University of Glasgow | School of Computing Science

Project Title

Michael Kilian – 1003819K
Tony Lau – 1102266L
Dan Tomosoiu –1102486T
Hector Grebbell – 1007414G
Peeranat Fupongsiripan – 2056647F

Level 3 Project — 19 March 2013

Abstract

The abstract goes here

Education Use Consent

We hereby give our permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: ____________________ Signature: ____________________

Name: ____________________ Signature: ____________________

Name: ____________________ Signature: ____________________

Name: ____________________ Signature: ____________________

Name: ____________________ Signature: ____________________

Name: ____________________ Signature: ____________________

Contents

Chapter 1

Introduction

1.1 Team Structure & Development Process

1.1.1 Team Structure

In order to develop a structured approach to task allocation, the software engineering tasks require were structured using the Administrative Programming Team [9]. This consists of the following roles:

- **Project Manager**
- **Librarian**
- **Configuration Manager**
- **Toolsmith**
- **Quality Assuror**

It should be emphasised that each person was not solely responsible for the tasks associated with their role; their responsibility is to coordinate these tasks within the team by proposing, implementing and maintaining effective procedures to acheive this.

In tandem with this, the team was further divided into two subteams for development:

- **Modelling and GUI Team:** responsible for development of any 3D models required including the final model of the ship. In the later stages of the project this team developed the graphical user interface.
- **Core Implementation Team:** responsible for all other development tasks, including implementation of the navigation mesh, population model, etc.

By seperating the development of the user interface from the development of the underlying program logic, the team aimed to promote a Model-View-Controller design. [**?**, Ch 6.3.1].

1.1.2 Development Process

One of the challenges of designing an evacuation simulator is defining a level of accuracy which can be deemed acceptable with respect to the project's resources, and then translating this into an effective design which balances the use of up to date techniques with an implementation plan. Many of the techniques that this project aimed to implement are complex. These techniques are discussed further in the Research section.
Ultimately, it became clear that the most tangible way to make progress was to use a strategy of incremental prototyping [**?**, Ch 2.3.2]. This would allow the team to progressively 'scale up' ideas and to investigate the feasability of implementing certain principles in a structured manner. However incremental prototyping carries considerable risk which must be addressed:

- A tendency to produce low quality and difficult to maintain code.
- Difficulties in managing change.
- Tendency to sacrifice quality assurance and documentation because of poorly understood aims.

To mitigate these risks, several techniques taken from the field of agile development [**?**, Ch. 3] were employed as follows:

- **Division Into Subteams:** The team was divided as outlined above so as to allow these subteams to work in parallel on orthogonal tasks, This reduced communication overhead and the difficulty of managing change to the system.
- **Constant Refactoring:** Before the completion of each prototype or upon fixing a defect, significant reactoring was undertaken to improve code quality.
- **Pair Programming:** This technique was particularly helpful when fixing defects related to navigation (see Implementation) due to the complexity of these defects.
- **Test First Development:** wherever the understanding of requirements was sufficient to allow it, test cases were developed for a feature before they were implemented. The full testing procedure is discussed in the Evaluation section.

Chapter 2

Research

2.1 Research into Human Behaviour

2.1.1 Non-adaptive Behaviour

Adaptive behaviour is any behaviour is any behaviour "which contributes directly or indirectly to an individuals survival". Conversely non-adaptive behaviour is any behaviour which may be counter productive to an individuals survival [1]. In the context of an evacuation this refers to high risk actions which occur in a crowd such as stampede, pushing and shoving others, trampling others, etc.
The introduction of non-adaptive behaviour can be connected to the stress a person is feeling. More specifically, Law et al [8] propose that there are three factors which contribute to the emergence of adaptive or non-adaptive behaviour:

- **Panic**: when a person perceives danger they are more likely to make irrational decisions based on instinct [10].
- **Decision-making**: although panicked, a person is still capable of making rational decisions. This increases the likelihood of the individual making adaptive choices, such as correctly recognising an exit or refraining from shoving upon exiting.
- **Levels of urgency to exit**: individuals within a group will experience varying levels of arguing to leave the environment based on the level of danger they perceive themselves to be in. High urgency causes individuals to behave aggresively and prioritise self-preservation.

All of these theories provide insights into human behaviour but have yet to be unified into a comprehensive theory.

2.1.2 Bounded Rationality

Bounded Rationality can be expressed as the principle that in a given situation the rational decisions a person can make are bounded by the set of possible options available to them and the time in which they can make a decision. This concept was initially proposed by Herbert Simon, who highlighted two interlocking components of bounded rationality [5]:

- **The limitations of the human mind**: the human mind does not have limitless processing power or memory and so must use approximate methods to handle most tasks. For computational purposes these methods can be expressed using simple heuristics.
- **Environmental structure**: Simon emphasised that the heuristics used must be adapted to the environment in which the decision is made.

An important example of this priciple is Simon's concept of satisficing: a "method for making a choice from a set of alternatives encountered sequentially when one does not know much about the possibilities ahead of time" [5]. In an evacuation a person will use this satisficing technique to make decisions such as where to move next based on what they can see.
It is important to note that, like other rational behaviours, this process can be disrupted by the introduction of stress factors, as previously discussed.

2.1.3 Conformity & Social Proof Theory

Conformity can be defined as a "change in behaviour or belief toward a group as a result of real or imagined group pressure". This can be further divided into *normative influence* and *informational influence*. Normative influence refers to changes in behaviour for the sake of winning the approval of other group members. Informational influence refers to conforming in an attempt to improve one's knowledge of reality and current situation. [7].
Informational influence, which is often called Social Proof Theory [3], has the effect that when facing a situation with uncertainty, a person may turn to the surrounding group for cues. Cialdini noted "we seem to assume that if a lot of people are doing the same thing, they must know something we don't" [10].
This principle is at the core of many of the behaviours observed in evacuations, particularly herding behaviour [10, 13].

2.1.4 Personal Space

According to Edward T. Hall [6] the space surrounding a person can be divided into four "reaction bubbles" which represent an acceptable radius around the person in which different categories of interaction. These are:

- **Public Space**: this space is used for speeches or other interactions with large audiences. This includes anything beyond roughly 2.4m
- **Social Space**: the distance reserved for interaction with strangers or newly formed groups. This ranges from around 1.2m to 2.4m
- **Personal Space**: this begins an arms length away from the individual's center and is reserved for interactions with friends or other close associates

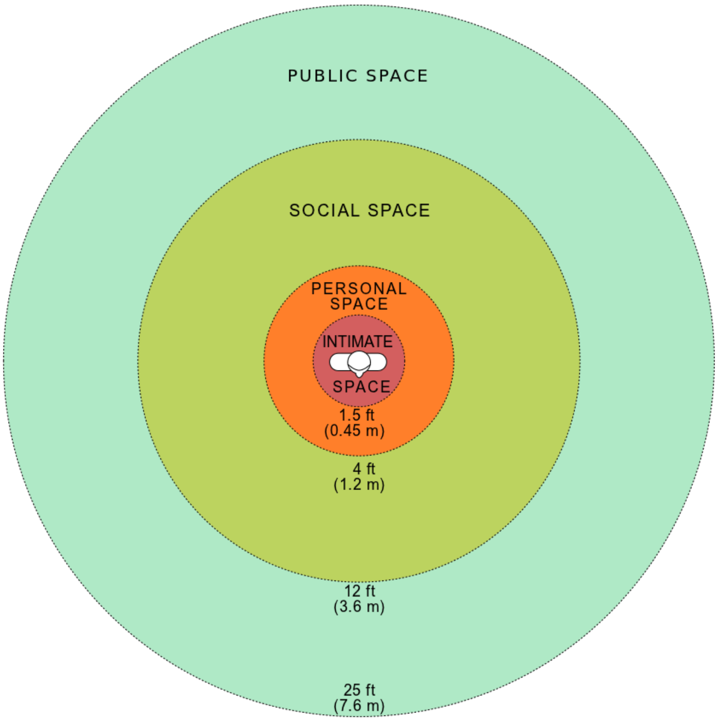

Figure 2.1: Edward T Hall's 'reaction bubbles'

- **Intimate Space**: The smallest of the spaces, this ranges from touching the person to around 50cm from their body. Interactions in this space are extremely personal and normally occur only with close members of family or friends

The last two of these spaces are of particular interest. Another definition of personal space is the area surrounding an individual "into which others may not intrude without causing discomfort" [7, pg. 424].
In a crowded environment the violation of ones personal space is likely to increase stress and anxiety (see page 7. Naturally this effect is even greater when the intimate space is violated. An individuals desire to re-establish this boundary can introduce non-adaptive behaviour.
It should be noted that the definition of these four spaces varies between individuals and between different cultures. Therefore the ranges discussed here are only an estimate [2]. 2.1 shows these 'bubbles'.

2.1.5 Classes of Evacuation Behaviour

The MASSEgress framework [13, 10, 14] defines three classes of evacuation behaviour which will be considered in this projects behavioural model for agents. These are *competitive behaviour*, *queueing* and *herding*.

Competitive Behaviour

Competitive behaviour is observed when individuals attempt to force their own exit by competing with others. This includes pushing, moving at an unsafe speed towards exits, etc. Such behaviour generally reduces the efficiency of egress, especially at narrow doorways or other tight spaces[12], compared to exiting in a non-competitive (queuing) manner [4].
In general this behaviour emerges when a person is highly stressed and perceives an urgent need to evacuate. However an individual can also take up competitive behaviour as a result of social proof, if other individuals around them are acting competitively.

Queueing

This can be seen as the converse of competitive behaviour. It is characterised by a group organising themselves to facilitate efficient and safe passage through an exit.

Herding

Broadly speaking one can define herding as ”the alignment of the thoughts or behaviours of individuals in a group (herd) through local interaction and without centralized coordination” [11]. Again social proof is a key contributor to the emergence of this behaviour. When an individual has a high degree of uncertainty they will follow the crowd.
Unlike other behaviours herding can be either adaptive or non-adaptive depending on the context in which it occurs. Suppose that an individual is following a crowd through an exit. It is possible that this exit is the correct way to proceed, so this choice has aided the person's egress. Equally it may lead to a dead end.

2.1.6 Unifying These Concepts In A Behavioural Model

We have discussed some of the properties of an individual and groups (such as panic, urgency to exit, etc.) which affect the behaviours exhibited in the evacuation process.
For modelling purposes, these factors are unified into a single parameter which shall be referred to as 'evacuation stress'. This can be expressed as a percentage where 0 is equivalent to an individual being in regular day-to-day conditions, whereas 100 represents an individual in blind panic acting almost entirely on instinct. Of course these two extremes are unlikely to occur in practice.
The reason for this extreme simplification is to limit the possible conditions which must be checked in decision making. The more attributes an agent possesses which may affect decision making there are, the more conditions must be checked at each decision making stage. Even a small number of variables can lead to an very large set of possible outcomes which all must be checked. This can exponentially increase both the complexity of the behavioural model for the programmer and the computational complexity of making a decision.

2.1.7 The Perceive, Decide, Act Process

The decision making process for each agent can be expressed in three stages:

1. **Perceive**: the agent scans the area around themselves for visible goals or other information. This produces a set of goals in sight.
2. **Decide**: based on what they can perceive and their current level of evacuation stress, the agent chooses an action with which to proceed.
3. **Act**: the agent carries out this action.

An example of this procedure is as follows:

1. An agent scans the room they are in. They perceive two exits, one with only two people moving through it, the other with twenty people waiting at it.
2. The agents current evacuation stress is high enough that they will engage in herding behaviour. They decide to move toward the most crowded of the two doors.
3. The agent plans their route to this destination and initiates the movement.

The implementation of this theory is discussed in the Implementation section.

Chapter 3

Design

3.1 Population

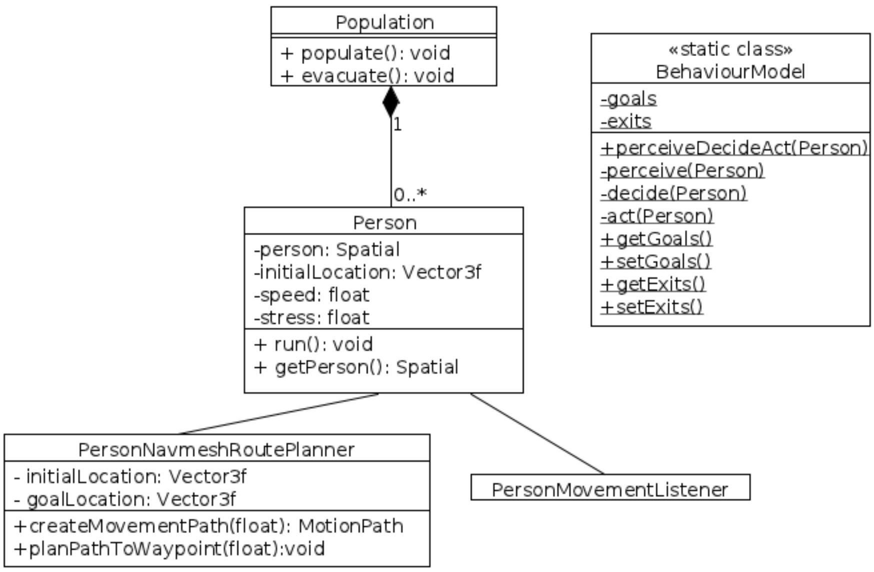

Figure 3.1: Population Package

Figure 3.1 shows the relationship between the various classes responsible for the realisation of autonomous agents and their behaviour within the simulator.
The Population class represents the set of agents. A population instance is ran as a seperate thread of execution which incrementally performs computations on the entire set of agents, such as computation of collision groups. It is also responsible for the generation of the set of agents using the information provided in a set of PersonCategories and the initiation of agent evacuation.
BehaviourModel is a static class designed to implement the Perceive, Decide, Act process. For each step there is a corresponding private method which is called. The output from the perceive method, a list of goals visible to the agent, is passed to the decide method which chooses a new target location for the agent to plan a path towards, and this location is passed to the act method which initiates the agents movement towards its new goal.

The seperation of these three stages is important to maximise the extensibility of the behavioural model. As mentioned, the decide method processes a set of goals and, based on a set of rules and heuristics, chooses a new location for the agent. To change or extend the range of decisions an agent can make, the programmer can simply add new rules to the decide method. In fact it is possible that multiple decision methods could be added where each implements a different behavioural model. The only restriction is that the decide method must take a set of Goals(see below) as one of its inputs and outputs 3D location. By calling the perceiveDecideAct method, an agent is taken through all three stages of the process and need have no knowledge of the underlying computations performed to select it's new goal.

An agent is primarily realised by the three classes Person, PersonMovementListener and PersonNavmeshRoutePlanner. Each Person holds the attributes, including speed and stress (see Behaviour Research) which represent the characteristics of an evacuee. It also coordinates the visual representation of an agent and the movement of the agent towards its goal.
In order to plan a route to a location, a Person must create a PersonNavmeshRoutePlanner instance. It's purpose is to act as if it were a 'ghost' agent: it rapidly traverses the navmesh to establish a route. This route is then returned for the agent to move over. A fresh instance of PersonNavmeshRoutePlanner should be used every time a new route must be calculated and can be disposed of once the route has been returned.

3.2 Goals

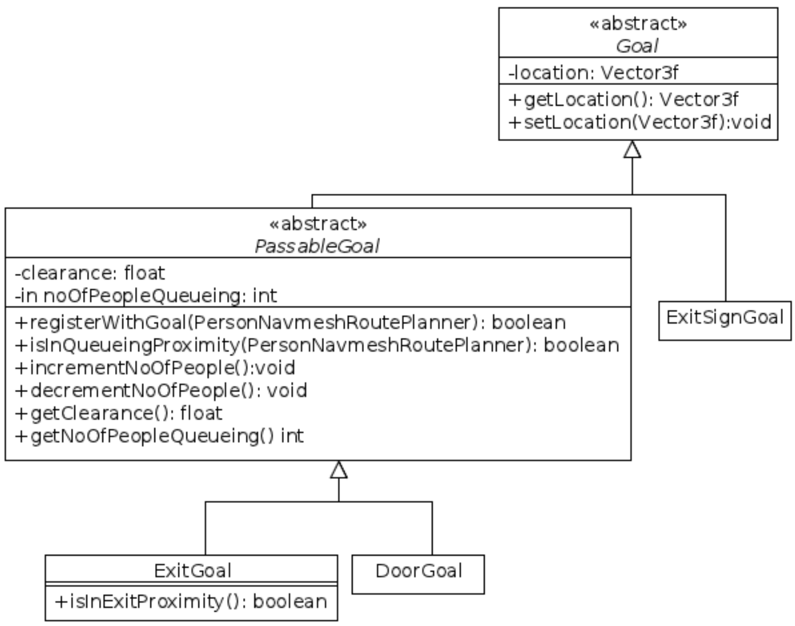

Figure 3.2: Hierarchy of goal package.

A goal can be defined as anything in the environment which can steer an agents direction, whether towards itself or another goal. Figure 3.2 shows the inheritance hierarchy for goals in this simulation. At the top level we define a generic Goal, which simply has a location and does not direct

agents explicitly.
A PassableGoal is any goal through which an agent can move, such as a door or exit. These also have a clearance; a radius around it's center which defines the maximum distance at which an agent can be considered to be queueing at that goal. This is vital for the computation of queueing behaviours. The no of people queueing at a PassableGoal is also stored and must be updated externally by agents as they approach or leave the goal. Exits are represented by ExitGoals which are equipped with methods for recognising when an agent is close enough to exit the simulation.
ExitSignGoal is an example of a non-passable goal which could direct agents to another goal. These however are not implemented in this project.

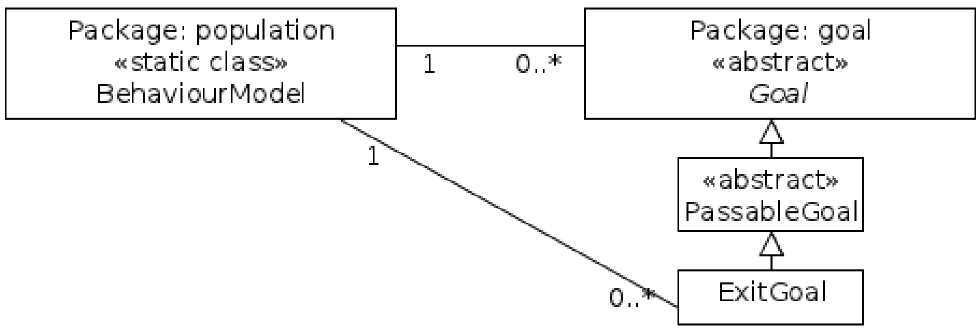

Figure 3.3: Relationship between BehaviourModel and Goals

As figure 3.3 shows, the static BehaviourModel instance holds a collection of all Goals in the simulation. For the purposes of calculating agents paths out of the environment, a collection of all the ExitGoals is also kept seperately, since it is anticipated that these will be used most frequently in calculations.

Chapter 4

Implementation

4.1 Route Planning

Planning an agent's route from its current point to a given point is acheived by a collaboration between a Person and PersonNavmeshRoutePlanner instance. Routes are stored using the jMonkeyEngine class MotionPath. This contains a set of waypoints and methods to animate the agent's movement between these waypoints. The Mmaximum distance between each waypoint is roughly constant and can be set as a parameter to the following procedure (it is not constant when the agent must turn at a point closer to its current position that the maximum distance between points). The distance used is an important decision in the implementation. If it is too large the accuracy of the animation tends to degrade. If it is too low then a large number of waypoints will be stored needlessly.After extensive experimentation 0.5 was found to be an acceptable value for the maximum distance between waypoints. In practice this provided a balance between quality and efficiency.
A route is calculated in two stages:

1. A path is calculated on the navigation mesh using a modified A* algorithm to traverse the mesh like a graph. This returns a small set of points. These points illustrate the lines of motion an agent must take to reach their goal. This is performed in the constructor for a PersonNavmeshRoutePlanner.

2. Using these points as guidance, the path is 'fleshed out' by moving along the path and placing MotionPath waypoints no further apart than the defined maximum distance. This terminates with a waypoint being placed on the goal location the agent must reach.

Note that a fresh PersonNavmeshRoutePlanner instance must be instantiated to calculate a route. The relationship between the Person and PersonNavmeshRoutePlanner instances is expressed in

4.2 Implementing the Perceive, Decide, Act Process

Here we discuss the techniques and algorithms used to realise the Perceive, Decide, Act process previously discussed [Reference to Research].

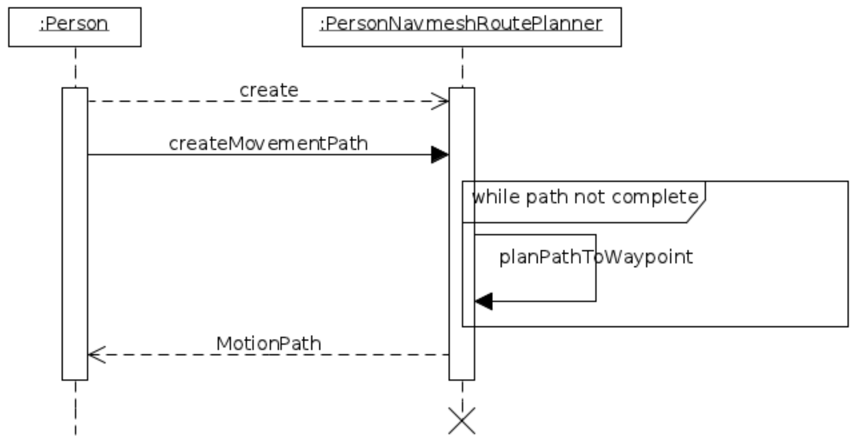

Figure 4.1: Generation of an agent route as a MotionPath

4.2.1 Perception

Agent perception is achieved using the following simple algorithm:

input : The set of goals in the environment: *goals*
output: A set of goals visible to the user at the given instant in time: *visibleGoals*

for *each goal g in goals* **do**
  if *g is in line of sight of agent* **then**
    add g to visibleGoals;
  end
end

Algorithm 1: Agent Perception Algorithm

4.2.2 Decision Making

In the Research Chapter [Reference] the three classes of human behaviour considered in the scope of this project were defined. In practice, only herding behaviour implemented as part of the Decide step; queueing and competetive behaviour must be handled asynchronously using collision avoidance techniques [Reference to Dan's work].
To decide on a new action, an agent must select one of the visible goals that were found in the Perceive step (if any). Otherwise it must continue on its current path. The decision making process is realised using an extendable algorithm, which sequentially considers sets of different classes of goal according to their priority. Exits have the highest priority.

The final algorithm only makes decisions regarding exits, for reasons we will discuss later in the report. However it is easy to extend this process to include other goal types by adding further conditionals following the pattern laid out below.

```
input  : Person person, ExitGoal[ ] exits /*add further exit types here as arrays */
output: Target Goal for agent to move toward
if no of exits > 0 then
    ExitGoal currentExit = exits[0];
    if person is stressed then
        for each exit e in exits do
            if number of people queuing at e > no. of people queueing at targetExit then
                targetExit = e;
            end
        end
        return targetExit;
    else
        Vector3f position = person.location;
        for each exit e in exits do
            if distance to e < distance to targetExit then
                targetExit = e;
            end
        end
        return targetExit;
    end
else
    return null;
end
```

4.2.3 Act

This step's representation in the BehaviouralModel class is trivial since the Decide step already returns a target goal. It is left in as a place for performing any calculations which should be performed before returning the target goal to the agent.
Upon receiving a new target goal, an agent should perform the following:

- Use a PersonNavmeshRoutePlanner to calculate a route to this goal
- Set the returned MotionPath as the current MotionPath for the agent
- Begin moving down this path

4.3 User Interface

Blah blah blah Blah blah blah Blah blah blah Blah blah blah

4.3.1 Foo

Blah blah blah Blah blah blah Blah blah blah Blah blah blah

4.4 Database Model

1. Blah blah blah
2. Blah blah blah
3. Blah blah blah
4. Blah blah blah

Chapter 5

Evaluation

5.1 User Interface Evaluation Methods

The simulator will be evaluated through a series of users tests with different participants. The goals of the user interface evaluation are to assess the effect of the interface on the user and to identify specific problems which should be rectified. The simulator will be first tested with the fire warden of The Tall Ship who is the primary user of the system. Secondly, feedback on usability of the system will be gathered from usability testing with the subjects being students at the University of Glasgow.

Two evaluation methods, namely Heuristic Evaluation and Usability Experiments, will be used to determine whether the requirements specified in section [insert section] have been met and also to determine the overall usability of the system.

5.1.1 Heuristic Evaluation

Heuristic evaluation is a usability inspection method pioneered by Jakob Nielsen and Rolf Molich which helps to identify problems with a user interface by judging the interfaces compliance to recognized usability principles – heuristics[].

The heuristics made use of in this part of the evaluation are Nielsen's Heuristics, developed by Jakob Nielsen and Rolf Molich in 1990 []. Nielsen refined the original set of heuristics in 1994 []. Below is list of the heuristics and a description of each one:
1. Visibility of system status: The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2. Match between system and the real world: The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3. User control and freedom: Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4. Consistency and standards: Users should not have to wonder whether different words, situa-

tions, or actions mean the same thing. Follow platform conventions.
5. Error prevention: Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
6. Recognition rather than recall: Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7. Flexibility and efficiency of use: Acceleratorsunseen by the novice usermay often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8. Aesthetic and minimalist design: Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
9. Help users recognize, diagnose, and recover from errors: Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
10. Help and documentation: Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

The user interface of the simulator will be examined and evaluated using the descriptions of each of Nielsens 10 Heuristics above. Heuristic Evaluation is a relatively quick and inexpensive way to evaluate a user interface. Deviation from recognized usability principles, identified from the evaluation, can provide great insight into how the user interface could be further refined to enhance the usability of the system.

5.1.2 Usability Experiments

Usability experiments can be used in addition to Heuristic Evaluation to gain further feedback on the system. These are particularly useful because they allow the experimenter to gain insight to the reactions of the users of the system first-hand.

Two different sets of users will be used for evaluation: the fire warden and a group of students. The fire warden will be able to tell us whether the system meets the requirements and also on how usable the system is. Since it is unlikely that the students work in the domain of fire safety, the set of students will be primarily used to test the ease in which the system can be used. Any suggestions of improvements to the system by either group will also be recorded.

Experimental Design

The experiment must be designed carefully in order to provide results that are both reliable and generalisable. Two types of experimental design can be used: within-subjects design and between-subjects design.

In a between-subjects (or randomized) design, each participant is given a different condition, of which there are at least 2. A control condition, where the independent variables are not changed, is needed to ensure the measured differences in the other conditions are true. Since each subject only performs under one condition, the likelihood of any learning effect from performing two similar conditions one after the other is mitigated. However, a between-subjects design requires a large number of participants if one is going to extract meaningful information.

In a within-subjects design, each subject is given the same conditions to perform. The effect of learning is more prominent in this method, which is a disadvantage but it has an advantage compared to between-subjects design because less subjects and time are required.

Considering the advantages and disadvantages of both methods, a within-subjects design will be adopted for the user interface evaluation of the evacuation simulator. Since limited resources are available in terms of time and users, the less costly within-subjects design is more appropriate. Learning effects can be lessened by changing the order in which the conditions are carried out by the participants. This allows a comparison of participants who carried out a condition first and participants who carried out the condition after another one, and therefore subject to learning. The results from the within-subjects evaluation will be analysed to determine whether effects of learning have adversely affected the results of the evaluation.

For both sets of users, they will carry out a set of fixed tasks and the time taken to perform these tasks as well as any mistakes they make will be recorded. This data will be used to form the evaluation results and will allow the identification of flaws in the usability of the system and areas for improvement. A section entitled Further Work will detail the improvements (if any) which could be made to the system to enhance both the usability and functionality of the system based on the results from both evaluation techniques detailed above.

Think-aloud Protocol

In addition to the usability experiment discussed above, the think-aloud protocol will also be used to gather information from users of the system. This method was introduced in the usability field by Clayton Lewis and is discussed in Task-Centered User Interface Design: A Practical Introduction by C. Lewis and J. Rieman. []. Think-aloud protocols involve participants thinking aloud as they perform a set of pre-specified tasks. The idea is to have the users of the system saying out loud exactly what they are doing and how they are feeling. This allows the experimenter to gain a first-hand sight of a user using the product and provides insightful knowledge into how the end user would go about performing tasks.

The information gathered from the experiment will be analysed and the difficulties the user had will be discussed and rectified by changing the user interface. Any major changes to the user interface will have to be evaluated again to ensure the changes actually improve the usability of the system as a whole.

5.1.3 NASA TLX: Task Load Index (TODO)

- NASA TLX was used after the Think Aloud evaluation to measure the workload of the participants relating to the tasks that they had just performed. Talk about what it is and how it measures workload.

- Include pictures of the rating scale and the description of the rating scale from the instruction manual.
- Discuss the pairwise comparison and whether to use Raw TLX (RTLX). Cite 20 years later paper.

5.2 User Interface Evaluation Results

The results from the user interface evaluation using heuristic evaluation, think-aloud and NASA Task Load Index (TLX) are discussed below.

5.2.1 Heuristic Evaluation

Visibility of system status

The system gives users little information about what is going on. For example, on pressing the populate or route buttons, no information is displayed on the screen informing the user that something is happening in the background. As a result, the user could be left wondering whether the button was correctly clicked.

Inclusion of messages on the screen stating what is happening at a particular moment in time would help the users to identify the status of the system. For example, inclusion of a message saying that the system is loading instead of a black screen will inform the user that something is indeed happening. Messages when the user clicks a button confirming that the action is happening and a message that displays on the screen when the evacuation simulation has finished will also increase the users awareness of the system status which in turn makes the system more user friendly.

Match between system and real world

The arrow keys (up, down, left and right) for camera movement are positioned in a logical order which would be familiar to the majority of users. This allows an easy mapping from real world natural conventions to the system which makes the system easier to use. Non-natural placement of these buttons on the screen would cause confusion in users and lead to frustration.

The system does use the term navmesh in the checkbox labelled show navmesh. The user is likely to be unfamiliar with this technical term and would thus have to consult the documentation to learn properly what it does. The system should use words and phrases familiar to the user so it would seem appropriate to replace the word navmesh with something along the lines of ship outline or ship frame.

User control and freedom

The user has the ability to control the camera manually using the on-screen buttons. The allows the user to view the ship in any way they want. There is also functionality to increase or decrease the speed of the camera and also to select preset camera locations from a drop-down menu which increases the control and freedom the user has. The number of preset camera locations is currently two. This should be expanded to give the user more options.

The user can change the population size from the main screen and a more advanced settings menu allows the user to create categories of people according to various factors. This gives advanced users more freedom in the interactions they make with the system.

Consistency and standards

In general, the buttons are labelled well and the user does have to wonder what they do. However, some problems exist in the camera settings part of the main screen. There are two up arrows and two down arrows and their functionality is not made entirely clear to the user. One of the up arrows is to pan up and the other up arrow is to rotate upwards. This should be made clear to the user by either increasing the size of the button and including a meaningful button name, or include the information as text above the buttons on the camera controls panel itself.

Error prevention

Some steps have been taken to prevent the users from executing an action which would lead to an error in the system. The evacuate button is grayed out and is only allowable to be clicked by the user when the ship has been successfully populated. By stopping the user from performing this action until it is appropriate, allows the prevention of an illegal system state – namely evacuating an empty ship.

Error prevention related to the other buttons was overlooked and should be corrected. Other buttons on the main panel of the user interface should also be grayed out when it would not be appropriate to click. For example, the

Where there exist fields which can be altered by the user (for example, the population size field) a maximum and minimum value has been defined to prevent the user from entering too high or too low a number. This eliminates the possibility that the system will enter a state which it cannot handle as a result of user input which protects the system from mistakes made by the user.

Recognition rather than recall

The main buttons on the user interface are made to be as self explanatory as possible, however one shortfall is the design of the camera location panel. As discussed in the consistency heuristic, it should be made more clear what these buttons actually do. This would remove the need of the user to consult documentation for help and would thus reduce the extent of recall from one dialog to the next.

Flexibility and efficiency of use

The system has the ability to cater for more experienced user as mentioned in the user control and freedom heuristic. There exists the ability for the more experienced user to change a variety of properties of the camera such as speed and location. There is also the ability to use the keyboard to navigate around the ship which would allow more experienced users to increase their speed of interaction with the system.

There is, however, no method of allowing the user to tailor frequent actions through the use of accelerators or custom keyboard shortcuts, for example. The addition of such features would increase development time and since the experienced users group is a minority, it would not be in the best interest to develop this feature at this time.

Aesthetic and minimalist design

The system has a main screen which includes the most often used actions, and a settings screen which includes extra functionality. This separation allows a less cluttered minimalist view in the main screen which is easier to comprehend for the user. The system had been designed to display elements positioned in a natural way, allowing the user to focus on using the system and not needing to familiarise themselves with the interface for too long.

Help users recognize, diagnose and recover from errors

As mentioned in the prevention of errors heuristic above, some steps have been taken to prevent the user from making errors. However, since not all sections of the interface are removed from use when they are not supposed to be used, it is possible for a user to crash the system by pressing the buttons repeatedly. There are no friendly error messages to tell the user that something has went wrong – they are presented with a black screen. This leaves the user wondering whether the system is busy in the background carrying out some task, or has crashed.

To resolve this problem, better messages should be displayed on the screen to the user to increase visibility of system status. This would eliminate any confusion from the user when using the system. A reset button should also be implemented as a last resort for the user to click if the system crashes for an undocumented reason. This will ensure robustness in the system and increase the user-friendliness as a whole.

Help and documentation

No help or documentation is provided to the user. While the majority of the system is, on the whole, quite intuitive to use, some parts such as the camera controls panel and the advanced settings dialog box would benefit from documentation.

Brief documentation on the basic parts of the interface and what each button does as well as more detailed documentation on the more complex parts of the system should be created to assist the user in using the system and making decisions.

5.2.2 Think Aloud

The Think Aloud evaluation of the user interface was carried out with eight participants. The participants were asked to complete the six tasks listed below and were encouraged to think aloud during the evaluation. The participants were observed and notes were taken to allow a discussion of improvements which could be made to the user interface after the evaluation.

Tasks

1. Set the population size to 50. Populate the ship.
2. Hide the ship frame from view, then show it again.
3. Generate the exit routes for the population
4. Change the camera angle so you have a birds eye view of the ship.
5. Evacuate the ship. While the evacuation is taking place, change the camera view to face the exits of the ship.
6. Read out loud the time the simulation took and the number of people evacuated.

A summary of the findings of the Think Aloud evaluation is discussed below. Detailed notes on a per-participant level are included in Appendix C.

- The visibility of system status was a clear shortfall as indicated by the participants. Particularly in tasks 1, 3 and 5 where buttons attached to many lines of code had to be pressed, the participant was left guessing whether anything was happening with the system or whether they had done something wrong.
- The meaning of the camera location buttons was also not as clear as they should have been. Participants had to investigate what the buttons did and clear labelling would have solved this issue.
- The participants had no difficulty identifying key information such as number of people evacuated and the time taken for the evacuation which confirms that the information is well visible and the labelling is unambiguous.
- Identification of the completion of the evacuation was most often done by looking at whether anything more was happening in the graphical representation rather than looking at the 'remaining people' metric at the right hand side of the interface. It was noted that a pop-up message when the simulation is complete would make this more clear to the user.
- The terminology used in the user interface must be standardised to remove jargon words such as navmesh which would be unclear to the user. Task 2 highlighted that while some users guessed that ship frame was equivalent to the navmesh in this particular situation, 5 of the 8 participants were confused by the technical language.

5.2.3 NASA TLX: Task Load Index (TODO)

- Insert graph.
- Discuss that there is a low workload in using the system.

5.3 Appendix C: Think Aloud Evaluation Participant Notes

Tasks 1. Set the population size to 50. Populate the ship. 2. Hide the ship frame from view, then show it again. 3. Generate the exit routes for the population 4. Change the camera angle so you have a birds eye view of the ship. 5. Evacuate the ship. While the evacuation is taking place, change the camera view to face the exits of the ship. 6. Read out loud the time the simulation took and the number of people evacuated.

Participant 1 (ID: 146) 1. Participant successfully changed the population size to 50 but was unclear about the status of the system after clicking the populate button. Participant questioned whether anything was happening. 2. Did not understand the task fully, particularly what was meant by ship frame. The participant tried to hide the ship from view by changing the location of the camera so that the ship was not in view but this was not what was wanted. 3. Participant was eventually able to accomplish the task although at one point was confused and thought camera location had an effect of route generation. This did not affect the outcome of the task, though. 4. Controls used correctly but the camera angle was slightly short of the birds eye angle asked for. 5. Participant knew to use the evacuate button. When changing the camera view, the participant made use of the camera location controls and manually panned the camera to find the exits instead of using the preset camera location drop-down menu. Judging the completion of the simulation was done by noticing the absence of people from the ship and not by the remaining people display panel. 6. Completed successfully.

Participant 2 (ID:102) 1. Accomplished successfully, although expressed views of uncertainty of system status. 2. No problems completing task. 3. Successfully generated routes although was unsure again about system status. The participant said that they did not know whether the button had worked or not. 4. Participant went straight to the camera location drop-down but realised another method was needed because birds eye was not there. Tried to use the manual controls and the mouse on the simulation environment but couldnt finish the task. 5. Clicked the correct button for evacuate but when selecting the exits item from the drop-down menu, did not immediately click the move button to register the action. The participant realised the mistake and clicked the move button to finish the task. 6. Completed successfully.

Participant 3 (ID:93) 1. Accomplished without hesitation. 2. Looked at the interface for longer to identify the correct option. At first the participant clicked the drop-down camera locations but realised this was a mistake. The participant was confused as to what the task was asking and assumed navmesh was the ship frame. 3. Moved mouse from the top to the bottom of the display to identify the button to press. The participant correctly identified and pressed the button although was unsure about the progress of the system. The participant assumed something was happening in the background because a lot of text was being displayed in the console. 4. The camera locations drop-down was the first thing the participant looked at. The participant could not find the correct option so looked for an alternative. The cam speed slider was tried a few times. The participant expected it to do something but nothing was happening on screen. The participant then pressed buttons haphazardly and admitted that they could not finish the task. 5. Successfully evacuated the ship, moved the camera to the exits using the drop-down menu and identified when the evacuation ended. 6. Completed successfully.

Participant 4 (ID:24) 1. Population size changed successfully but there was a slight hesitation finding the populate button. 2. Completed successfully although the participant had some confusion over whether the frame asked for in the question and the navmesh were the same thing. 3. Clicked the route button but was unsure whether the routes were actually being calculated. 4. Could not find birds eye view in the drop-down menu and was confused over the semantics of the camera control buttons. The participant failed to complete the task. 5. The participant completed the task successfully. The completion of the evacuation was immediately noticed. 6. Completed successfully.

Participant 5 (ID:46) 1. Participant correctly changed the population size but thought that on entering the population, the system would populate itself. The participant clicked the populate button but was unclear when the system was finished completing the task. 2. At first the participant did not bring the ship frame back into view but was able to hide it with no problems. 3. Correctly identified the button to press but was unclear about the system state. 4. Participant was able to use the mouse and the buttons on screen to manipulate the camera angle. It was noted that the participant went to the on screen buttons first and then used the mouse on the simulation canvas to finish the task. 5. The participant panned to the exits using the manual buttons and not the preset camera location drop-down menu as anticipated. The participant also announced the end of the evacuation prematurely by not looking at the remaining people metric at the right hand side. 6. Completed successfully.

Participant 6 (ID:81) 1. Completed successfully but was unsure of system state. 2. The participant had a slight hesitation to find the correct button to press but accomplished the task nevertheless. 3. The route button was found clearly by the participant although it was remarked that nothing was happening on screen when there was really something happening in the background. 4. The participant went straight to control the ship with the mouse directly interacting with the canvas. This was supplemented by use of the camera control buttons provided on the interface and the task was completed successfully. 5. The participant did not notice the drop-down menu item to set the view to the exits view already defined. Instead, the manual controls were used to pan to the exit location. The remaining participants metric in the sidebar was not used to determine whether the evacuation was finished. Instead, the participant preferred to see whether there were any people still on the ship by looking at the simulation graphics. 6. Completed successfully.

Participant 7 (ID:3344862) 1. Participant thought it was quite obvious how to accomplish the task and completed it without any trouble. The system status was not clear to the participant. 2. The terminology of ship frame versus navmesh was questioned and assumed by the participant to mean the same thing. The task was completed after this initial hesitation. 3. Participant remarked that the button was called route and though that it should be called routes to indicate more than one route. The participant was not sure whether anything was happening and was confused about the overlapping lines which appeared on the screen after this task was accomplished. 4. Participant went straight to the camera locations drop-down to find a preset. After noticing that there was no preset, the participant proceeded to instinctively use the mouse and a combination of the on screen buttons to accomplish the task. 5. Successfully identified and clicked the correct button and then proceeded to investigate the scene using the mouse and the scroll on the mouse. The participant raised some questions such as how many exits there are and how the user is supposed to know where a particular exit is. 6. Completed successfully.

Participant 8 (ID:33621329) 1. The participant highlighted the text box and changed the population size without any difficulty. 2. The terminology of frame confused the candidate and wasnt sure whether she had done the correct thing by hiding and showing again the navmesh. 3. The participant was not sure whether the route button was the correct button to press. It was assumed by the participant to be correct after noticing that more things appeared on the screen. 4. The camera location drop-down was selected first and the participant selected exits to see what it did. This was the wrong thing to do and realising the error, the participant proceeded to press all the camera location buttons to try to complete the task. There was not much structure to the way the buttons were pressed. On discovery of the fact that the mouse could be used on the simulation canvas, the task was completed successfully. 5. The participant was not sure they had to click the move button to register the camera move event but eventually realised this had to be done. 6. Completed successfully.

Chapter 6

Conclusion

A great project!

6.1 Contributions

Here we explain that Lewis Carroll wrote chapter 1. John Wayne was out riding his horse every day and didn't do anything. Marilyn Monroe was great at getting the requirements specification and coordinating the writing of the report. Betty Davis did the coding of the kernel of the project, described in Chapter 4. James Dean handled the multimedia content of the project.

Bibliography

[1] http://en.wikipedia.org/wiki/Adaptive_behavior_(ecology).

[2] http://en.wikipedia.org/wiki/Proxemics.

[3] http://en.wikipedia.org/wiki/Social_proof.

[4] Katsuhiro Nishinari Andreas Schadschneider Michael Schreckenberg Ansgar Kirchnera, Hubert Klupfel. *Simulation of competitive egress behavior: comparison with aircraft evacuation data*. PhD thesis, Various Institutes, 2003.

[5] Gerd Gigerenzer and Peter M. Todd. Simple heuristics that make us smart. .

[6] Edward T. Hall. *The Hidden Dimension*. Bantam Doubleday Dell Publishing Group, 1966.

[7] Theodore Millon & Melvin J. Lerner Irving B. Weiner, editor. *Handbook of Psychology, Volume 5: Personality and Social Psychology*. John Wiley & Sons, 2003.

[8] Ken Dauber Kincho H. Law and Xiaoshan Pan. *Computational Modeling of Nonadaptive Crowd Behaviors for Egress Analysis*. PhD thesis, Stanford University, 2006.

[9] Stephanie Ludi. *Student Survival Guide to Managing Group Projects 2.5*. http://www.se.rit.edu/~sal/SEmanual/TableOfContents.html, 2006.

[10] Xiaoshan Pan. *COMPUTATIONAL MODELING OF HUMAN AND SOCIAL BEHAVIORS FOR EMERGENCY EGRESS ANALYSIS*. PhD thesis, Stanford University, 2006.

[11] Nick Chater Ramsey M. Raafat and Chris Frith. Herding in humans. *Trends in Cognitive Science. Vol. 13 No. 10*, 2009.

[12] Anna Grunebohm Tobias Kretz and Michael Schreckenberg. *Experimental study of pedestrian flow through a bottleneck*. PhD thesis, University of Dulsburg-Essen, 2008.

[13] Ken Dauber Kincho H. Law Xiaoshan Pan, Charles S. Han. *A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations*. PhD thesis, Stanford University, 2006.

[14] Kincho H. Law Xiaoshan Pan, Chuck Han and Jean-Claude Latombe. *A computational framework to simulate human and social behaviours for egress analysis*. PhD thesis, Stanford University, 2006.