always attempt to seek the target regardless of their relative positions
Slowing Radius	The slowing radius specifies the point at which the IE begins to attempt to slow down and arrive at a standstill at the contact radius (or earlier). If set to -1, this feature is turned *off* and the IE will never attempt to stop moving when it reaches its target. This feature of Seek To is sometimes referred to as "Arrival". It is to be noted that the slowing radius is taken to be the distance from the contact radius, which itself is the distance from the external radius of the target.

Attribute	Description
Use Slowing Radius	This parameter allows defining whether or not the *Slowing Radius* feature is used. If this is *off*, the IE will not attempt to slow down when reaching the target.
Desired Speed	The desired speed instructs an IE to move towards the target at the specified speed. If this is set to a negative number or *Use Desired Speed* is *off*, this feature is turned off and the IE will attempt to approach the target at its maximum speed.
Use Desired Speed	This parameter allows defining whether or not the *Desired Speed* attribute will be used. If this is *off*, the IE will attempt to approach the target at its maximum speed.

Flee From

The Flee From behaviour allows an IE to flee from another IE or from a group of IEs. When an IE flees from a group, it will flee from the nearest member of the group at any time. The Flee From behaviour has the same attributes as the Seek To behaviour, however, it produces the opposite steering force. Since the parameters allowing defining the Flee From behaviour are very similar to those of the Seek To behaviour, they will not be described herein in more detail.

Look At

The Look At behaviour allows an IE to face another IE or a group of IEs. If the target of the behaviour is a group, the IE attempts to look at the nearest member of the group.

Strafe

The Strafe behaviour causes the IE to "orbit" its target, in other words to move in a direction perpendicular to its line of sight to the target. A probability parameter allows to determine how likely it is at each frame that the IE will turn around and start orbiting in the other direction. This can be used, for instance, to make a moth orbit a flame.

For example, the effect of a guard walking sideways while looking or shooting at its target can be achieved by turning off the guard's Forward Motion Only property, and adding a Look At behaviour set towards the guard's target. It is to be noted that, to do this, Strafe is set to Affects direction only, whereas Look At is set to Affects orientation only.

A parameter specific to this behaviour may be, for example, the Probability, which may take a value between 0 and 1 that determines how often the IE change direction of orbit. For example, at 24 frames per second, a value of 0.04 will trigger a random direction change on average every second, whereas a value of 0.01 will trigger a change on average every four seconds.

Go Between

The Go Between behaviour allows an IE to get in-between the first target and a second target. For example this behaviour can be used to enable a bodyguard character to protect a character from a group of enemies.

The following parameter allow specifying this behaviour:, which may take a value between 0 and 1 that determines how close to the second target one wish the entity to go.

Follow Path

The Follow Path behaviour allows an IE to follow a path. For example this behaviour can be used to enable a racecar to move around a racetrack.

The following parameters allow defining this behaviour:

Parameter	Description
Use Speed Limits	This parameter allows defining whether or not the IE will attempt to use the speed limits of the waypoints on the path. If this parameter is set to *off*, the IE will attempt to follow the path at its maximum speed.
Path Is Looped	This parameter allows defining whether or not the IE will go to the first waypoint when it reaches the last waypoint. If the parameter is set to *off*, when the IE reaches the last waypoint it will hover around that waypoint.

Group Behaviours

Group behaviours allow grouping individual IEs so that they act as a group while still maintaining individuality. Examples include a school of fish, a flock of birds, etc.

The following parameters may be used to define group behaviours:

Parameter	Description
Neighbourhood Radius	This parameter is similar to the "activation radius" in targeted behaviours. The IE will "see" only those members that are within its neighbourhood radius. The neighbourhood radius is independent of the IE's radius.
Use Max Neighbours	This parameter allows defining whether or not the *Max Neighbours* attribute will be used. If this parameter is set to *off*, then all the group members in the neighbourhood radius are used to calculate the effect of the behaviour.
Max Neighbours	This parameter allows defining the maximum number of neighbours to be used in calculating the effect of the behaviour.

The following includes brief descriptions of examples of group behaviours.

Align With

The Align With behaviour allows an IE to maintain the same orientation and speed as other members of a group. The IE may or may not be a member of the group.

Join With

The Join With behaviour allows an IE to stay close to members of a group. The IE may or may not be a member of the group.

An example of parameter that can be used to define this behaviour is the Join Distance, which is similar to the "contact radius" in

targeted behaviours. Each member of the group within the neighbourhood radius and outside the join distance is taken into account when calculating the steering force of the behaviour. The join distance is the external distance between the characters (i.e. the distance between the outsides of the bounding spheres of the characters). The value of this parameter determines the closeness that members of the group attempt to maintain.

Separate From

The Separate From behaviour allows an IE to keep a certain distance away from members of a group. For example, this can be used to prevent a school of fish from becoming too crowded. The IE to which the behaviour is applied may or may not be a member of the group.

The Separation Distance is an example of parameters that can be used to define this behaviour. Each member of the group within the neighbourhood radius and inside the separation distance will be taken into account when calculating the steering force of the behaviour. The separation distance is the external distance between the IEs (i.e. the distance between the outsides of the bounding spheres of the IEs). The value of this parameter determines the external separation distance that members of the group will attempt to maintain.

Flock With

This behaviour allows IEs to flock with each other. It combines the effects of the Align With, Join With, and Separate From behaviours.

The following table describes parameters that can be used to define this behaviour:

Parameter	Description
Alignment Intensity	This parameter allows defining the relative intensity of the Align With behaviour.
Join Intensity	This parameter allows defining the relative intensity of the Join With behaviour.
Separation Intensity	This parameter allows defining the relative intensity of the Separate From behaviour.
Join Distance	This parameter determines the closeness that members of the group will attempt to maintain.
Separation Distance	This parameter determines the external separation distance that members of the group will attempt to maintain.

Combining Behaviours

An IE can have multiple active behaviours associated thereto at any given time. Therefore, means can be provided to assign importance to a given behaviour.

A first means to achieve this is by assigning intensity and priority to a behaviour. The assigned intensity of a behaviour affects how strong the steering force generated by the behaviour will be. The higher

the intensity the greater the generated behavioural steering forces. The priority of a behaviour defines the precedence the behaviour should have over other behaviours. When a behaviour of a higher priority is activated, those of lower priority are effectively ignored. By assigning intensities and priorities to behaviours the animator informs the solver which behaviours are more important in which situations in order to produce a more realistic animation.

In order for the solver to calculate the new speed, position, and orientation of an IE, the solver calculates the desired motion of all behaviours, sums up these motions based on each behaviour's intensity, while ignoring those with lower priority, and enforces the maximum speed, acceleration, deceleration, and turning radii defined in the IE's attributes. Finally, braking due to turning may be taken into account. Indeed, based on the values of the character's Braking Softness and Brake Padding attributes, the character may slow down in order to turn.

Returning to Figure 7, the detailed description of the method 100 will now continue with reference to a simple digital world in the form of inner rooms of a one-floor building 30 is illustrated in Figure 11, the walls 32, represented by dark lines, being obstacles and the area enclosed within defining the reachable space 34 for the movable entity (not shown).

In step 104, a navigation mesh 35 is created for the movable digital entity (not shown). This is achieved by dividing or converting the reachable space 34 into convex cells 36 as illustrated in Figure 12 for the example of the one-floor building from Figure 11.

The navigation mesh 35 can be created either manually or automatically using, for example, the collision layer or the rendering geometry. A collision layer is a geometric mesh that is a simplification of the rendering geometry for the purposes of physics collision detection/resolution. In this case, the navigation mesh is the subset of the collision layer upon which the movable entity could move (this is typically the floors and not the walls).

Deriving the navigation mesh from the rendering geometry requires simplifying the geometry as much as possible and fusing the geometry into a seamless a mesh as possible (e.g., removal of intersecting polygons, etc.).

In a manual creation, a 3D operator (typically a 3D artist) inspects the input geometry, fuses the polygons correctly and strips out the non-reachable space. It is to be noted that algorithms exist that can automatically handle this to a high degree.

Convex polygons are used as cells in the creation of the navigation mesh 35 since any point within such a cell is directly reachable in a straight line to any other point in the cell.

An edge Exy connecting cells Cx and Cy in the navigation mesh will be considered "passable", if the entity can pass from cell Cx to Cy via Exy.

In step 106, the starting and end points (not shown) are located and the corresponding cells that includes each of those two points are identified.

The expressions "starting point" and "end point" should not be construed herein in a limited way. Indeed, unless the digital movable entity is pixel size, the starting and end point will refer to a location or a zone in the virtual world.

In step 108, a first verification is done whether the starting and end points are both located in the same cell. If this is the case, the method 100 proceeds with step 110, wherein the digital entity is moved from the starting to the end point before the method stops. As it has been mentioned hereinabove, the use of convex cells allows the digital movable entity to move or to be moved in straight line within a cell. However, in some cases, objects or other movable entities for example, may force the movable entities to adopt a collision avoidance strategy. As will be described hereinbelow in more detail, a method 100 according to a more specific illustrative embodiment of the present invention may yield a movable digital entity with such an adaptive behaviour.

Returning to the method 100, if the starting and end points are not located in the same cell, a sequence of cells among the convex cells is determined from the starting cell to the end cell so as to yield a path for the digital movable entity therebetween (step 112).

Step 112 can be achieved first by constructing a connectivity graph 38, which is obtained by replacing each cell 36 by a node 40 and connecting each pair of passable cell (node) by a line 42. An example of connectivity graph 38 is illustrated in Figure 13 for the example illustrated in Figures 11 and 12. Of course, such a graph 38 is purely virtual and is not actually graphically created.

Then, the resulting graph 38 is searched to find a path between the two nodes 40 representing respectively the starting and end points. Many known techniques can be used to solve such a graph searching problem so as to yield a path between these two corresponding nodes. The path, if it exists is returned as corresponding cells.

For example, a breadth first search (BFS) can be used to search the graph 38. The well known BFS method allows providing the path of lowest cost but can be very expensive in terms of number of nodes explored. A depth first search (DFS), which can also be used, would be significantly less expensive in terms of nodes explored but does not allow to provide the path at the lowest cost. Heuristics can be placed on the DFS to try to improve path quality while maintaining computational efficiency.

It is to be noted that there can be situations where there is no such sequence of cells and the method stop at step 112 with the entity prevented from being able to navigate to the end point.

If there exists such a sequence of cells, intermediary points (not shown) are determined on respective boundary between consecutive cells in the sequence of cells (step 114). In other words, entry/exit points to and from each convex cell are selected.

For example, the centerpoint of each edge can be selected. Of course, other points can alternatively be chosen. The point on the cell edge (cells interface) can be chosen so as to reduce the distance traveled between cells and then further smooth the path.

The digital entity is then moved from the starting point to each selected consecutive intermediary points, finally to the ending points (step 116).

Turning now to Figures 14 to 21, the method 100 will be illustrated with reference to another simple world 44 (see Figure 14) delimitated by walls 46.

As illustrated in Figure 15, a navigation mesh 47 is created and the world 44 is divided in convex cells 48 identified from A to Z for reference purposes.

The starting and end points 50 and 52 are shown in Figure 16. Following step 106 of the method 100, they are found in cells 'Z' and 'J' respectively. Since they are not located in the same cell, the method continues with step 112 with the determination of a sequence of cells between the points 50 and 52, yielding the following sequence as

illustrated in Figure 16: 'Z', 'Y', '0', 'X', 'W', 'V', 'U', 'T', 'H', and 'J'. After determining intermediary points on the respective boundary between consecutive cells in the sequence of cells (step 114), the method continues with the entity moving along the determined path 54 as illustrated in Figure 17. As can be seen in Figure 17, the path can be smooth to yield a more realistic trail.

Since navigation method according to the present invention is used when an entity is to seek a target, the navigation mesh can be dynamically modified at run-time (step 104). For example, as illustrated in Figures 18 and 19, cells can be turned off via blind data to simulate road blocks or congestion due to excess people or physics-driven debris or turned on to simulate a door opening, congestion ending or a passage through a destroyed wall. In Figure 18, former cell 'H' has been turned off, resulting in a first alternative path 56 and in Figure 19, both former cells 'B' and 'H' have been turned off, resulting in a second alternative path 58.

The method 100 also allows taking into consideration the dimension of the movable digital entity, and more specifically its transversal dimension relatively to its moving direction such as its width. The creation of the navigation mesh in step 104 may take into account such characteristic of the movable entity so that the method 100 outputs a path that the entity can pass through. This is illustrated in Figures 20 and 21.

Figure 20 illustrates a path 60 obtained from the method 100 to move a digital entity from a starting point 62 to and end point 64. By

doubling the width of the movable digital entity (illustrated by the sphere 66) in Figure 21 (see sphere 66' in Figure 21), the former cell 'L' is no longer part of the navigation mesh 68 and a new path 70 is provided by the method 100.

The method 100 is not limited to two-dimensional digital world. As it is illustrated in Figures 22 and 23, the method 100 can be used to determine the path between a starting point to an end point and then move a digital movable entity, such as an animated character, between those two points in a three dimensional digital world. Figure 22 illustrates the output of the floor plan generator on a small city part of a simulator, a game or an animation. Figure 23 illustrates the navigation mesh resulting from step 104 from the method 100.

As will now be described with reference to Figures 24-27, the method 100 can be adapted for outdoor and indoor environments.

An outdoor environment typically consists of buildings and open spaces such as market spaces, parks with trees, separated by sidewalks, roads and rivers. Correspondingly, a floor plan generator uses the exterior building walls to cut out holes in the navigation mesh.

In order to allow the movable entity to differentiate between different kinds of reachable space, blind data are used to characterize different part of the reachable space. In some application, blind data can then be associated to the cells of the navigation mesh to specify the differences in navigable surfaces (e.g., roads and sidewalks) and have the

entities navigate accordingly (e.g., keep vehicles on the road and humans on the sidewalks).

Figure 24 illustrates a digital world in the form of a city street. Figure 25 illustrates a navigation mesh obtained from step 104 of method 100. Blind data are used to differentiate between roadway (white cells) and sidewalk (grey cells).

As illustrated in Figure 26, an indoor environment is typically multi-layer and consists of floors divided into rooms via inner walls and doors; and connected by stairways. Correspondingly, a floor plan generator calculates the navigation mesh for each floor using the walls as barriers and then links the navigation surfaces by the cells that correspond to the stairways. This results in a 3D navigation mesh in which cells may be on top on one another. Path finding is now modified to determine which surface cell the digital movable entity is on rather than which cell the character is in. In brief, when a three-dimensional world is defined by a plurality of levels, a navigation mesh is created for each of the levels and two consecutive navigation meshes are interconnected by connecting cells.

Returning to Figure 7 and more specifically to step 116, it is reminded that within a convex cell, the method 100 allows the movable entity to move from a predetermined intermediary point to the next in a straight line. Therefore, the only things that can prevent the entity from going in a straight line may be dynamic obstacles. To cope with this situation, the movable entity may be provided with sensors. Before

describing in more details a dynamic collision avoidance method according to a method from the present invention, the concept of sensor and other relevant concepts such as data information, commands, decisions and decision trees will first be described briefly. It is to be noted however that neither the collision avoidance method according to the present invention nor the navigation method 100 are to be construed as being limited to a specific embodiment of sensors or decision rules for the digital movable entity, etc.

Data Information

An entity's data information can be thought of as its internal memory. Each datum is an element of information stored in the entity's internal memory. For example, a datum could hold information such as whether or not an enemy is seen or who is the weakest ally. A Datum can also be used as a state variable for an IE.

Data are written to by an entity's Sensors, or by Commands within a Decision Tree. The Datum's value is used by the Decision Tree to activate and deactivate behaviours and animations, or to test the entity's state. Sensors and Decision trees will be described hereinbelow in more detail.

Sensors

Entities use sensors to gain information about the world. A sensor will store its sensed information in a datum belonging to the entity.

A parameter can be used to trigger the activation of a sensor. If a sensor is set off, it will be ignored by the solver and will not store information in any datum.

An example of sensor will now be described in more detail. Of course, it is believed to be within the reach of a person skilled in the art to provide additional or alternate sensors depending on the application.

Vision Sensor

The vision sensor is the eyes and ears of a character and allows the character to sense other physical objects or movable entities in the virtual world, which can be autonomous or non-autonomous characters, barriers, and waypoints, for example.

The following parameters allow, for example, defining the vision sensor:

Parameter	Description
Visibility Distance	This parameter allows defining the maximum distance from the IE that it can sense other objects *i.e.* how far can the IE see. The visibility distance is the external distance between the IE, *i.e.* the distance between the outsides of the bounding spheres of the IEs.
Visibility Angles	This parameter allows defining the following four angles: *Visibility Right Angle*, *Visibility Left Angle*, *Visibility Up Angle*, and *Visibility Down Angle*, specify the field of view of the visibility sensor measured in degrees. Any object outside the frustum defined by these angles will

Parameter	Description
	be ignored.
Can See Through Opaque Barriers	If this parameter is set to *off*, then the sensor will not sense objects behind opaque barriers.
Object Type Filter	This parameter allows defining the type of objects this sensor will look for. The options are: All Objects, Barriers, Way Points, or IEs. For example, if Barriers is chosen then the sensor will only find barriers.
Object Filter	This parameter allows defining the objects this sensor will look for. If this is set to a group, then the sensor will only look for objects in the selected group. If this is set to a path, then the sensor will only look for waypoints on the path. If this is set to a specific object (*e.g.* a character, a waypoint, or a barrier), then the sensor will ignore all other objects in the world.
Evaluation Function	This parameter allows defining the evaluation function assigns a value to each sensed object. The value of the object, in conjunction with the *Min Max* attribute, is used to determine the "best" object of all the ones sensed. The possible values are: - Any: this chooses the first object sensed. This is the most efficient value of the Evaluation Function. This value could possibly choose the same object every time. If you want a randomly selected object, set the value of the Evaluation Function to "Random". - Distance: this chooses an object based on its distance from the character. If the *Min Max* attribute is set to minimum, the nearest object to the IE is chosen. If the *Min Max* attribute is set to maximum, the furthest object (within the visibility distance) to the IE is chosen. - Random: this randomly chooses an object.
Min Max	This parameter allows defining whether the object with the minimum or maximum value is considered the "best" object.
Is Any Object Seen Datum	This parameter allows defining the datum that will be used to store whether or not any object was seen, *i.e.*

Parameter	Description
	did the IE see what it was looking for.
Best Object Datum	This parameter allows defining the datum that will be used to store which "best" object that was sensed, *i.e.* what exactly did the IE see.

Commands, Decisions, and Decision Trees

Decision trees are used to process the data information gathered using sensors.

A command is used to activate a behaviour or an animation, or to modify an IE's internal memory.

Commands are invoked by decisions. A single Decision includes a conditional expression and a list of commands to invoke.

A decision tree includes a root decision node, which can own child decision nodes. Each of those children may in turn own children of their own, each of which may own more children, etc.

A parameter indicative of whether or not the decision tree is to be evaluated can be used in defining the decision tree.

Whenever the command corresponds to activating an animation and a transition is defined between the current animation and the new one, then that transition is first activated.

Similarly, whenever the command corresponds to activating a behaviour, a blend time can be provided between the current animation and the new one.

Moreover, whenever the command corresponds to activating a behaviour, the target is changed to the object specified by a datum.

Dynamic collision avoidance

It will now be described substeps of both the steps 110 and 116 from the method 100 allowing the entity to avoid hitting any moving obstacle (e.g., another character or a physics object) as it travels along its path as determined through steps 102-114.

Just as path finding can be seen as a two-step process of i) determining the path and then ii) following it; obstacle avoidance can also be seen as a two-step method 200, which is illustrated in Figure 28, of:

202 - assessing threats of potential collisions between the movable digital entity and a moving obstacle, and

204 – if there is such a threat, the movable digital entity responding accordingly by adopting a strategy to avoid the moving obstacle

Collision threat assessment

In step 202, each entity 72 uses its sensor (see Figure 29) to detect what potential obstacles are in its vicinity and decide which of those

obstacles poses the greatest threat of collision. As illustrated in Figure 30, the sensor is configured as a field of view 74 around the entity 72, characterized by a depth of field, defining how far the entity can see. The field of view of each movable entity can be defined as a pie (or a sphere depending on the application) surrounding the entity.

If there are no obstacles, then the entity's field of view 74 around itself would be completely free and the pie is complete. However, as illustrated in Figure 30, each obstacle removes a piece of this pie. The size of the piece removed depends on the obstacle's size and its distance from the entity.

Unobstructed sections of the pie are will be referred to herein as holes 76-78 (see Figure 31). The entity 72 searches for the best hole to continue through.

The best hole can be determined in several ways. The typical way is as follows:

- The holes 76-78 are sorted in order of increasing radial distance from the desired direction of the entity 72;
- The first hole that is large enough for the entity to pass through is chosen;
- If there is no hole, the agent is completely blocked and will stop moving.

Depending on the chosen hole, the movable entity can move into that hole by either turning or reversing.

Depending on the application of the simulator and on the characteristics of the entity, the obstacles can be characterized as having different avoidance importance. For example, a Hummvee may consider avoiding other vehicles as its highest priority, pedestrians as a secondary priority and small animals such as dogs as a very low priority; and a civilian pedestrian may consider vehicles as its highest priority and other pedestrians as a secondary priority. Of course, extreme situations such as a riot may dynamically change these priorities. Thus, different obstacle groups have different avoidance constraints. The most basic constraint is assigning an obstacle to be a threat. Accordingly, for each obstacle group, there is an associated awareness radius. As the character moves through its world, its sensor sweeps around it, for every obstacle detected in its sweep that is within its awareness radius, it is flagged as a potential collision threat.

For each perceived threat, different kinds of collision threat states can be assigned, including:

- stationary: the obstacle is not moving but is in the movable entity's way;
- incoming: the obstacle is coming towards the movable entity;
- outgoing: the obstacle is going away from the movable entity but the entity will rear-end it; and

- sideswiping: the obstacle is expected to hit the movable entity from the side.

Avoidance strategies

For each collision threat, there are two kinds of avoidance strategies (step 204):

- circumvention: try avoiding the collision by going around the obstacle; and
- queuing: try avoiding the collision by slowing down (and potentially stopping) until the obstacle exits the collision path.

It is to be noted that both above-described avoidance strategies are heuristics and as such neither is guaranteed to work in all circumstances.

Figures 32A-32C, 33A-33C, 34A-34C, and 35A-35C illustrate three examples of collision threats (Figures 32A, 33A, 34A, and 35A), each with both corresponding avoidance strategies. Figures 32A-35C show how circumvention is useful for going around stationary obstacles and getting out of the way of incoming obstacles. However, it can cause a lot of jostling on outgoing obstacles. Nonetheless, circumvention has the advantage of minimizing gridlock and eventually finding a way around.

Queuing always works well on outgoing obstacles and incoming obstacles (provided they are circumventing). However, it has

been found that queuing incoming obstacles and stationary obstacles can cause gridlock. In most cases, the most effective way to avoid gridlock is to use decision logic to change strategies on the fly.

Group-based movement modifiers (formations)

It is a well-observed fact that herd-based animals including birds, fish and humans move differently when in groups. This movement behaviour is commonly called flocking. With humans, group based movement is very varied from the loose groups of Somali militiamen running through a twisty city street to rigid formations of marching soldiers on parade to coordinated cover and sweep routines of modern Delta Force operators. In each case, the group's "collective conscience" influences the movement of each navigating individual in the group.

The group movement modifier that is most rapidly identified with computer graphic artificial intelligence is flocking, made famous by Reynolds [Reynolds, 1987] who modelled flocks of birds called boids as super particles. Reynolds identified three basic elements to flocking:

- alignment: the tendency of group members to harmonize their motion by aligning themselves in the same direction with the same speed;
- separation: the tendency of group members to maintain a certain amount of space between them; and
- joining: the tendency of group members to maintain a certain proximity with one another.

Considering now a group of friends walking down the street, slower members of the group will speed up to catch up to the others, the fastest members (assuming they are polite) will slow down slightly to allow the stragglers to catch up. Depending on the cultural background of the group more or less space is required or tolerated between the friends (cf. urban dwellers to rural dwellers).

Figures 36A-36F shows the effects of different flocking strategies on a group of five characters following a leader character.

Such group-based modifier can be used to yield a more natural effect when the method 100 is used to simultaneously move a group of entities in a digital world between starting and end points.

Even though the method and system for moving a digital entity on-screen from starting to end points in a digital world has been described as being included in a specific illustrative embodiment of a 3D application, it can be included in any 3D application requiring the autonomous displacement on-screen of image element. For example, a navigation method according to the present invention can be used to move digital entity not characterized by behaviours such as described hereinabove.

A navigation method according to present invention can be used to navigate any number of entities and is not limited to any type or configuration of digital world. The present method and system can be

used to plan the displacement of a movable object or entity in a virtual world without further movement of the object or entity.

Although the present invention has been described hereinabove by way of preferred embodiments thereof, it can be modified without departing from the spirit and nature of the subject invention, as defined in the appended claims.

WHAT IS CLAIMED IS:

1. A method in a computer system for moving at least one digital entity on-screen from starting to end points in a digital world, comprising:

i) providing respective positions of obstacles for the at least one movable digital entity in the digital world; defining at least portions of said digital world without obstacles as reachable space for the at least one movable digital entity;

ii) creating a navigation mesh for the at least one movable digital entity by dividing said reachable space into at least one convex cell;

iii) locating a start cell and an end cell among said at least one convex cell including respectively the start and end points; and

iv) verifying whether said starting cell corresponds to said end cell; if said starting cell corresponds to said end cell, then: iv)a) moving the at least one movable digital entity from the starting point to the end point; if said starting cell does not correspond to said end cell, then iv)b) i) determining a sequence of cells among said at least one convex cell from said starting cell to said end cell, and iv)b)ii)determining at least one intermediary point located on a respective boundary between consecutive cells in said sequence of cells, and iv)b)iii) moving the at least one movable digital entity from the starting point to each consecutive said at least one intermediary point to said end point.

2. A method as recited in claim 1, wherein said obstacles being dynamic obstacles, yielding changes in said digital world; wherein in

ii) said navigation mesh is dynamically created to cope for said changes in said digital world caused by said dynamic obstacles.

3. A method as recited in claim 1, wherein iv)b)i) determining a sequence of cells among said at least one convex cell from said starting cell to said end cell includes constructing a connectivity graph and searching said connectivity graph for a path between said starting cell to said end cell.

4. A method as recited in claim 2, wherein a breadth first search (BFS) or depth first search (DFS) is used in searching said connectivity graph.

5. A method as recited in claim 1, wherein in iv)b)ii) at least one intermediary point located on a respective boundary between consecutive cells in said sequence of cells includes a centerpoint.

6. A method as recited in claim 1, wherein in iv)b)ii) at least one intermediary point located on a respective boundary between consecutive cells in said sequence of cells is selected so as to improve the quality of motion.

7. A method as recited in claim 6, wherein said sequence of cells is selected so as to reduce a distance travelled within at least one of said consecutive cells.

8. A method as recited in claim 1, wherein said digital world further includes at least one moving obstacle within said at least one of said convex cell; wherein at least one of said iv)a) moving the at least one movable digital entity from the starting point to the end point and iv)b)iii) moving the at least one movable digital entity from the starting point to each consecutive said at least one intermediary point to said end point further including a) assessing whether there is a collision threat between said at least one moving obstacle and said at least one digital movable entity, and b) said at least one digital movable entity adopting a strategy to avoid said at least one moving obstacle when there is a collision threat.

9. A method as recited in claim 8, wherein a) assessing whether there is a collision threat between said at least one moving obstacle and said at least one digital movable entity includes using a sensor to detect said at least one moving obstacle.

10. A method as recited in claim 9, wherein using said sensor yields a field of view between said at least one movable digital entity characterized by a depth of field.

11. A method as recited in claim 10, wherein using a sensor to detect said at least one moving obstacle yields a hole in said field of view; said hole characterizing said at least one moving obstacle.

12. A method as recited in claim 11, wherein said hole having at least one characteristic; said at least one digital movable entity

prioritizes its strategy to avoid said at least one moving obstacle according to said at least one characteristic of said hole.

13. A method as recited in claim 12, wherein said at least one characteristic of said hole includes a radial distance and a width.

14. A method as recited in claim 13, wherein said strategy to avoid said at least one moving obstacle is circumvention or queuing.

15. A method as recited in claim 1, yielding a path between the staring and end points; the method further comprising smoothing said path so as to yield a more realistic path.

16. A method as recited in claim 15, wherein said at least one movable digital entity includes a plurality of movable digital entity all adopting a flocking strategy to follow said path.

17. A method as recited in claim 16, wherein said flocking strategy includes at least one of following, alignment, separation and joining.

18. A method as recited in claim 1, wherein said at least one movable digital entity includes a plurality of movable digital entity.

19. A method as recited in claim 1, wherein at least one of said convex cell being characterized by a blind datum allowing differentiating between types of navigational cells.

20. A method as recited in claim 19, wherein the digital world further comprises state changing entity; said state changing entity being operative onto said blind datum.

21. A method as recited in claim 1, wherein said navigation mesh is created manually or automatically.

22. A method as recited in claim 1, wherein said digital world is defined by a rendering or physics geometry; said navigation mesh being created using said rendering or physics geometry.

23. A method as recited in claim 1, wherein said digital world is a two-dimensional world.

24. A method as recited in claim 1, wherein said digital world is a three-dimensional world.

25. A method as recited in claim 24, wherein said three-dimensional world is defined by a plurality of levels; in ii) a level navigation mesh is created for each of said plurality of levels; consecutive level navigation meshes being interconnected by connecting cells.

26. A method as recited in claim 1, wherein said at least one movable digital entity is autonomous or non-autonomous.

27. A system for moving a digital entity on-screen from starting to end points in a digital world, comprising:

a world database for storing information about the digital world and for providing respective positions of obstacles for the movable digital entity in the digital world;

a navigation module

i) for defining at least portions of said digital world without obstacles as reachable space for said movable digital entity;

ii) for creating a navigation mesh for said movable digital entity by dividing said reachable space into at least one convex cell;

iii) for locating a start cell and an end cell among said at least one convex cell including respectively the start and end points; and

iv) for verifying whether said starting cell corresponds to said end cell; and

if said starting cell does not correspond to said end cell, for further

v) determining a sequence of cells among said at least one convex cell from said starting cell to said end cell, and vi) determining at least one intermediary point located on a respective boundary between consecutive cells in said sequence of cells;

and

a simulator coupled to said navigation module and to said world database for moving the digital entity on-screen via an image generator coupled to said simulator from the starting point to the end point if said starting cell corresponds to said end cell as verified in iv); or for moving the digital entity from the starting point to each consecutive said at

least one intermediary point to said end point; if said starting cell does not correspond to said end cell.

28. A system as recited in claim 27, wherein said world database being further for storing information about at least one moving obstacle within said at least one of said convex cell; said navigation module being further for a) assessing whether there is a collision threat between said at least one moving obstacle and said at least one digital movable entity, and b) said at least one digital movable entity adopting a strategy to avoid said at least one moving obstacle when there is a collision threat.

29. A system as recited in claim 27, wherein said navigation module is part of said simulator.

30. A system as recited in claim 27, further comprising a decision-making module coupled to the simulator for dynamic adjustment of the navigation unit following said digital world acting on said at least one digital movable entity.

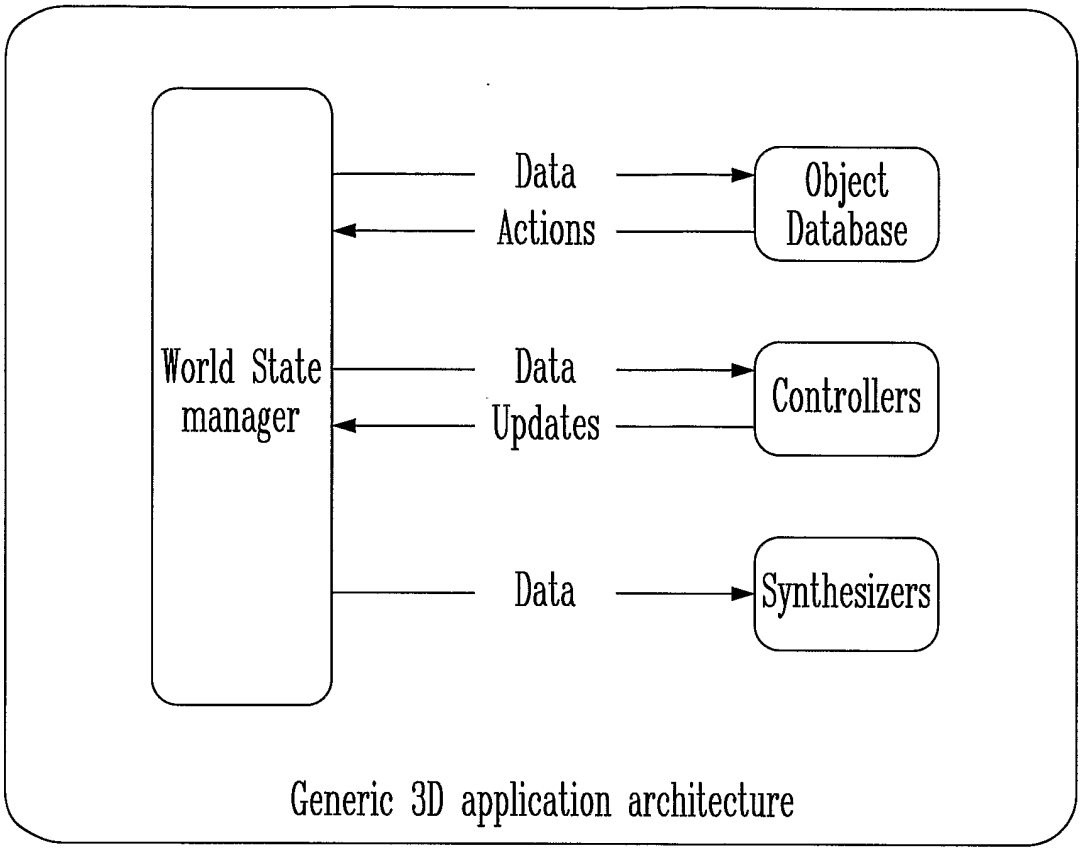


Fig. 1 (Prior Art)

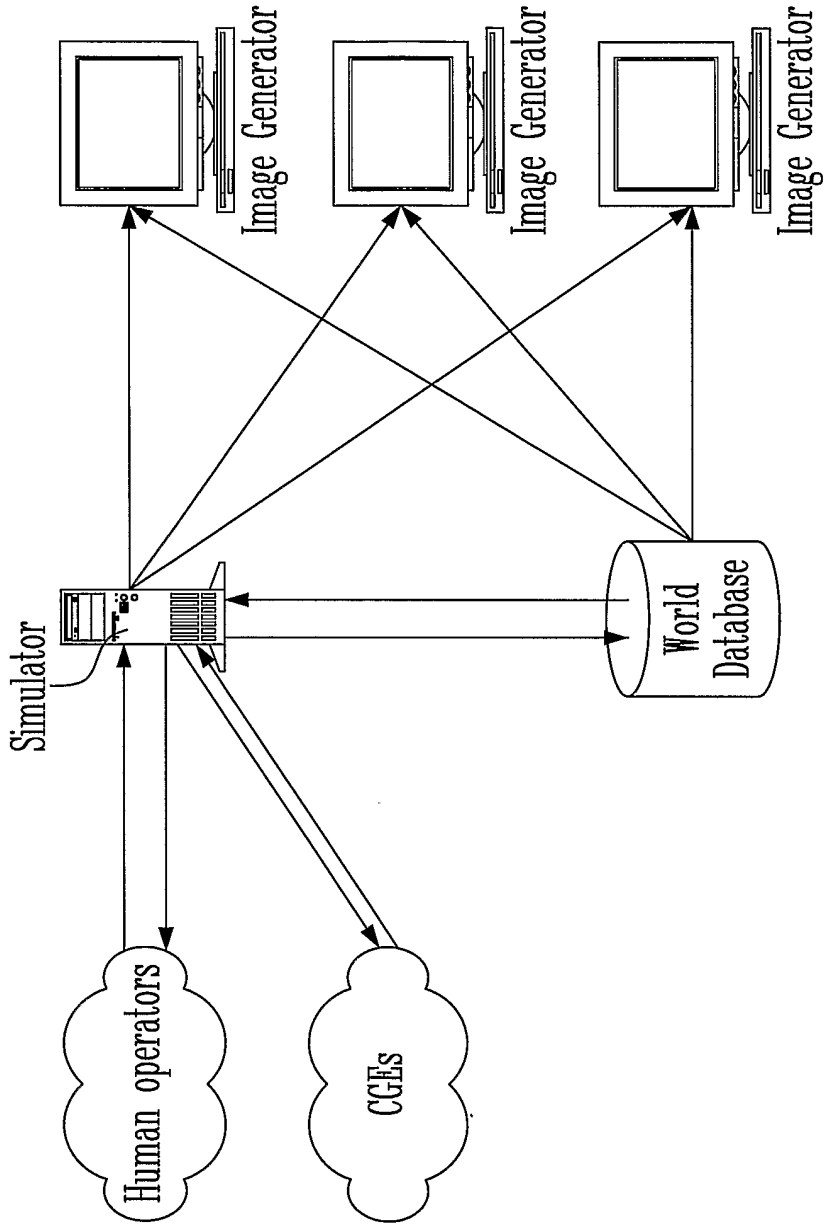


Fig. 2 (Prior Art)

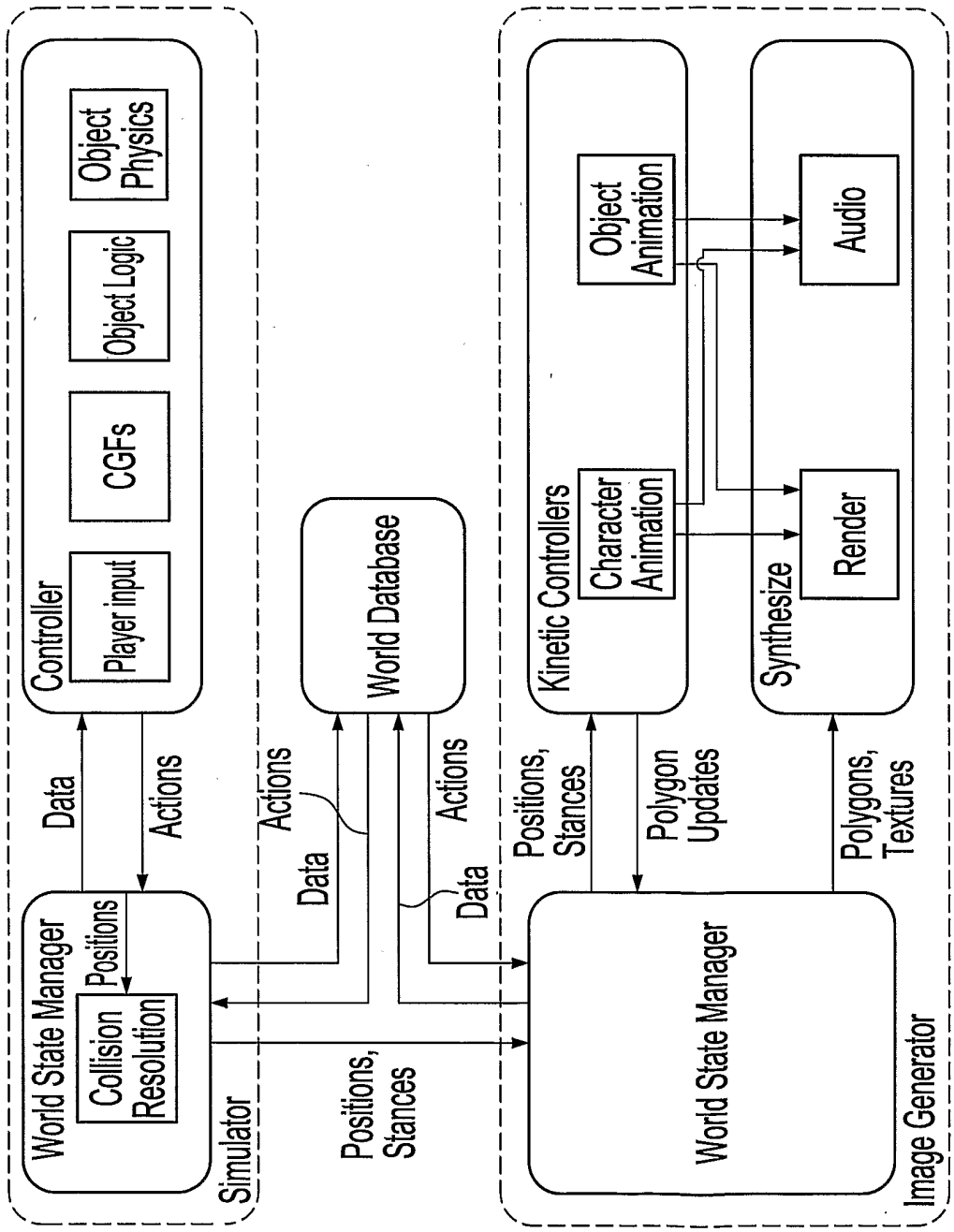


Fig. 3 (Prior Art)

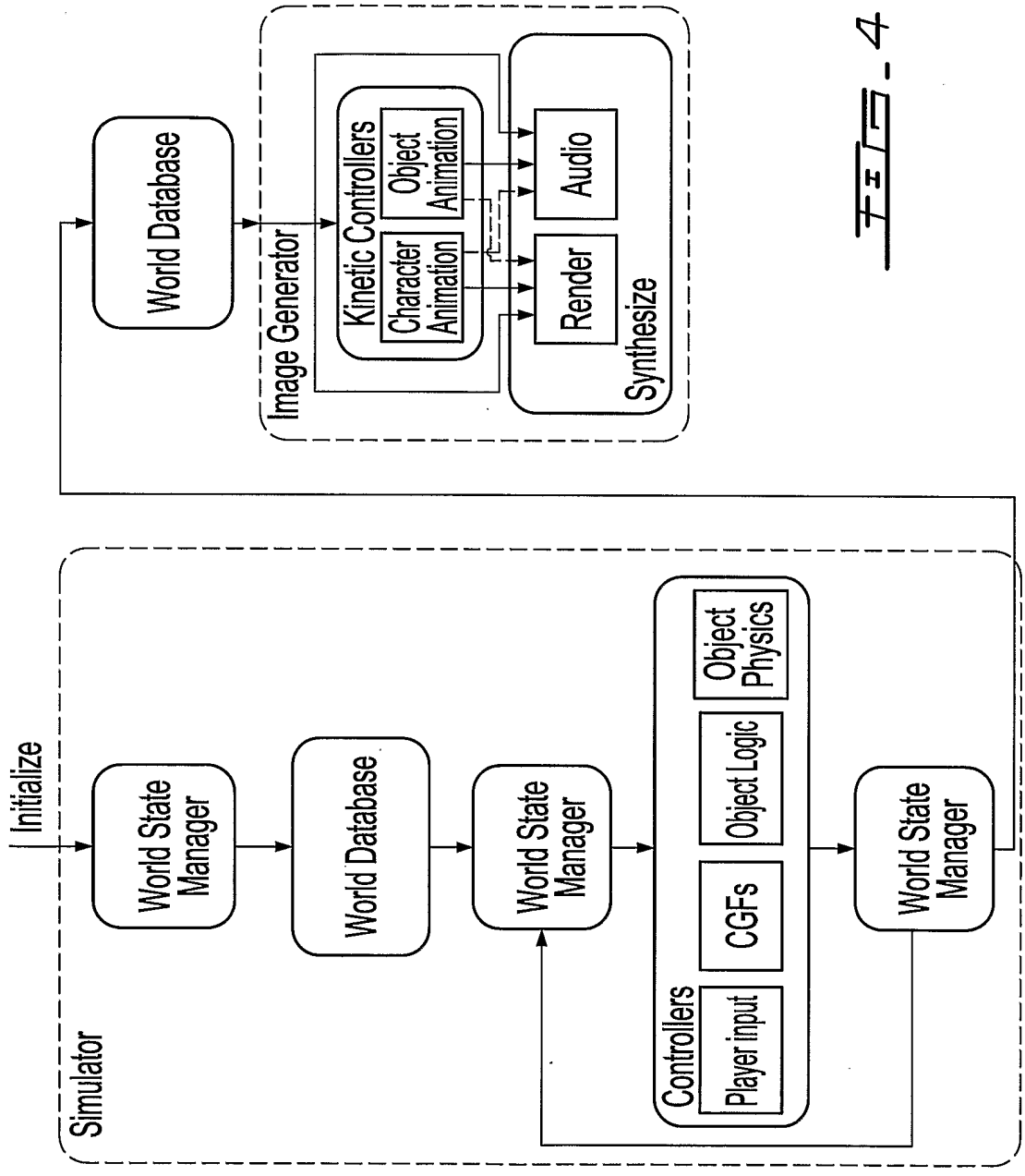


Fig. 4 (Prior Art)

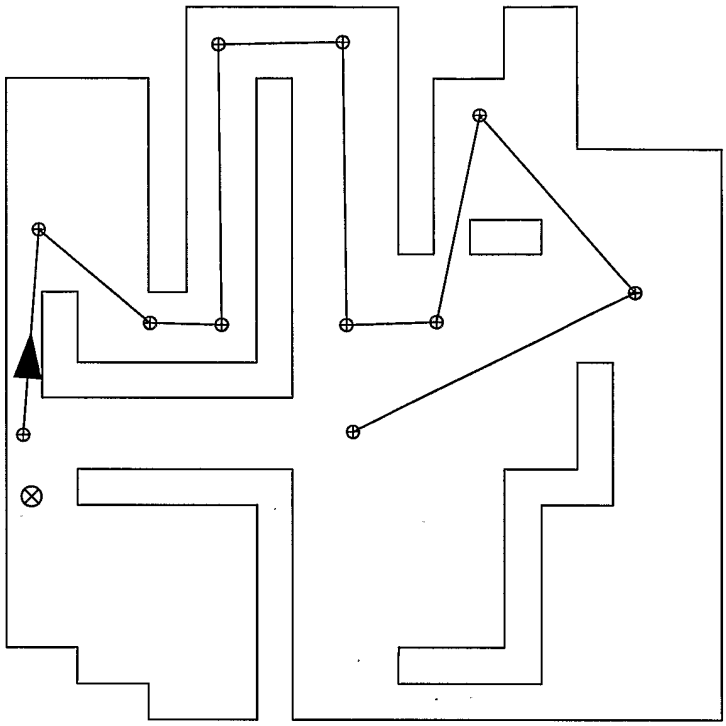


FIG_5 (Prior Art)

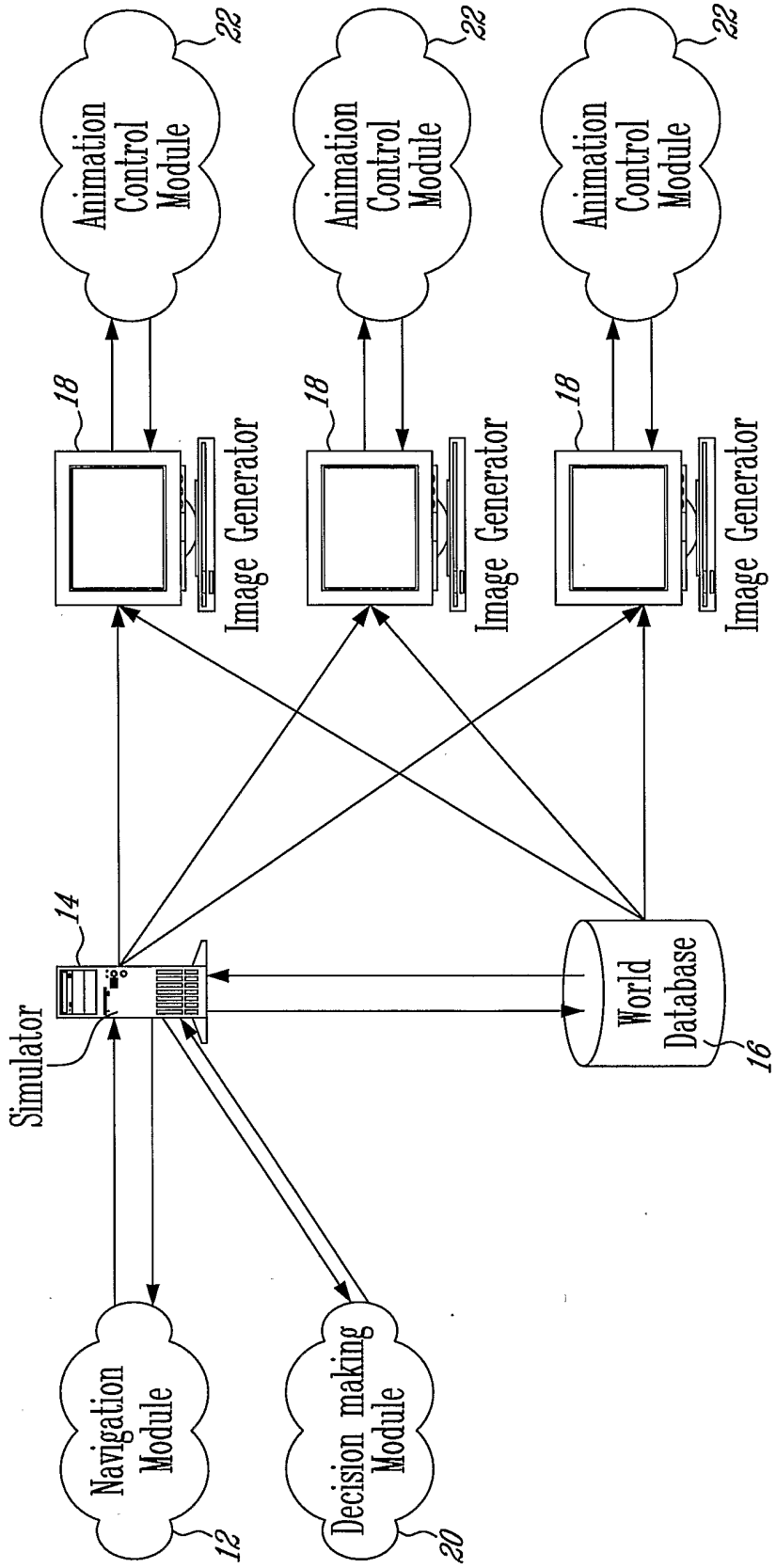

Animation Control Module
22
Image Generator
18
10
Simulator
14
World Database
16
Navigation Module
12
Decision making Module
20

FIG. 6

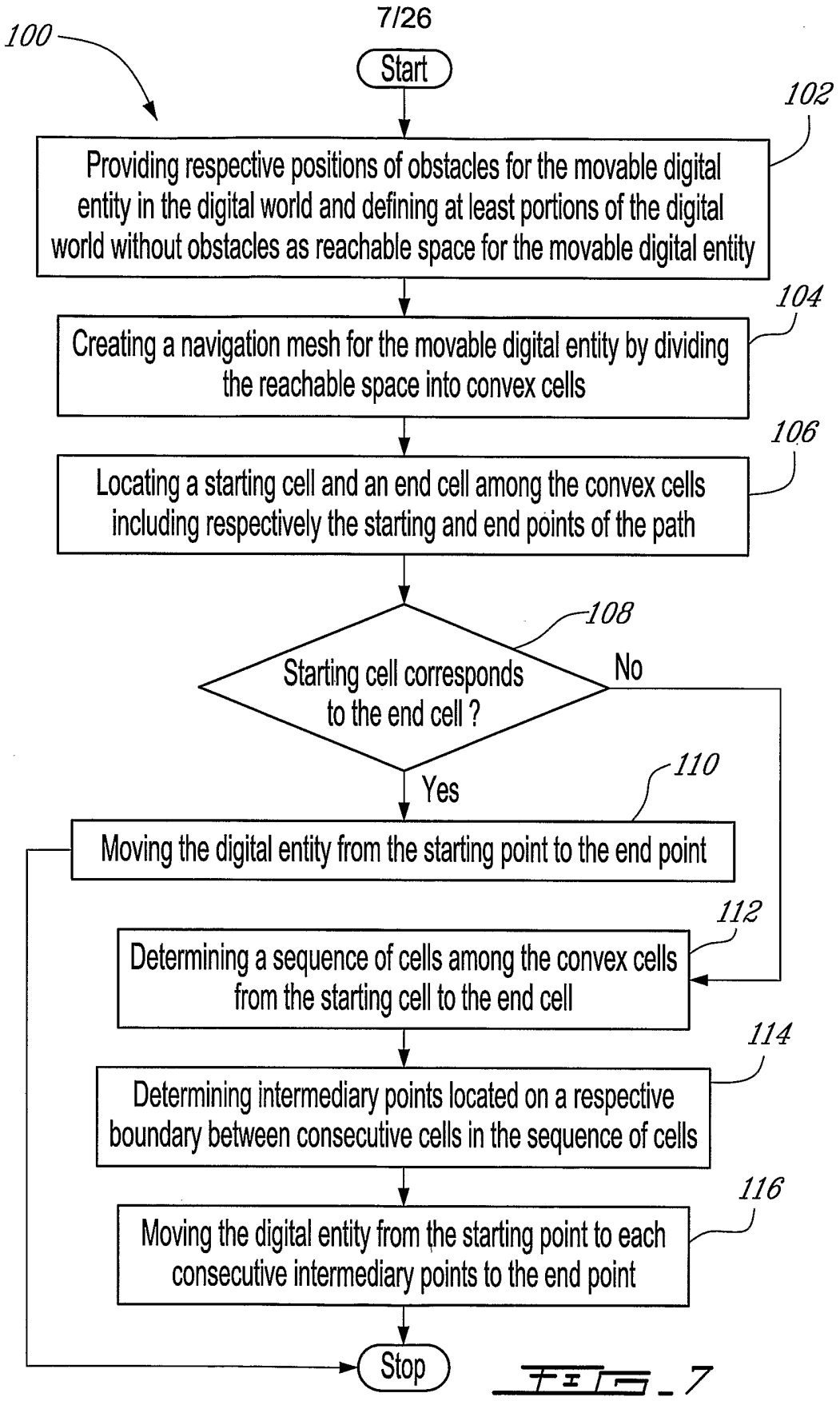
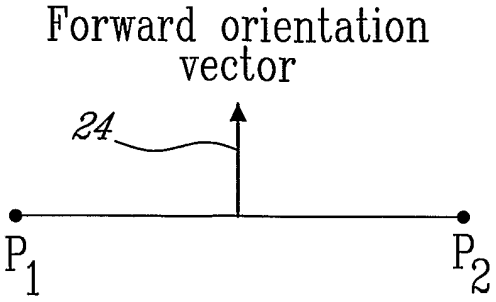
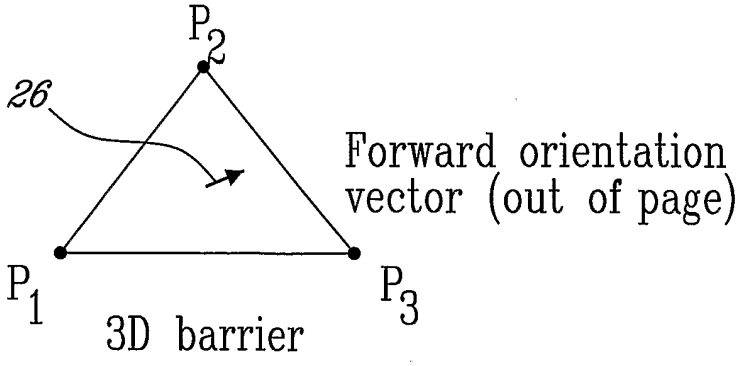

100
Start
102
Providing respective positions of obstacles for the movable digital entity in the digital world and defining at least portions of the digital world without obstacles as reachable space for the movable digital entity
104
Creating a navigation mesh for the movable digital entity by dividing the reachable space into convex cells
106
Locating a starting cell and an end cell among the convex cells including respectively the starting and end points of the path
108
Starting cell corresponds to the end cell ?
No
Yes
110
Moving the digital entity from the starting point to the end point
112
Determining a sequence of cells among the convex cells from the starting cell to the end cell
114
Determining intermediary points located on a respective boundary between consecutive cells in the sequence of cells
116
Moving the digital entity from the starting point to each consecutive intermediary points to the end point
Stop

Fig. 7



2D barrier

Fig. 8



3D barrier

Fig. 9

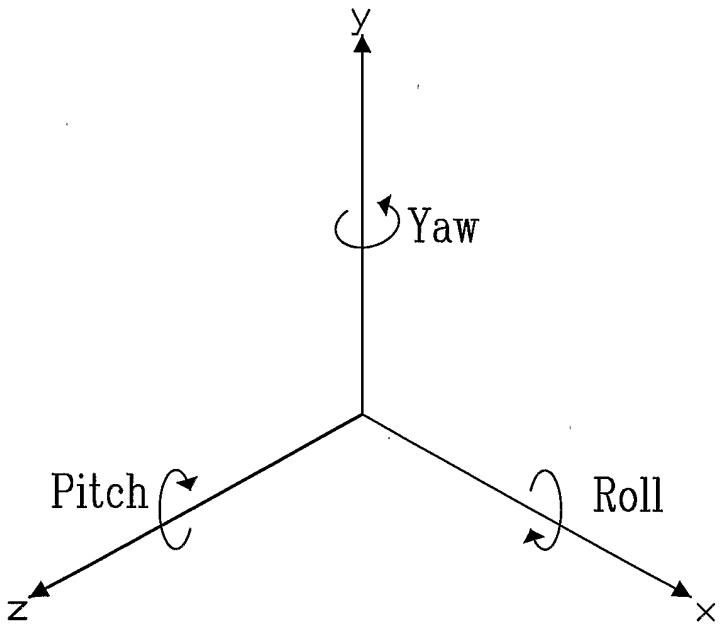


FIG. 10

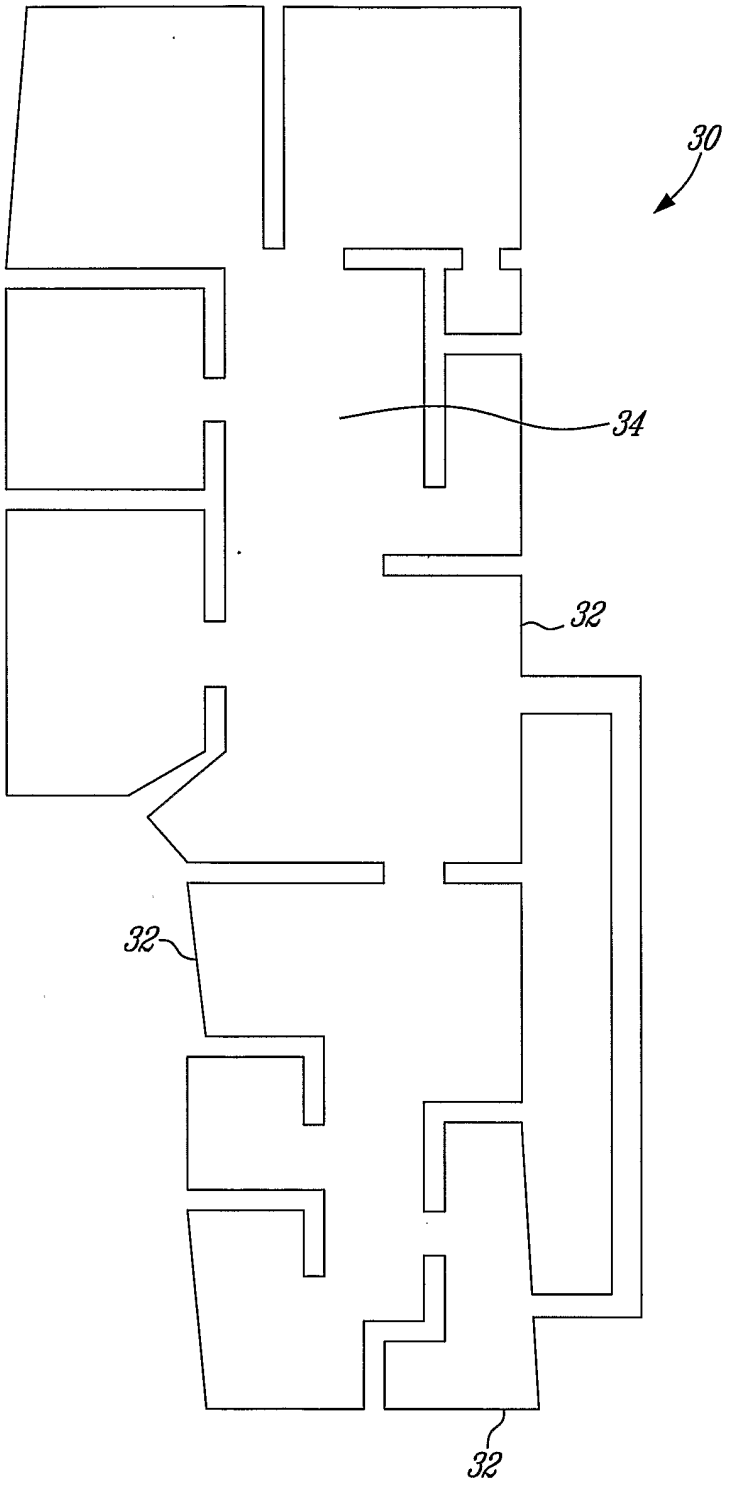


Fig. 11

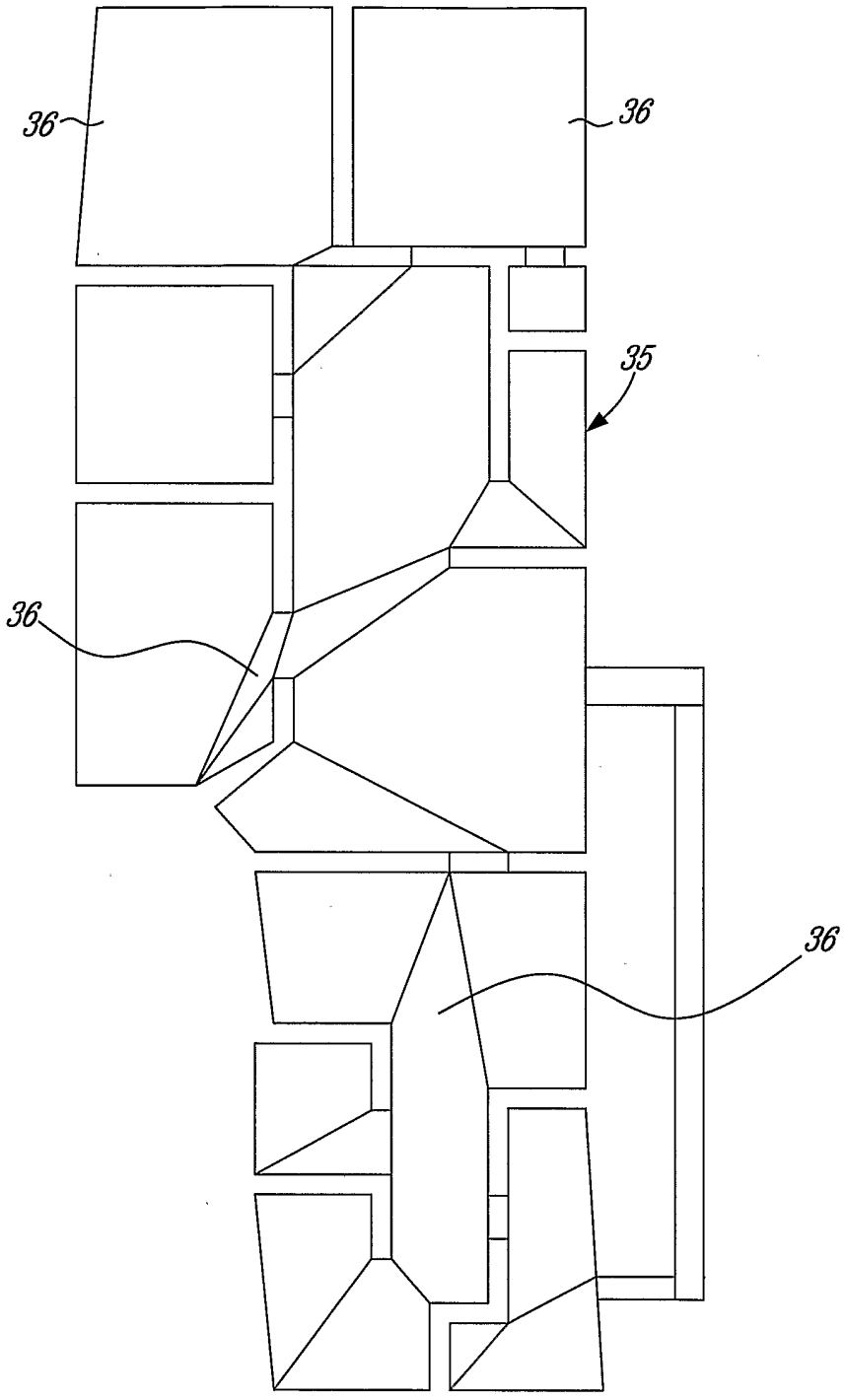


Fig. 12

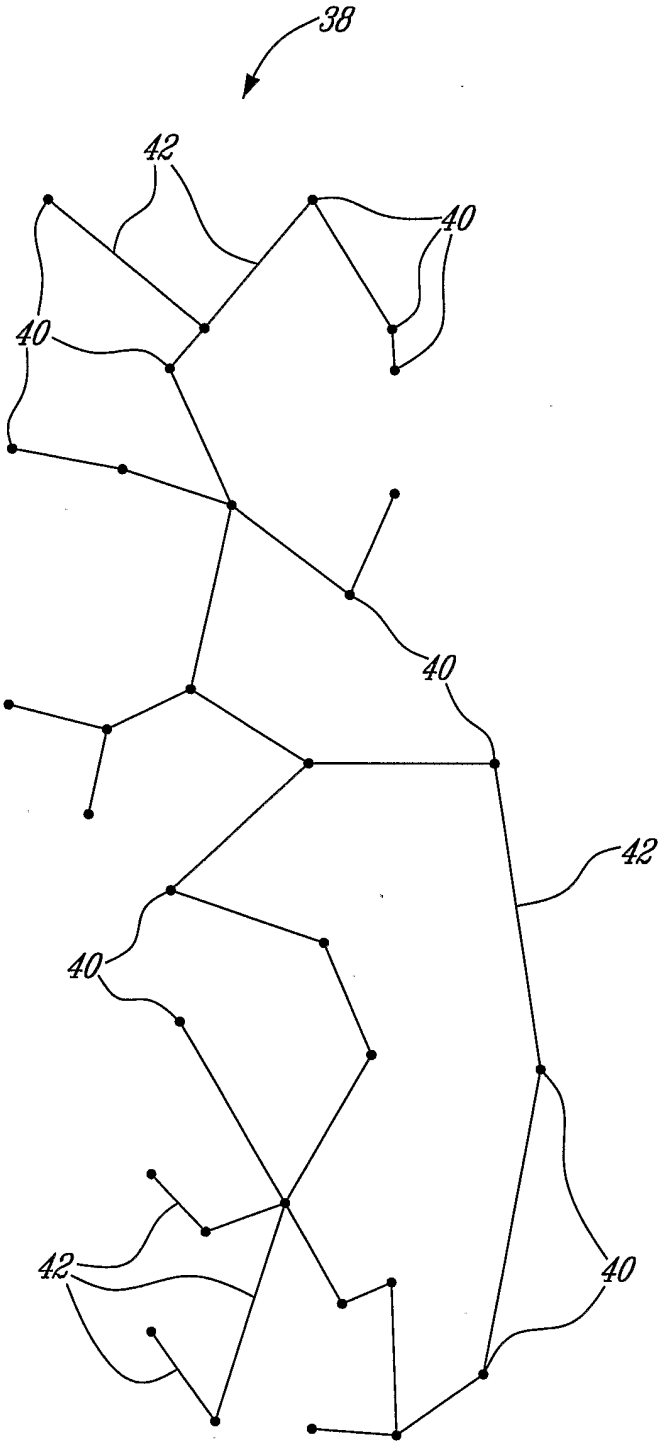


Fig. 13

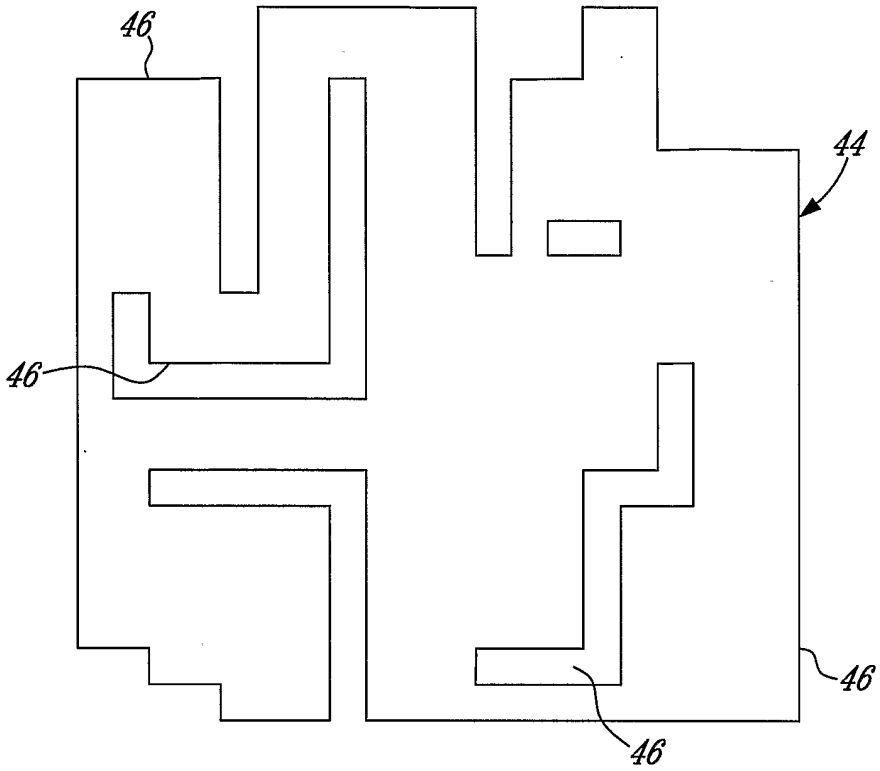


FIG. 14

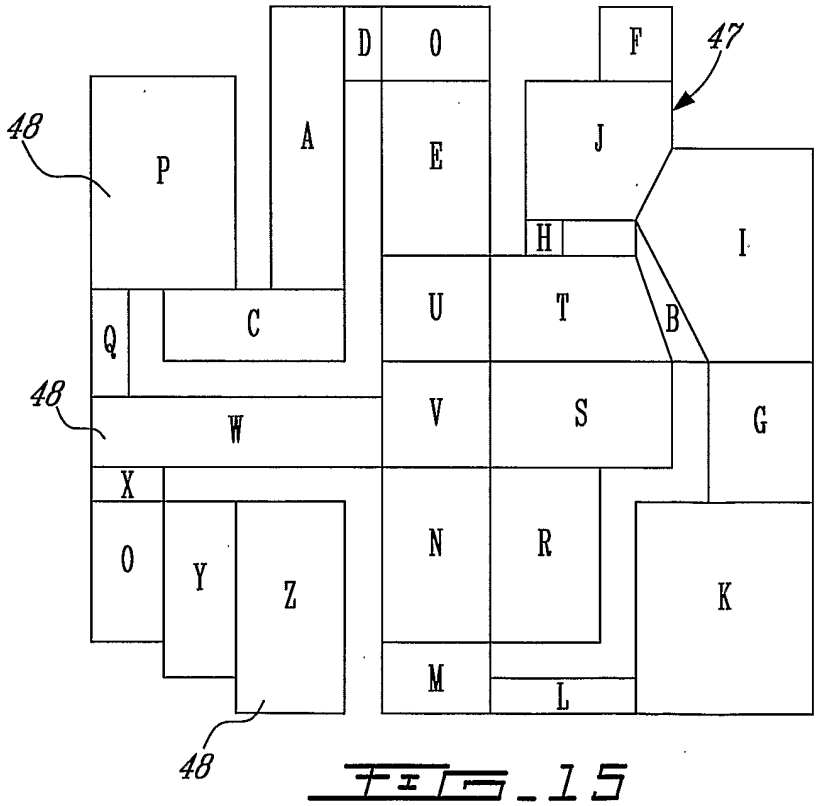


FIG. 15

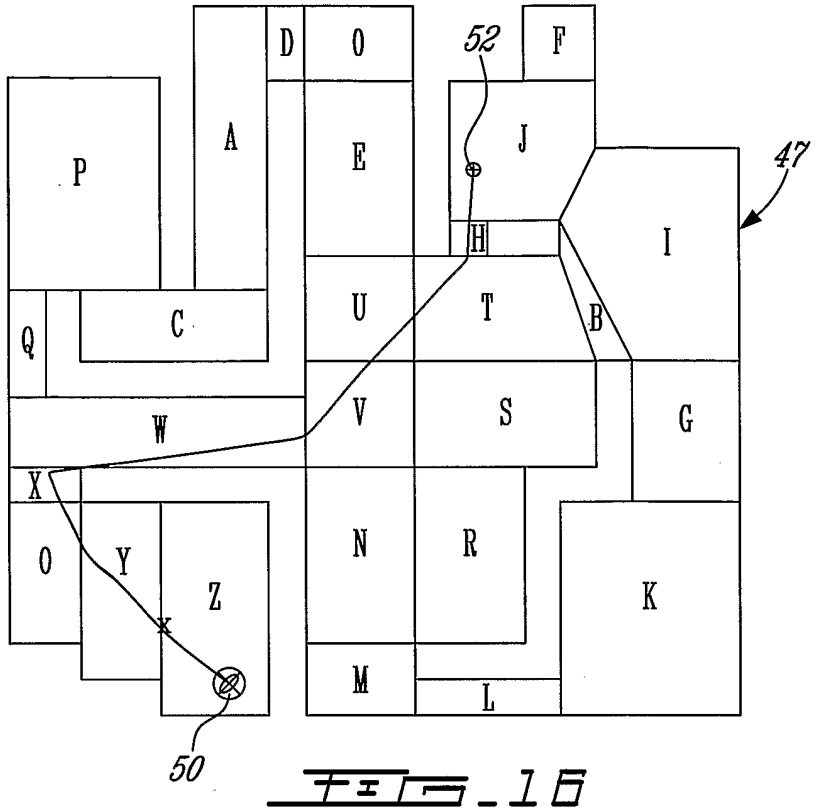

D
O
52
F
A
E
J
P
47
H
I
U
T
B
Q
C
W
V
S
G
X
N
R
O
Y
Z
K
M
L
50

FIG_16

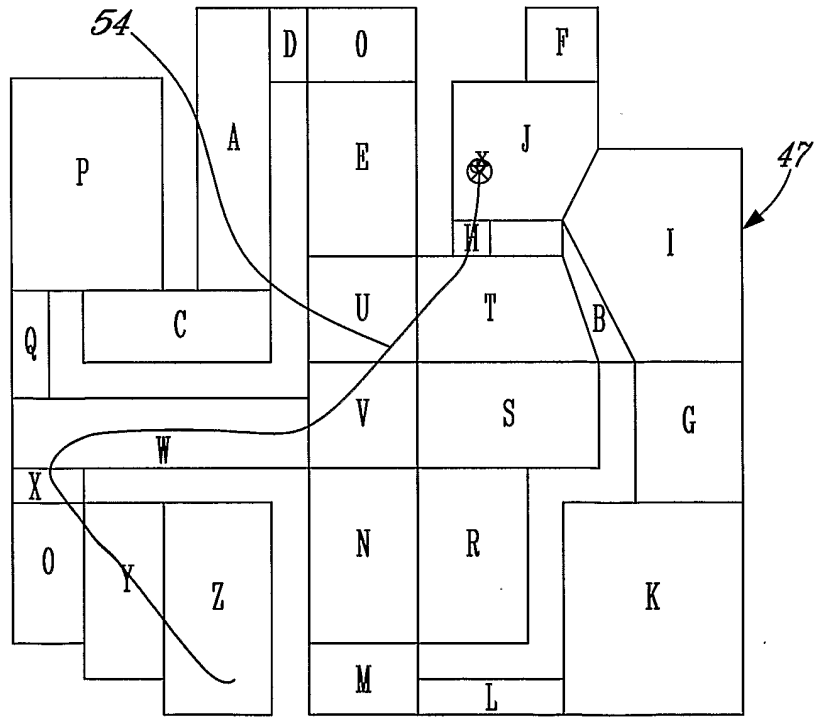

54
D
O
F
A
E
J
P
47
H
I
U
T
B
Q
C
W
V
S
G
X
N
R
O
Y
Z
K
M
L

FIG_17

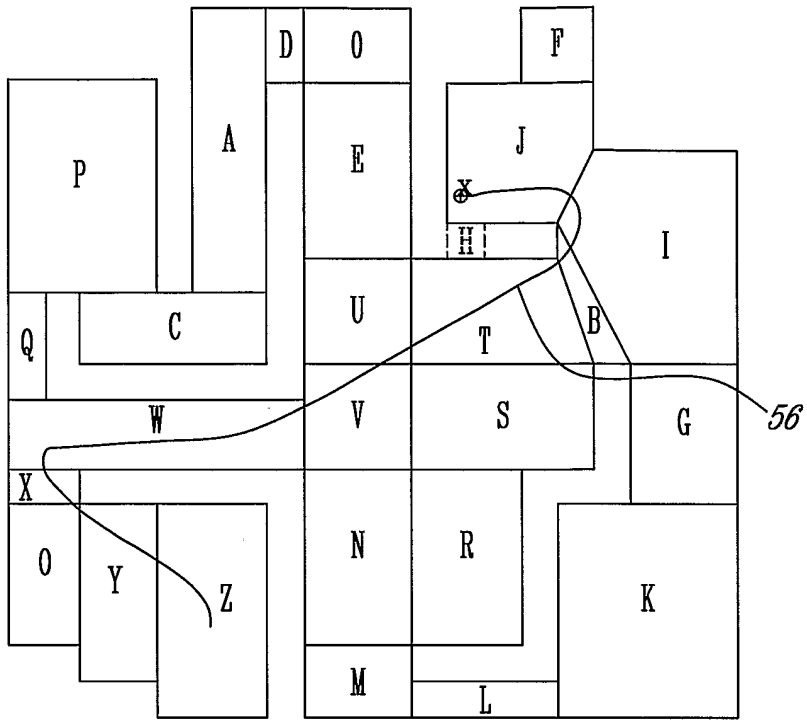

D
O
F
P
A
E
J
X
H
I
U
T
B
Q
C
W
V
S
G
56
X
O
Y
Z
N
R
K
M
L

FIG_18

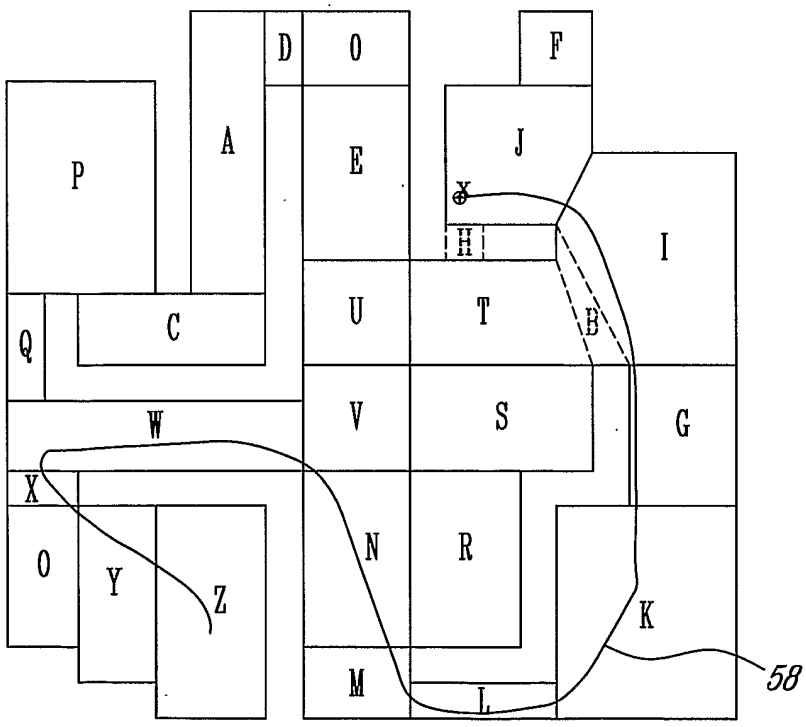

D
O
F
P
A
E
J
X
H
I
U
T
B
Q
C
W
V
S
G
X
O
Y
Z
N
R
K
58
M
L

FIG_19

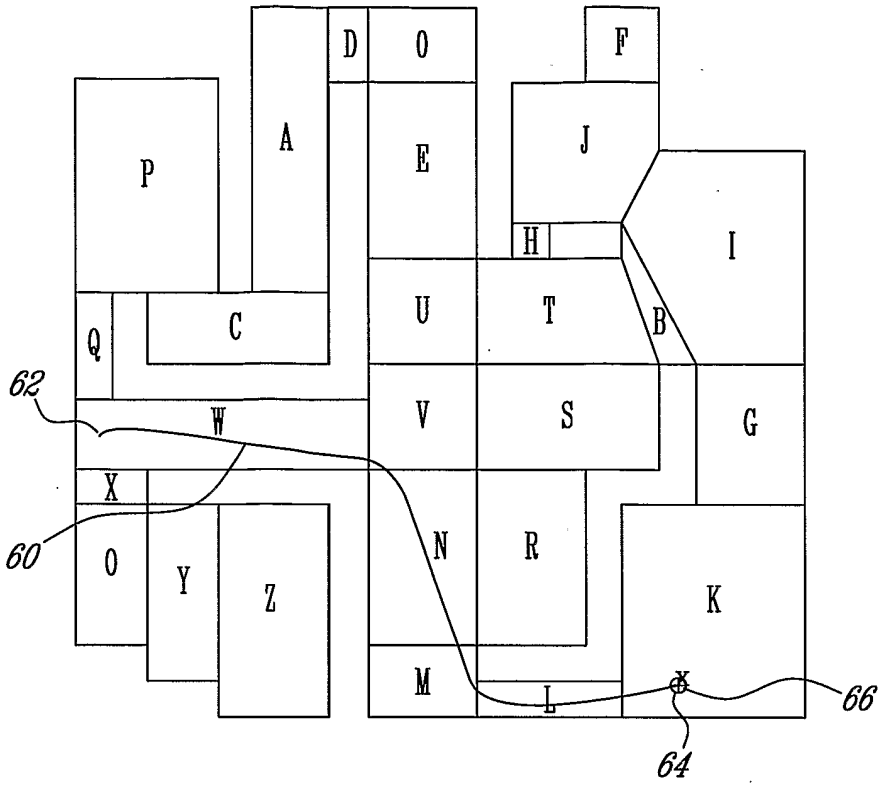

D
O
F
P
A
E
J
H
I
U
T
B
Q
C
62
W
V
S
G
X
60
N
R
O
Y
Z
K
M
L
66
64

FIG. 20

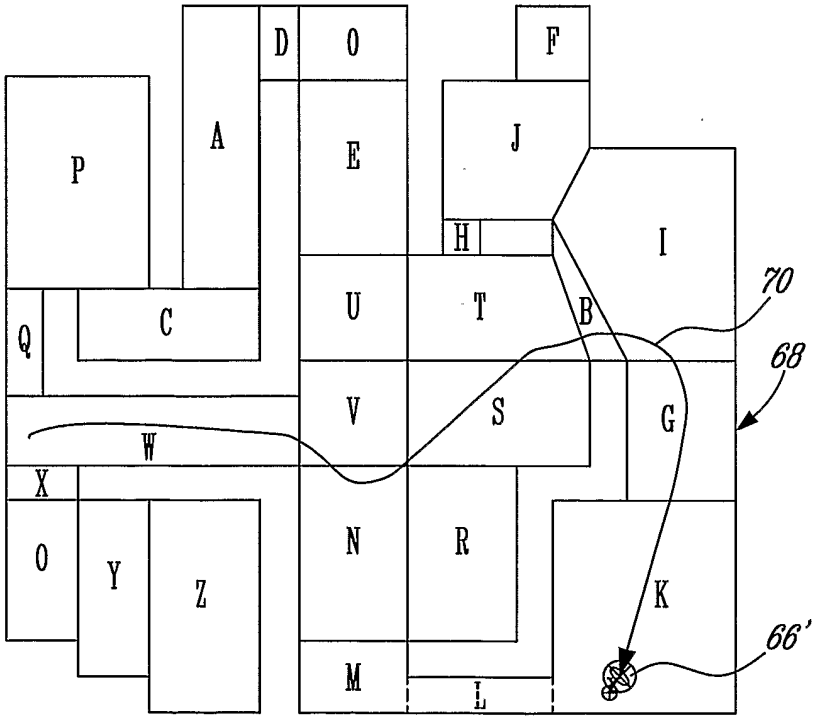

D
O
F
P
A
E
J
H
I
70
U
T
B
Q
C
68
V
S
G
W
X
N
R
O
Y
Z
K
66'
M
L

FIG. 21

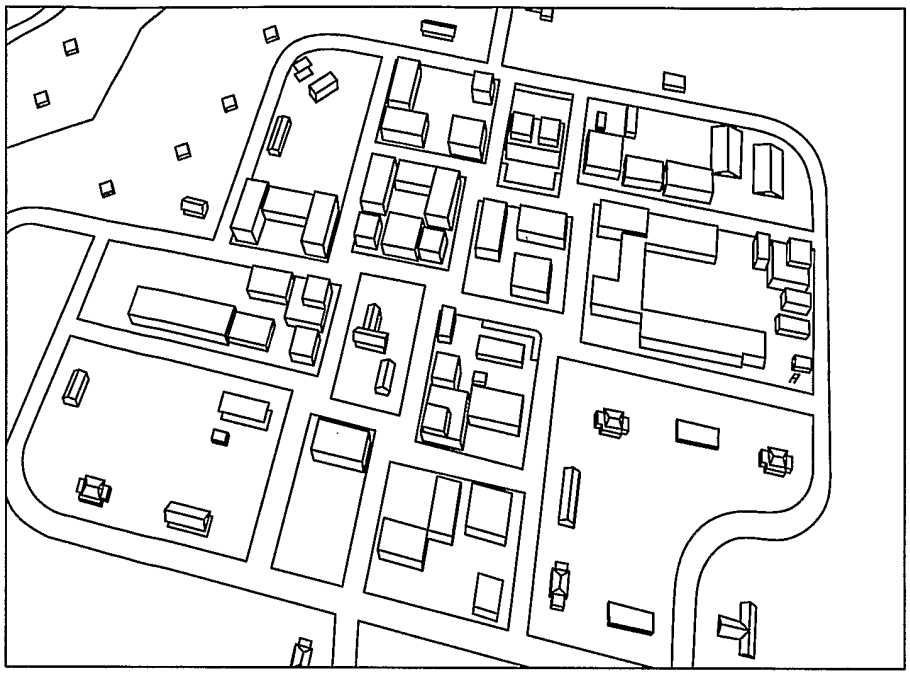


FIG_22

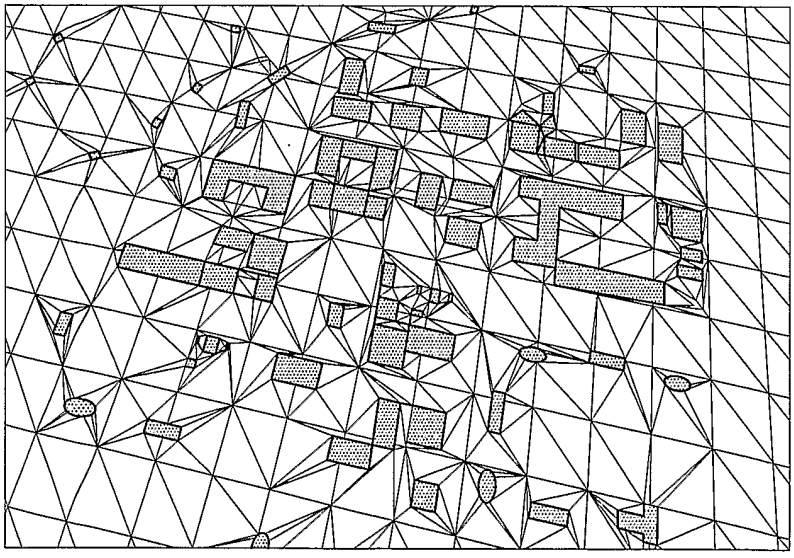


FIG_23

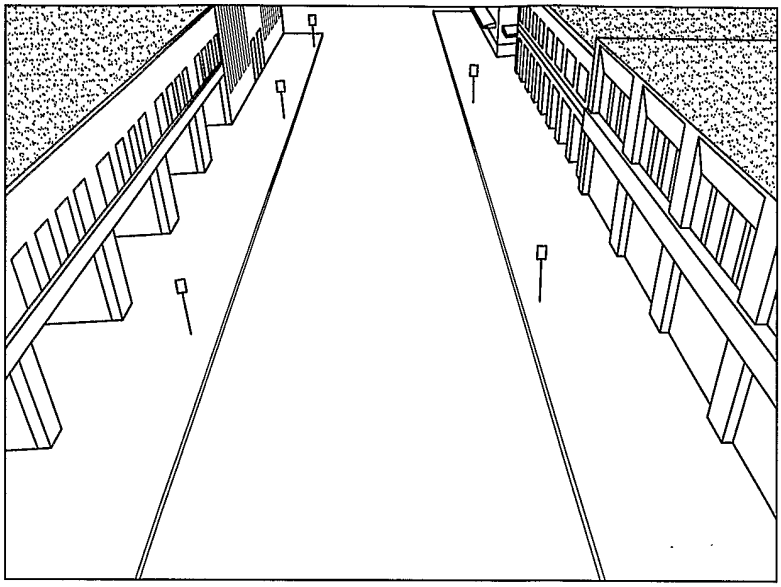


FIG_24

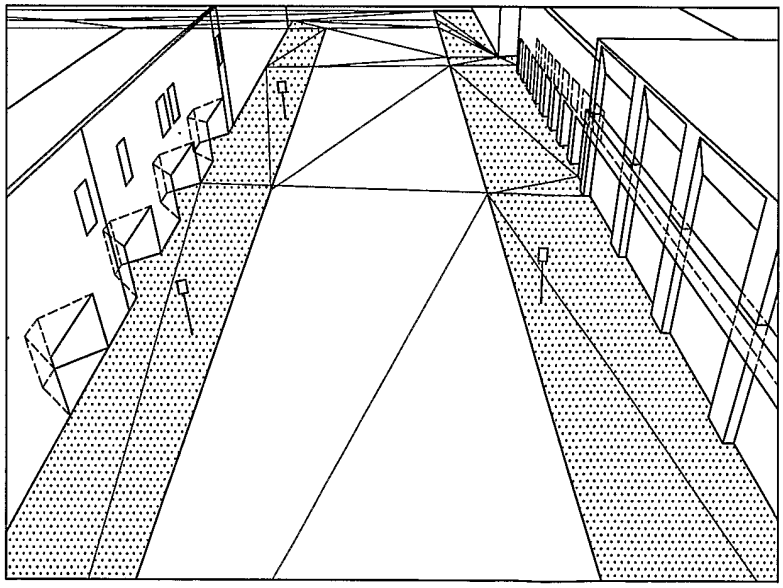


FIG_25

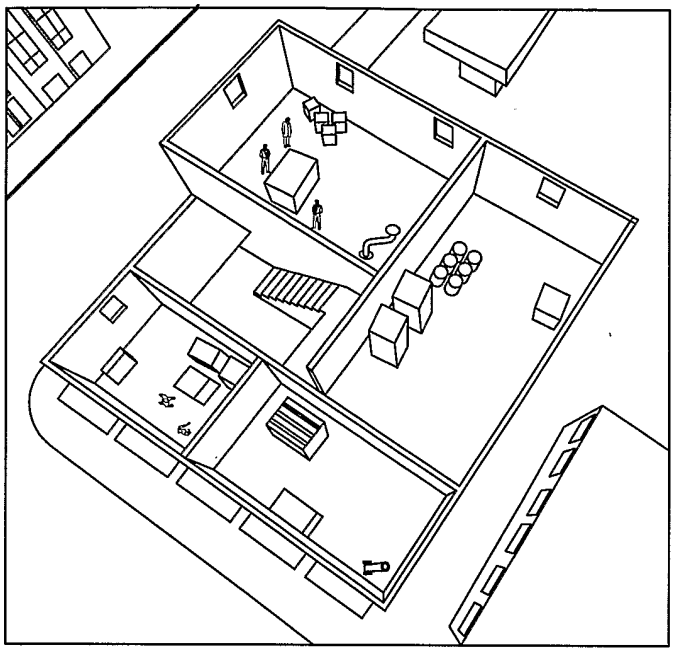


FIG_26

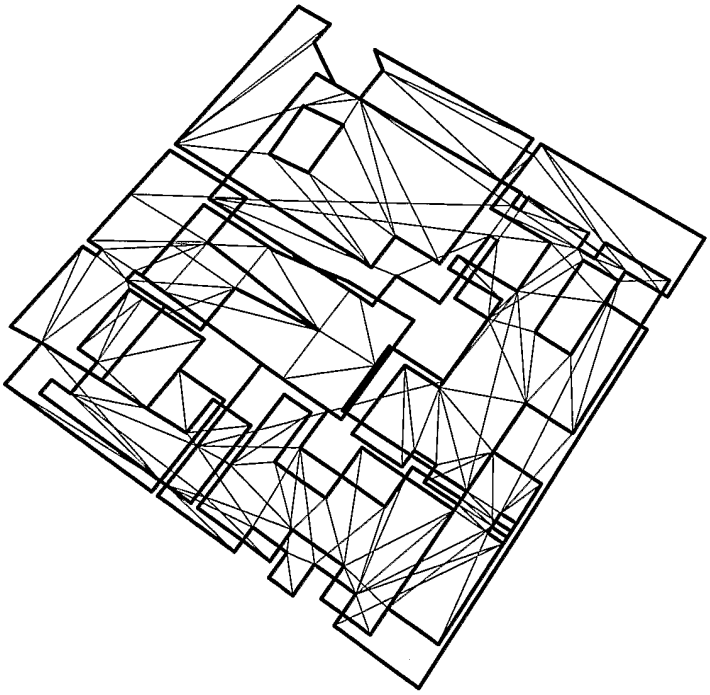


FIG_27

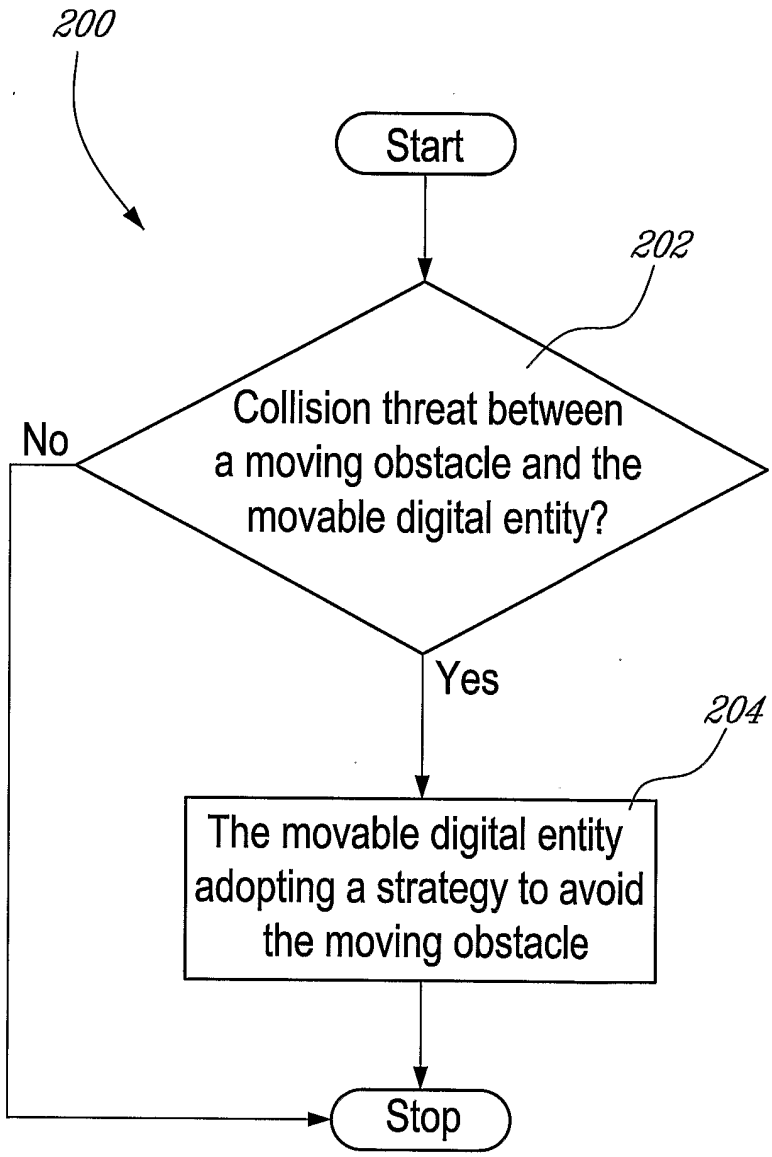

200
Start
202
Collision threat between a moving obstacle and the movable digital entity?
No
Yes
204
The movable digital entity adopting a strategy to avoid the moving obstacle
Stop

FIG. 2B

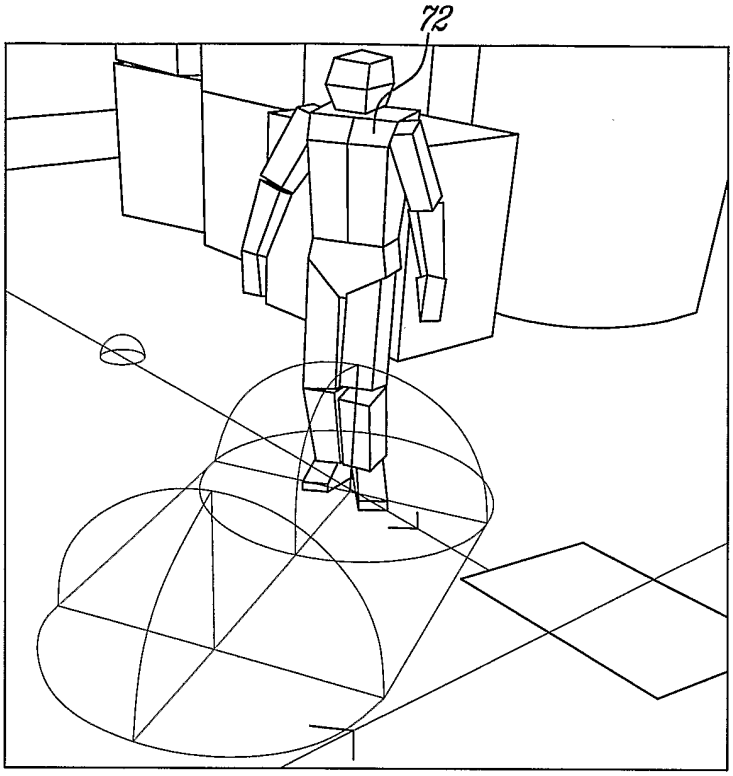


FIG_29

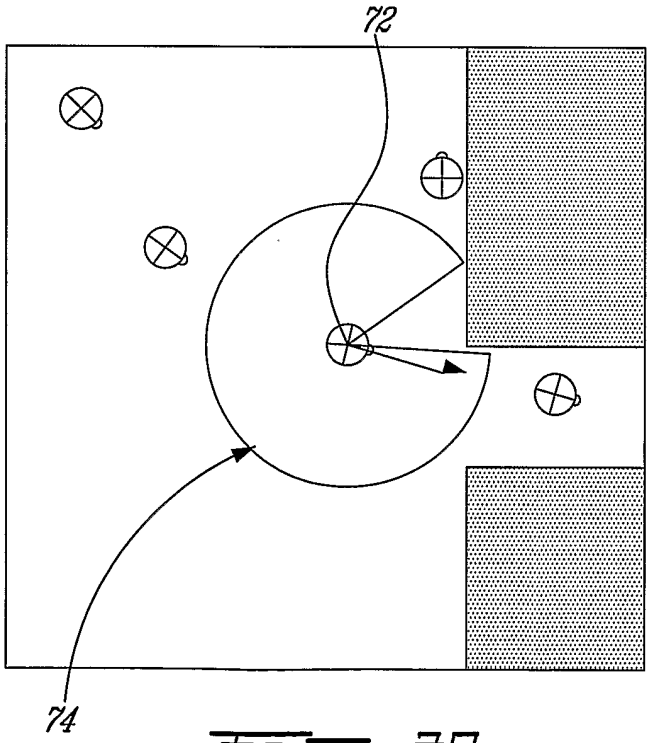


FIG_30

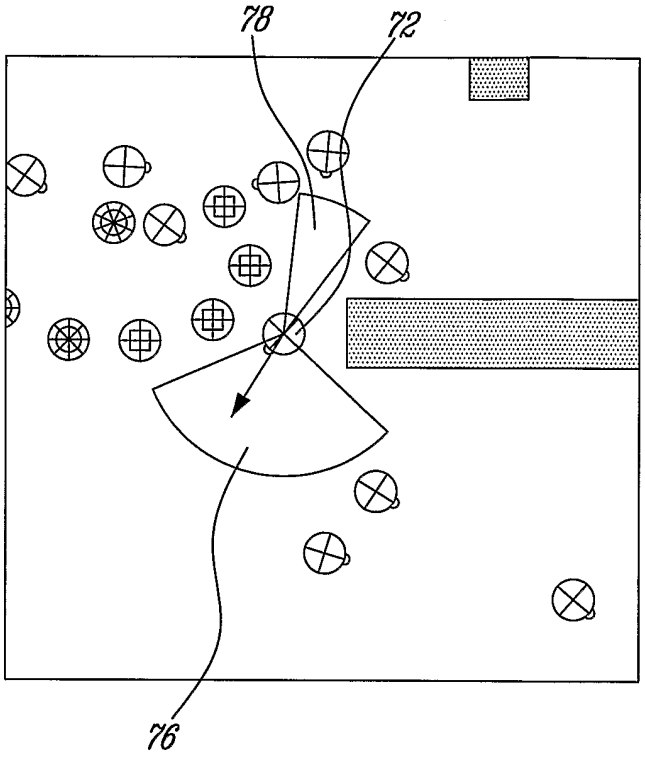


FIG. 31

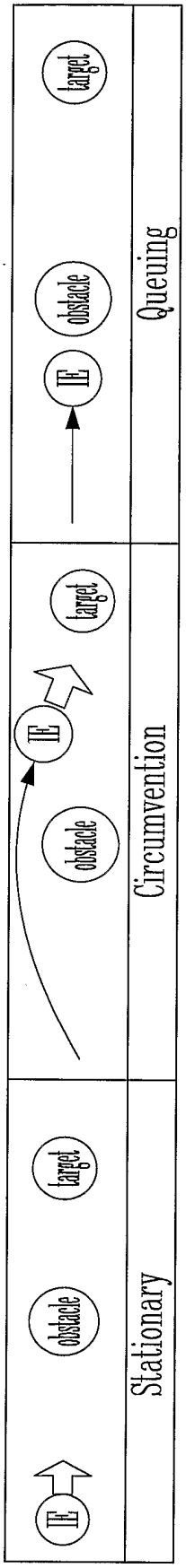


Fig. 32A

Fig. 32B

Fig. 32C

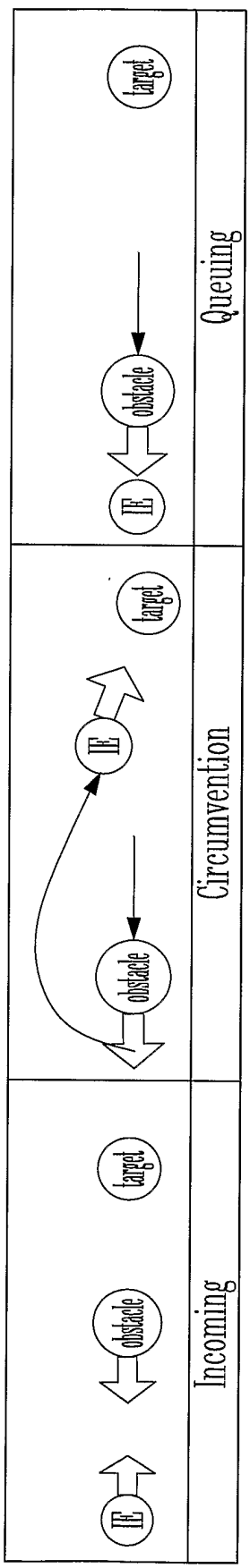


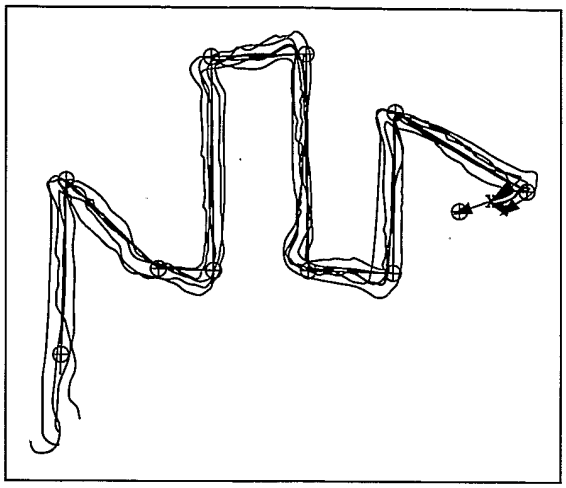
Fig. 33A

Fig. 33B

Fig. 33C

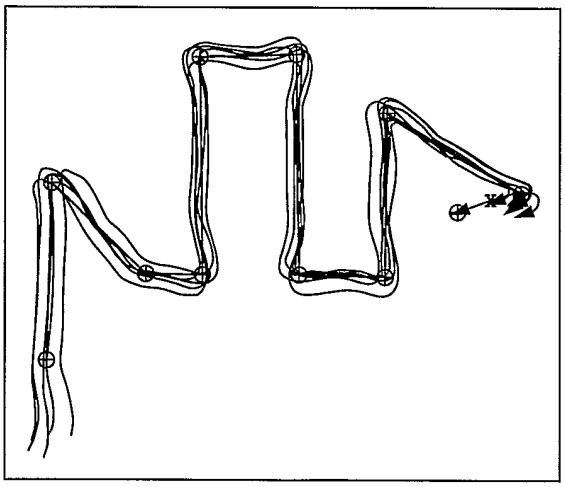
Fig. 34A Fig. 34B Fig. 34C

Fig. 35A Fig. 35B Fig. 35C



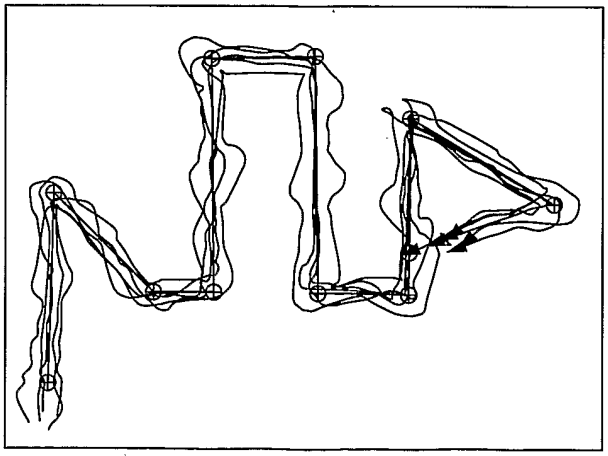
Just following

FIG_36A



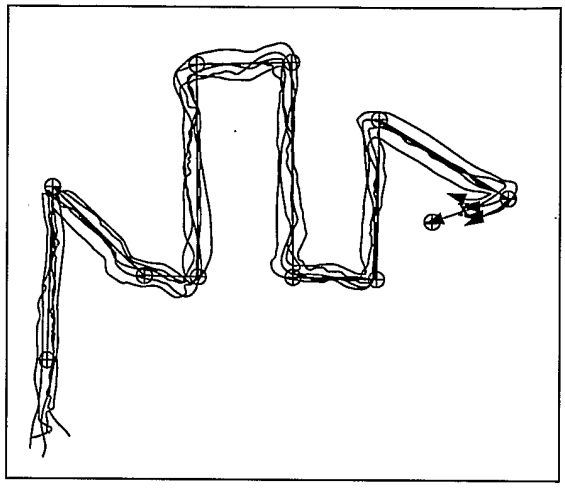
Alignment only

FIG_36B



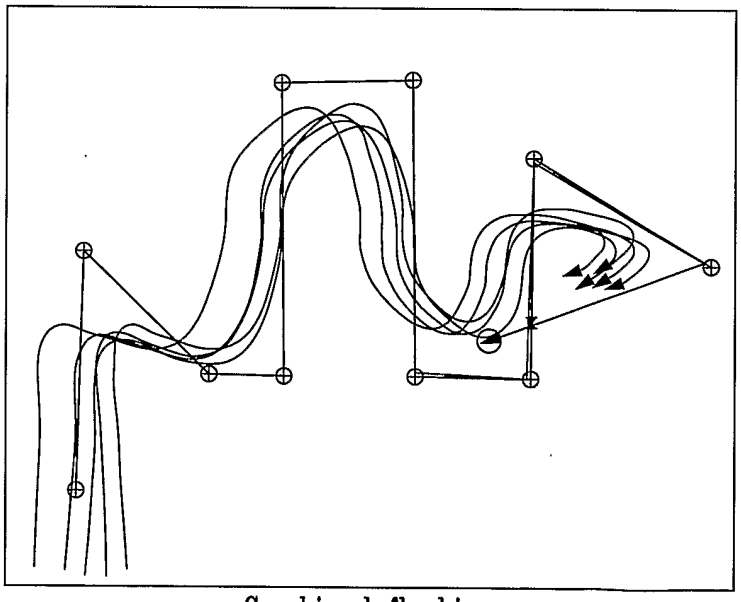
Separation only

FIG_36C



Joining only

FIG_36D



Combined flocking

FIG_36E

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2005/000426

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7): G06F 19/00, G06F 15/18
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC(7): G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Delphion, IEEE, Association for Computing Machinery database (ACM portal), the internet specifically CiteSeer and Google
keywords used: avatar, character, path finding, navigation, navigation mesh, collision detection, virtual sensor

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Kuffner, J.J. 'Goal-Directed Navigation for Animated Characters Using Real-Time Path Planning and Control', Proceedings of CAPTECH'98, November 1998	1, 2, 5, 6, 15, 18, 21, 23, 25
Y	abstract, 4 Goal-Directed Navigation, 5 Algorithm Overview, Path Planning the last paragraph, 6.1 Obstacle Projection, 6.2 Path Search, 7 Path Following (citations for both X and Y)	3, 7, 8, 16-18, 24, 26-28
Y	US 4 862 373 (Meng) 29 August 1989 (29-08-1989) abstract, column 1 line 12 - column 2 line 26, column 2 lines 30-37 and lines 60-67, column 3 lines 1-10, lines 26-28, lines 40-46, column 4 lines 23-66, column 7 lines 27-35, column 9 lines 17-25, Figure 16	3, 8, 24, 26-28
Y	Salomon et al. 'Interactive Navigation in Complex Environments Using Path Planning', Proceedings of the 2003 Symposium on Interactive 3D graphics, Monterey California, April 27-30. 2003, Pages 41-50. For Y see pages 45-46	3, 7, 26, 28
A	For A see whole document	1-27

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 May 2005 (27-05-2005)	21 June 2005 (21-06-2005)

Name and mailing address of the ISA/CA	Authorized officer
Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001(819)953-2476	Kristina Deczky (819) 934-4156

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	Reynolds, C. W. 'Steering Behaviors For Autonomous Characters', Proceedings of Game Developers Conference, San Jose, California. Miller Freeman Game Group, Pages 763-782, 1999. For Y see page 15	16-18
A	For A see the paragraphs under the headings Introduction, Locomotion, Seek, Evaision, Offset pursuit, Obstacle Avoidance, Path Following, Separation, Cohesion, and Alignment	1-27
A	O'Neill J. 'Efficient Navigation Mesh Implementation', Journal of Game Development, March 2004 see whole document	1-27
A	Pinter M. 'Toward More Realistic Pathfinding', Gamasutra, March 14 2004 http://www.gamasutra.com/features/200103014/pinter_01.htm see whole document	1, 3, 5

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/CA2005/000426

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US4862373	29-08-1989	JP1065411 A	10-03-1989
		US4862373 A	29-08-1989