KINDLING : A Game Platform for Crowd-Sourcing Fire Evacuation Data

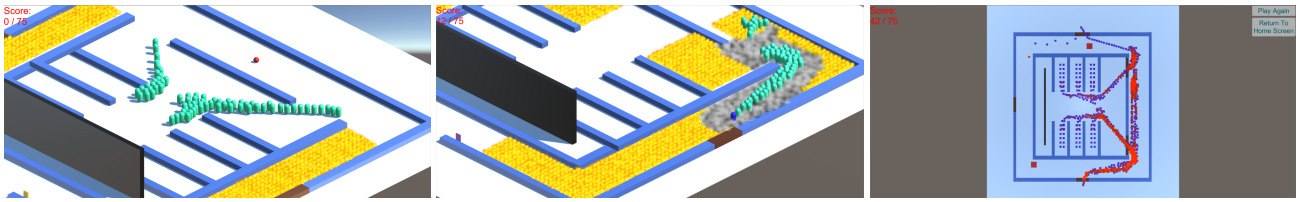

Figure 1: *Game-play from KINDLING. Agents attempt to escape from a fire in a user-designed environment.*

Abstract

KINDLING sets forth to make an engaging and intuitive platform for fire evacuation analytics. By modeling the problem as a game, real or imaginary buildings can be created as game levels with a level editor, and attacked by other players, providing a highly advanced worst case scenario. Attacking players use a limited number of fires to attempt the most efficient destruction of the simulated crowd in a level designed by another player. Scores are recorded and displayed much like any other video game. However, each attempt at the level collects data for important metrics like location of agent death, locations of destructive fires, crowd flow, etc. With this data represented as a heat-map overlay on the level, the creator can improve upon the level by adding precautionary measures to dangerous locations; such as an extinguisher to clear the way to an exit, or a fire door to hold back the spread of the fire. In this way, KINDLING provides both meaningful feedback on the evacuation safety of a real or virtual space, and evolving dynamic game-play between the level creator and other players.

Keywords: Crowd Simulation, Collaborative Content Creation, Environment Optimization

Concepts: •**Human-centered computing** → **Collaborative content creation;** *Interaction design theory, concepts and paradigms; Heat maps;*

1 Introduction

Evacuation of a burning building is an obviously important problem, but it is also a highly complex one. Many different variables can change the relative success of an evacuation, such as architectural design of the floor plan, inclusion of fire-safety equipment, and even proper signage.

Our motivation for this project is simple; to create a platform with which users can identify emergency escape weaknesses in both hypothetical and real-world floor plans and repair them, by taking advantage of the incredible resource of modern video game players on the internet.

To do this, KINDLING logs relevant game-play data to create visual representations of dangerous floor plans in fire emergency situations, allowing for easy identification of important changes that could be made to existing structures, such as ones own home or office building.

We identify the following challenges involved in implementing our solution, and address them individually in our implementation details below.

1. Design Complexity- There are an incredible number of relevant factors to the success of an escape, creating a very large potential space for designs.

2. Implementation Complexity- To be modeled as a video game, computation must be fast enough to create engaging game-play.

3. Agent Complexity- To create realistic behavior, agents will require individual knowledge domains and must detect surroundings autonomously.

While much work has been done to explore this problem, many previous solutions have tried to perform exhaustive searches on the huge search space that represents building design. In order to make this possible, most implementations keep variations simple among possible floor plans, testing the effectiveness of tiny changes like the placement of pillars or the size of doors, which leaves much to be desired in terms of architectural design. Architectural design is a problem that is ill-suited for an algorithmic approach, as it is difficult to quantify aspects like aesthetic, especially in tandem with other more mathematical quantities like efficiency of evacuation.

By creating a game which models fire evacuation, our implementation takes advantage of the cognitive power of video game fans everywhere. Levels are designed by human minds rather than algorithms, and are likewise challenged by gamers. This allows for humans to select traits that a normal algorithm might be unable to, like aesthetics. Likewise, we can increase the complexity of the simulation and the size of the search space nearly infinitely without needing to consider the complexity for a typical search algorithm.

This paper makes the following main contributions:

1. Inclusion of smart objects- Increase realism of simulation by adding numerous ”smart objects”. Fire extinguishers, fire alarms, signs, and dynamic environments add elements of real world fire prevention to the game.

MIG 2016, Oct. 10th-12th, 2016, San Francisco, USA
ISBN: 978-1-4503-ABCD-E/16/07
DOI: http://doi.acm.org/10.1145/9999997.9999999

2. Crowd-sourcing solutions- By effectively crowd-sourcing solutions with a state-of-the-art in-game level editor, and having the simulation actually played by humans, the need for a complex search algorithm is circumvented.

3. Human Decision-making- Because the optimization is performed by humans, unconventional solutions may arise that take other factors into account; such as aesthetic, theme, or personal use.

We suggest our proposed solution has the following main benefits over prior work. By implementing the many fire-safety devices that protect our modern homes and offices, we can more accurately simulate an evacuation which takes these affordances into account. Human decision-making provides a highly advanced worst case scenario for a fire evacuation, and also allows unique solutions to arise to optimization problems that take other factors into account; such as aesthetics, or other intentions. A floor plan could be designed with a particular intention in mind; for example, specific sizes of doors may be considered preferable to others by an algorithm, but someone designing a hospital building could purposefully choose an in-optimal door sizes so that hospital stretchers can fit through the frame. This same concept applies to other specifications as well, such as architectural style.

Intention is an important part of KINDLING , as it is human-powered. It's a game, and part of the fun is experimenting, and allowing a user to see how changes to the floor plan affect the evacuation. Users with a mind for architectural design may design fantastic, professional-quality floor plans. Users with a mind for legos may design elaborate castles or buildings that look like cartoon characters. KINDLING is not an automated approach, and does not promise professional-quality floor plans to users. It attempts to help the user approach a more-optimal solution by their own invention, a strategy distinct from most other optimization methods. In that respect, it is meant to offer an alternative to other optimization methods, not a replacement. When compared to results of traditional algorithms, we do not claim improved results, rather, alternative results that correspond to the users intention.

Furthermore, the solutions produced by this system are not applicable to real-world environments. In this regard, the game is more like a proof-of-concept. We only show that the system optimizes according to our incredibly simplistic simulation, however it stands to reason that the concept could be applied to a more realistic simulation, using the same framework model.

In the end, the main deliverable of the project is the video game, KINDLING , described in detail below. The game features any number of agents trying to escape a burning building, with the player in the role of the antagonist. By selecting locations to start fires, the "kindling" will take advantage of a suite of smart objects in the level to escape in the most effective way. In addition, data analysis provides meaningful data to improve players' ability to optimize their levels. Players will be able to iterate on their floor plans using the data collected, allowing for levels to be improved over time for increased challenge.

2 Related Works

Environment Optimization for crowds is very heavily researched subject. Environment optimization, especially in games, is a problem that involves many moving parts. The choice of path-finding algorithm is an important component of evacuation [Helbing et al. 2007], as well as many other aspects of the environment and the affordances of the evacuation agents.

Environment optimization is traditionally modeled by a search algorithm of some sort [Michalek et al. 2002], which searches a space for the optimal position of a particular variable factor of the environment. KINDLING circumvents this by replacing the need for a search algorithm with an intelligent player base, which has been shown to be an effective method of problem solving by the game FoldIt [Khatib et al. 2011], among others.

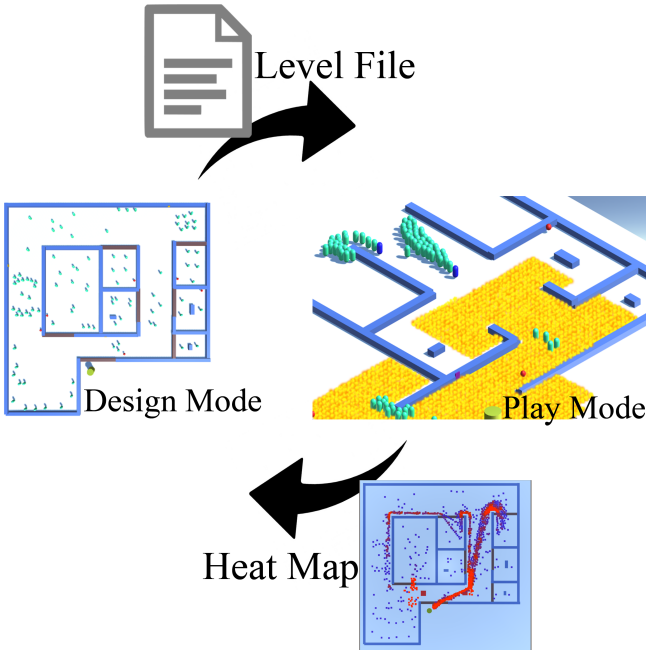

Figure 2: *Results of the System Effectiveness Study*

Level editing is a very important part of our project, and an interesting area of research as well. Baur et. al. explore a means to redesign levels to automatically [Bauer et al. 2013] adjust the difficulty of a game level. Michalek also explored the inclusion of human decision-making in layout optimization [Michalek 2001] as a means of quantifying more subjective elements of architectural design.

Many strides have been made in the field of A.I. and path-finding for evacuation. Effective and realistic steering in agents can be incredibly complex because it is a simulation of human behavior. These researchers provide a model for simulating psychological phenomena such as panic in a crowd [Helbing et al. 2000], which can help create a more lifelike simulation. A perfectly life-like simulation is ideal for a system which hoped to provide useful real-world data. Haworth et. al. discuss the importance of agent *level of service* in successful evacuation [Haworth et al. 2015], showing the relevance of ego centrism in simulations. Schuerman explores *situation agents*, which take command and alter the steering of agents within their sphere of influence, an aspect of human interaction between agents.

3 KINDLING Overview

KINDLING follows an iterative two-step design and play process, as illustrated in Figure 2. First, players use a custom level editor to create their own levels. Other players then test the newly created levels by playing the game. Finally, collected metrics from the play-testing are incorporated into the new iterations of the custom level, which enter the cycle to be play tested again. Below is a complete description of the different game modes, displayed

chronologically for a single iteration of the process.

Design Mode: Players use level modeling tools to design a floor plan that they wish to optimize. It may be helpful to imagine the player taking on the role of the protagonist in this mode, as the goal of this mode is to create a safe environment for the agents in the event of a fire evacuation. This level can be modeled after a real world environment, or an imagined one. The game offers a suite of interesting safety affordances, such as extinguishers, alarms, and doors. This allows for incredibly complex environments to be designed, and because KINDLING does not use an exhaustive search algorithm to solve the problem, such levels are entirely feasible. Once their game level has been created, it is saved and ready to be optimized.

Play Mode: Users then share their completed levels with other players, who take on the role of the antagonist. Given a limited ammo system, other players attempt to destroy as many agents as possible by setting fire to the environment. As the fire spreads, different parts of the level may become impassable, or an area may be flooded by agents as they race to escape from the spreading fires. The number of agents that fail to escape is used to calculate a score, which is added to the leader-board for the level. This provides a highly advanced worst-case scenario in a dynamic environment, where the system can collect data on the agents and the flow of the crowd in different locations. While only allowed a limited number of fires during a single simulation, players are allowed any number of individual attempts so they can improve and learn how to recognize weaknesses in floor plans.

Iterative User-in-the-loop Level Optimization: Back in design mode, the original player who uploaded the level now has access to these collected metrics inside the level editor, allowing them to iterate on the level. Players can take advantage of the information provided to them by the other players and make alterations to their level before re-submitting it to be played again. This cycle of gameplay continues, optimizing the environment further each time.

The back-and-forth mechanic of the game creates a competitive problem solving environment that makes for an engaging gameplay experience, as well a powerful solution to this heavily researched topic.

4 Play Mode

Play mode is a simple fire emergency simulation. Armed with two fire charges, the user is tasked with stopping as many of the agents as possible. The player can left click to start a small fire and consume one of their two charges. The fire quickly spreads outwards from the initial location, quickly consuming much of the floor space. The fire cannot burn through walls or doors, but any agent who makes contact with fire will die.

Agents will begin to evacuate as soon as they become aware of the fire. It can be detected by sight or sound. Agents then proceed to the nearest goal. In addition, they will use smart objects to aid their escape.

When all the agents are safe, dead, or trapped (where they have no viable path to the goal) the simulation ends. The user's score is calculated as the percentage of agents killed from the total. If a user's strategy is particularly effective, they may achieve a new high score, and the data collected during their run will be saved to the server for use in Design Mode.

5 Design Mode

Design mode gives players all the tools they need to create and edit game levels to make them as safe as possible. It is a very simple tool that contains the basic functionality necessary to do so.

The load and save functions are an important part of the system. Saving the level transforms the virtual space the user designed into an exportable file which is transferred to our test server. The load function allows users to download a file (designated by name) from the server, which is then read and built instantly.

The editing tools are what allows the user to create their own level. KINDLING offers on-screen buttons to add walls, agents and goals, as well as available smart objects, like the extinguisher and fire door. Another button allows users to delete objects from the scene, and players can rotate objects with an additional keyboard input. Finally, walls and doors can be stretched parallel to the ground plane.

The final, most important feature is the integration with play mode. The heat button toggles the heat-map overlay for the level. This heat-map displays agent location at discrete intervals of time during the simulation of the highest scoring run, as well as starting fire locations. Locations are colored according to how often they are traveled by agents. Blue locations (low temperature) are locations traveled only briefly by few agents. Red locations (high temperature) are locations that were traveled by many agents, often identifying bottlenecks or other particularly high traffic features. For shades of purple, the relative red-coloring corresponds to more traffic, while the blue-coloring corresponds to less traffic. In addition, users can play-test their levels by pressing the play button. This will simulate play mode and allow the level creator to test the basic functionality of their level.

6 Implementation Details

6.1 Environment Controller Grid

The entire game environment exists within a grid data structure built from individual node objects. Each node corresponds to a single point in the level, and stores data on that location. These nodes represent information in a form that is accessible to the Agent Controller, Path Manager, and other control modules. They can represent unwalkable terrain, burning terrain, normal terrain, and contain additional relevant info.

6.2 Agent Logic

The logic which controls the AI is a combination of two main modules, a state-based AI which controls higher level behavior patterns, and a navigation system which controls how and where agents move.

Agent Behavior Dynamic Agent Navigation and Collision-Avoidance The agent behavior is controlled by a probabilistic fuzzy state machine. The three main states are idle, escape, and panic. Transitions occur based on agent-specific knowledge each character individually collects. Agents have multiple methods by which they can receive information. Most obviously, agents can see things in a cone-shape in front of them (simulating front-facing eyes). In addition, they have basic auditory checks, which allow them to hear fire alarms. Idle agents do not attempt to escape, until sensory updates cause their state to change. Escaping agents attempt to follow a dynamically updated path generated by the A* path algorithm, similar to that used in [Yao et al. 2010]. Agents' panic level increases as the fire spreads, and when an agent has no available path remaining, its panic level increases dramatically.

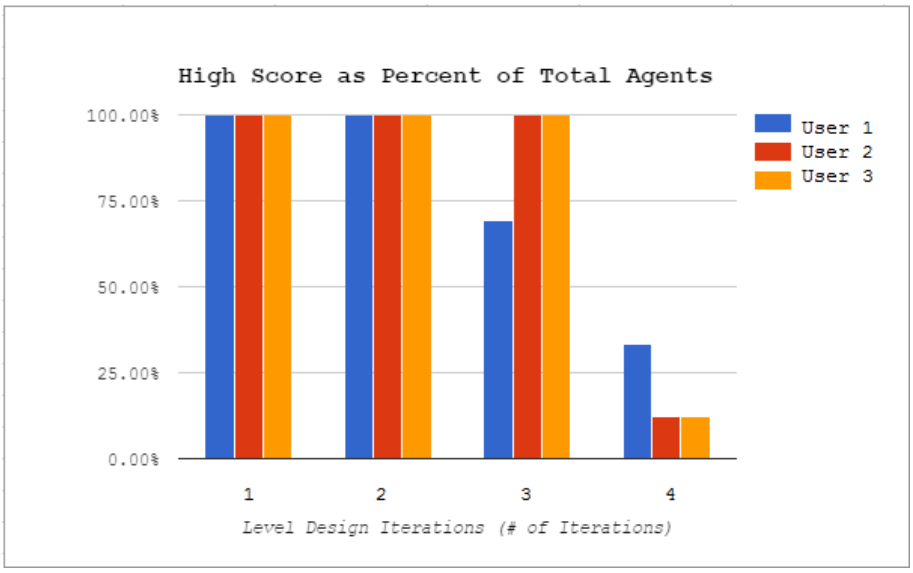

Figure 3: *Results of the System Effectiveness Study*

Panicking agents will brave the fire and use fire extinguishers to escape.

On start-up, the agents request a path from the path manager, which returns a hypothetically viable A* path. Agents use RVO, as in [van den Berg et al. 2008], for basic collision avoidance. When an agent encounters a fire that blocks its current path, it requests a new one from the path manager.

6.3 Path Request Manager

The Path Request Manager receives path requests from units as they are needed. The manager stores requests in a queue, which it resolves in FIFO order. Large volumes of requests are staggered across frames to avoid high latency with dynamic fires. The path request manager returns a viable path using the A* search algorithm.

6.4 Fire Propagation

Fire is propagated through nodes in the grid. Normal terrain will catch fire when adjacent to other flaming terrain. More adjacent fires will increase the speed at which fire spreads to any node. These changes are updated in the grid.

6.5 Smart Objects

Smart objects are special game items which are automatically used by the agents when trying to escape.

Doors are a special type of terrain which block fire spread but do not impede travel by agents. Agents ignore them when pathing, as though they do not exist.

Fire alarms alert all agents of the fire without needing to actually see it, so they can leave in a timely manner. They update the agents' logic to begin evacuation.

Fire extinguishers are a last resort used when no viable path can be found. Agents will instead try to path to a fire extinguisher, and use it to escape. We chose this implementation because it is likely that real people in an emergency situation would not fight the fire and put themselves in harm's way when another avenue of escape is available; they only use the fire extinguisher when they have no other way out. The fire extinguisher transforms impassable fire terrain into walk-able terrain, allowing other agents to follow. Agents will follow other agents who possess a fire extinguisher.

6.6 Level Editor

The level editor allows users to create levels. It simply instantiates objects such as walls, doors, and other smart objects to be placed in the scene by the user. In addition, it allows users to destroy, rotate, translate, and transform objects. Completed levels are saved to a web server, where they can be loaded later. Data collected during game-play is visualized in the editor to provide useful information to the level designer.

6.7 Web Server / Data Collection

Data is collected from the scripts in real time and saved locally. After the game is completed, it is uploaded to a server. The server receives the data via a simple PHP script and saves it to a text file. Each level has three files associated with it; one which contains the data needed to build the level, one which contains the data needed to build the heat-map, and one which contains the high score.

6.8 Data Visualization

Post-game data is visualized as a heat-map, with higher temperature indicating a higher density of agents over time. It also marks the locations of the original fires. This heat-map is stored on the server whenever a new high score is reached. To aid the user in intelligent optimization of their level, it is available at any time from the level editor.

7 User Study

We ran three studies to evaluate the usability of KINDLING and to test the validity of our hypothesis. These studies were conducted at an American university with a population consisting mostly of

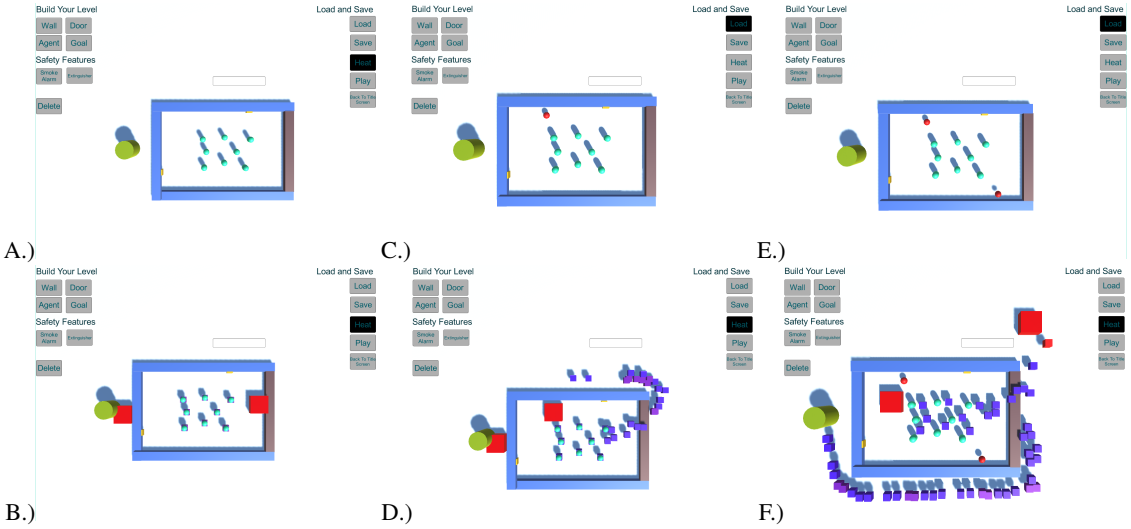

Figure 4: *A Basic Optimization Cycle*
Figures a-f display an average optimization cycle in KINDLING .
Each iteration of the level is displayed above(A,C,E) with its corresponding heat-map below(B,D,F).

graduate and undergraduate computer science majors. The population therefor consists mostly of male students age 20-25, who are likely more adept with computers and video games due to their studies.

Our first two preliminary tests were designed to test the concepts at play while the software was in-development. The final test more appropriately tests the effectiveness of the system. We hypothesized that given an environment optimization problem in the form of a game level, users would naturally identify weaknesses in environment layouts as they play the game, and metrics collected from the game would provide meaningful insight into optimization of a given environment. As such, we expect that over multiple iterations of level design, the score users are able to achieve should decline as the creator is given access to simulation data from players.

Independent Variable: Iterations of Game Level
Dependent Variable: High Score

7.1 User Study 1 : Identifying Weaknesses

Population Size: 10 The first user study aimed to answer one question: "Are players able to naturally find weaknesses in floor plans as they play?" While the sample size was small, and some of the data was too incomplete to be meaningful, we found conditional support for the hypothesis. Some players showed an obviously increasing trend in their scores, suggesting that their ability to identify weaknesses in these levels improved as they played.

7.2 User Study 2 : Effects of Smart Objects

Population Size: 20 Our second user study tested another question: "Does the addition of smart object safety affordances in level editing show a noticeable change in the safety of the environment?" To answer this we collected data on the scores of the users, with relatively high scores indicating a dangerous environment (as users were able to kill many of the agents) and low scores indicating relative safety. The data showed, with some certainty, that the inclusion of these smart objects (fire extinguishers, fire doors, etc) lowered the score by an obvious factor. The factor was not consistent across levels, as some safety features were more effective on one level than another, but nonetheless, scores showed a noticeable decrease.

7.3 User Study 3 : System Effectiveness

Population Size: 20

Our third user study tests the effectiveness of the whole project, and answers the question, "Does the feedback provided by user play show a meaningful increase in the relative safety of an environment when given to level editors?"

The study was carried out by giving 3 users instructions to build a level, after allowing them to familiarize themselves with a practice level.The other users played their level as the antagonist, and produced heatmap and score data for the newly created levels. This was repeated until the users were happy with their results, as it would be in a real use case of KINDLING .

Unfortunately, our user study was limited by time and available test population, but preliminary results show that designers with the feedback from the data visualization tool were able to successfully lower their average high scores over time. In our test, every level saw over a 50% decrease in high score by the end of the fourth phase, as seen in Figure 3.

An example from our user study is displayed in Figure 4, chosen for its simplicity, but also because it shows the user's problem solving fairly obviously. Image a shows the first iteration of the level, and image B shows the heatmap. Without any safety features, the agents didn't even respond to the fire fast enough to leave the building before becoming engulfed in flame. In image C, the author added a fire extinguisher, to prevent a similar event. However in image D, the high scoring player chose to place their fires on the extinguisher and near the goal, preventing the agents from using the extinguisher to reach the goal. In image E, the user added another fire extinguisher to the building, so even if one is blocked, the other is still available. This proved to be an effective strategy and lowered the high score from 100% to only 33%.

Another example of the process in action is displayed in Figure 5. This level is much more complex, and begins to actually resemble

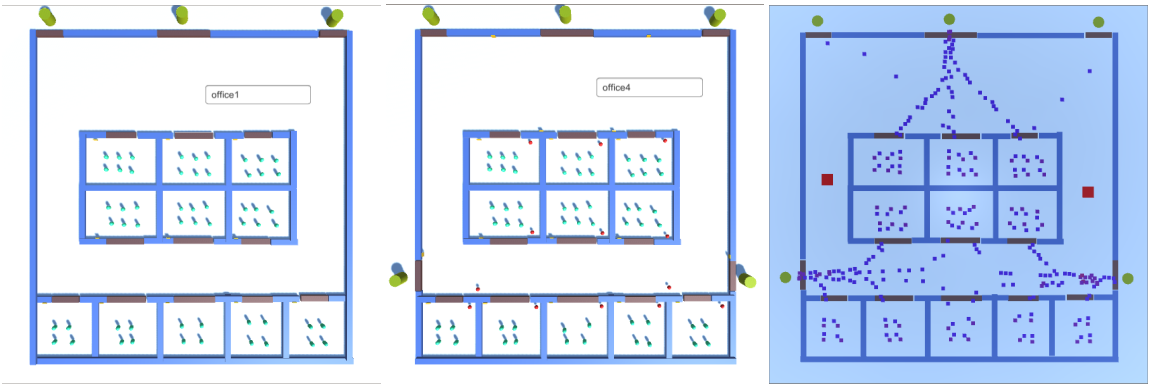

Figure 5: *A More Complex Environment*
Before, after, and the final heatmap of a more complex level.

a realistic floor plan for a small hotel type building. As iteration on the level continues, time-tested safety features begin to appear; an alarm in each room to warn occupants of fire, an extinguisher in each room so agents don't become trapped in the room, and even fire exits in the rear of the building. This level showed remarkable improvement, going from a 75% death rate to just 4%.

7.4 System Usability Study

In addition to the user study, a System Usability Study (SUS) was conducted via post-game survey. The SUS is a 5-point Likert scale designed to act as an unbiased way to collect data on the usability of a system. A modestly sized population of 15 students were polled after playing an in-development version of the game. The system received an 82.17 SUS score, indicating a high level of basic usability despite not having access to any explanatory materials or tutorials.

8 Conclusion

KINDLING is modeled to take advantage of a large community of online gamers. As such, for its true functionality to be explored, it would need a study on a much larger scale where the competitive nature of gamers would push players to optimize levels to their true limits. While our study has shown that this is conditionally true in a small test group, it would be much more meaningful if the population was expanded by a factor of 100, at least.

By incorporating human intelligence, the process leaves room for human variation, in terms of aesthetic and expression. Looking at Figure 6, it's clear that this level is not yet optimized, but also, it's some sort of weird dog face. And when the user realized that a single exit left the agents unable to escape in most simulations, they made an even sillier change; a top-hat emergency escape room.

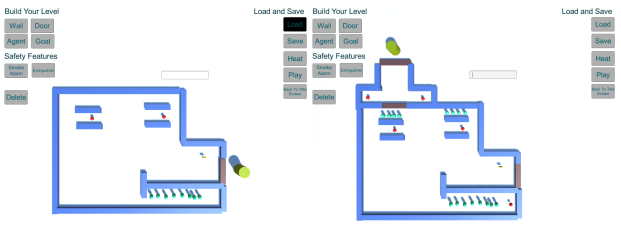

Figure 6: *Interesting Results*

Almost unbelievably, the addition of this top-hat reduced the death rate in the dog-face room by over 75%. This behavior is incredibly relevant in a problem like architectural optimization, where solutions regularly ignore aesthetic features of an environment in favor of more optimal ones. This element of KINDLING is truly unique among technological solutions in this field.

Limitations The success of KINDLING relies entirely on its users. Therefore, the project is limited by the willingness of users to participate as the engine by which levels are created and tested. In addition, any level's success is dependent on the user's intention, and relies on human problem solving skills- something that may be in short supply for young users that may be attracted to the lego-like nature of building and destroying houses.

Furthermore, fire escape simulation is a very complex dynamic simulation, and the version KINDLING offers is very basic in nature. While our results show the system does optimize levels, the simulation is so basic that the data is not currently applicable to real world environments without much more work. In real fire emergencies, smoke inhalation is a major component to the danger which our simulation doesn't include at all.

Future Work Many other features could provide a more realistic simulation and more useful resultant data. These aspects include both physical simulation components of fire spread, sound propagation/localization, etc. as well as components of agent psychology like panic. Improvements to agent intelligence could be made based on [Kapadia et al. 2009], where agents are given vector-based affordance fields. Agent steering is incredibly naive in our implementation, when much more advanced options are available in terms of complexity [Kapadia et al. 2013], speed [Singh et al. 2011], and quality. Agent intelligence could be increased further by allowing designers to use a job system for individual agents; special agents could change the flow of the crowd (like situation agents [Schuerman et al. 2010]) or even have special affordances available to them.

In addition, there is plenty of room for growth in the game aspect as well, as the data can't be collected if the game isnt considered enjoyable enough to play. Complexity in user levels could be detected automatically [Berseth et al. 2013] and used to dynamically alter the difficulty of the game (by providing more ammunition). Many of the other improvements to the AI would also increase the fun of the game, as well as the realism of the simulation.

References

BAUER, A. W., COOPER, S., AND POPOVIC, Z. 2013. Automated redesign of local playspace properties. In *FDG*, 190–197.

BERSETH, G., KAPADIA, M., AND FALOUTSOS, P. 2013. Steerplex: Estimating scenario complexity for simulated crowds. In *Proceedings of Motion on Games*, ACM, New York, NY, USA, MIG '13, 45:67–45:76.

HAWORTH, B., USMAN, M., BERSETH, G., KAPADIA, M., AND FALOUTSOS, P. 2015. Evaluating and optimizing level of service for crowd evacuations. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, ACM, New York, NY, USA, MIG '15, 91–96.

HELBING, D., FARKAS, I., AND VICSEK, T. 2000. Simulating dynamical features of escape panic. *Nature 407*, 6803, 487–490.

HELBING, D., JOHANSSON, A., AND AL-ABIDEEN, H. Z. 2007. Dynamics of crowd disasters: An empirical study. *Physical review E 75*, 4, 046109.

KAPADIA, M., SINGH, S., HEWLETT, W., AND FALOUTSOS, P. 2009. Egocentric affordance fields in pedestrian steering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, ACM, 215–223.

KAPADIA, M., BEACCO, A., GARCIA, F., REDDY, V., PELECHANO, N., AND BADLER, N. I. 2013. Multi-domain real-time planning in dynamic environments. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, 115–124.

KHATIB, F., DIMAIO, F., COOPER, S., KAZMIERCZYK, M., GILSKI, M., KRZYWDA, S., ZABRANSKA, H., PICHOVA, I., THOMPSON, J., POPOVIĆ, Z., ET AL. 2011. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature structural & molecular biology 18*, 10, 1175–1177.

MICHALEK, J., CHOUDHARY, R., AND PAPALAMBROS, P. 2002. Architectural layout design optimization. *Engineering optimization 34*, 5, 461–484.

MICHALEK, J. J. 2001. *Interactive layout design optimization*. PhD thesis, University of Michigan.

SCHUERMAN, M., SINGH, S., KAPADIA, M., AND FALOUTSOS, P. 2010. Situation agents: agent-based externalized steering logic. *Computer Animation and Virtual Worlds 21*, 3-4, 267–276.

SINGH, S., KAPADIA, M., HEWLETT, B., REINMAN, G., AND FALOUTSOS, P. 2011. A modular framework for adaptive agent-based steering. In *Symposium on Interactive 3D Graphics and Games*, ACM, PAGE–9.

VAN DEN BERG, J., LIN, M., AND MANOCHA, D. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 1928–1935.

YAO, J., LIN, C., XIE, X., WANG, A. J., AND HUNG, C. C. 2010. Path planning for virtual human motion using improved a* star algorithm. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, 1154–1158.