vFireVI: 3D Virtual Interface for vFire

Christopher Lewis
Computer Science and Engineering
University of Nevada, Reno
Reno, United States
christopher_le1@nevada.unr.edu

Ronn Siedrik Quijada
Computer Science and Engineering
University of Nevada, Reno
Reno, United States
ronn.quijada@nevada.unr.edu

Frederick C Harris, Jr.
Computer Science and Engineering
University of Nevada, Reno
Reno, United States
fred.harris@cse.unr.edu

***Abstract*—Wildfires cause severe amounts of damage to wildlife habitats and property. The most successful way of escaping safely or quelling a fire is to do so with communication, and as a group. While there are several other wildfire simulators, visualization and multi-user components are lacking or non-existent in most of them. vFireVI aims to provide a safe and accurate virtual environment for simulation and interaction with wildfires through multi-user collaboration, accurate terrain, and realistic fire spread. In order to do this, an interface was built between vFireVI and an earlier project, vFireLib, to allow transmission of simulation data back and forth. Combining these two allows for an intuitive user interface, responsive multiplayer, quick server communication, and rapid simulations. All of this is done in virtual reality to provide a meaningful and immersive experience, where users can collaborate and test each other.**

***Index Terms*—fire, simulation, virtual reality, multiplayer**

I. Introduction

Occupations that operate in dangerous conditions on a daily basis, such as military and emergency settings, face the issue of training employees for these dangerous conditions. Subjecting them to equally dangerous training exercises increases risk of injury, so other methods of training that provide a similar experience to on-site work are preferred, due to the minimal risk involved. In recent times, the growing popularity of consumer head mounted displays (HMD) has made Virtual Reality an increasingly viable option to provide minimal risk training.

In the case of forest fires, their immense scale and unpredictable nature make physical analogs costly and dangerous, thus having a virtual analog that has a minimal and flat cost is very efficient and useful. Using the virtual 3D environment provided by Unity, paired with an immersive HMD we created a safe and viable option for providing minimal risk fire safety training in optionally cooperative situations.

Virtual reality (VR) is fantastic at conveying a sense of immersion and user enjoyment; however, VR can also cause some users discomfort and detachment from reality. Through the use of Unity, VR is carefully implemented to provide a realistic 3D environment, and visualizer for the simulation. With minimal real world movement by the user, moving throughout the 3D environment reduces VR discomfort and motion sickness, which makes teleportation a good mode of locomotion for the 3D landscape, while still maintaining the user's autonomy while moving around.

The rest of this paper is structured as follows: Section II talks about the background behind this project, more specifically vFire and it's successors, and then related works, which are mostly fire simulators. Section III is where the design and implementation of vFireVI gets explained in detail, from the implementation of Unity to the Burning Simulation itself. Section V is Conclusion and Future work, which is where the paper shows its findings and explains future work that could come from this project.

II. Background & Related Works

A. Fire Simulators

Another fire simulation was used for wildland two dimensional fire spread and was made by Finney at *et al.* [1]. This simulation approaches fire simulation through a unique perspective. All of it's data is based off of historical U.S. data. This includes weather patterns, wind speed, and moisture. It uses this data and then compares it against data from 91 fires occurring from 2007 to 2009. Their results consistently had fire sizes higher than the real fire, and consistently smaller farthest burn distance than the real fire.

H. Xue *et al.* [2] made a fire simulator that compares different combustion models in enclosed simulations. These combustion models are the volumetric heat source model, the eddy break-up model, and the presumed probability density function model in 3 situations; a room fire, a shopping mall fire, and a tunnel fire. Comparing each set of data, H. Xue *et al.* found that none of the models are consistent over each situation. They suggest that there is a need for adequate turbulent combustion models.

H. S. He *et al.* [3] created a fire simulator that discusses the effects of fire on wildlife. The fire simulator incorporates fire, wind-throw, and harvest disturbance in terms of species-level fauna to determine patterns over large spatial and time domains. This simulator; however, does not predict individual events, it predicts the future of the species-level ecosystem.

B. vFire

vFire was a simulation developed by Hoang *et al.* [4], [5] in 2010. The simulation utilized a four-sided and six-sided CAVE™ virtual interface was used to display a reconstruction of the terrain near Kyle Canyon, Nevada. The simulation also used a UI to simulate and modify various forest fire situations and terrain data. The height map and vegetation data, used in the simulation, can be seen visualized in Figure 1 and Figure 2, respectively. One of the largest issues with the application was that both the underlying simulation system and virtual interface were highly coupled, meaning that in order to update the fire model used, the entire application would need to be updated in order to use the new model. Another large issue with the application was that the virtual interface was created in OpenGL before OpenGL implemented pre-shaders, which caused huge calculations in any simulation, and made the project time consuming and difficult to upgrade or work with. In addition, if stakeholders would want to update the visuals, it would require a rebuild of the entire system. This would be true even if the underlying simulation had not changed.

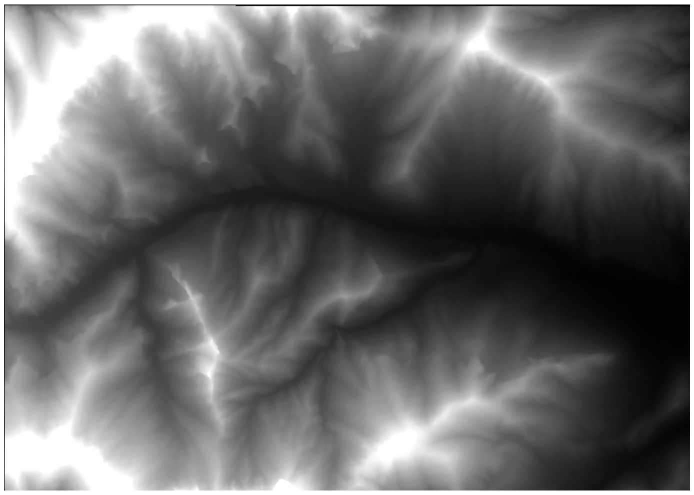
Fig. 1. A height map generated using data retrieved from vFireLib

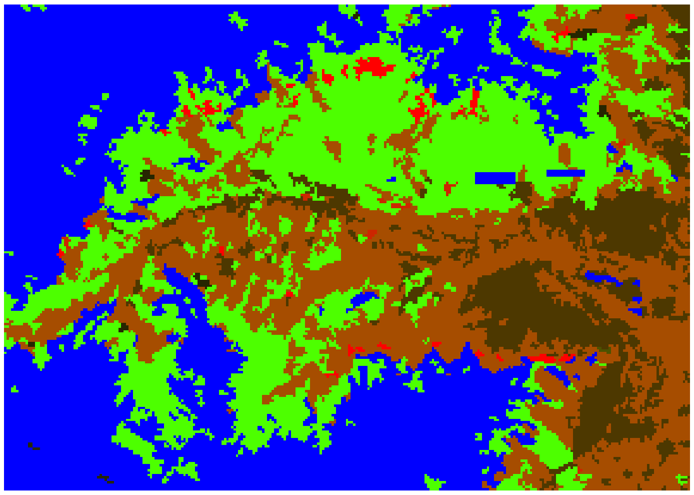
Fig. 2. Vegetation map generated from the vFireLib fuel load index map; each color representing variable attributes.

C. *vFireLib*

Due to the restrictions of the vFire simulation, a new system called vFireLib was created, in part by J. E. Smith [6], R. Wui *et al.* [7] , and R. Wui *et al.* [8], to decouple the fire simulation simulation from the visual interface. vFireLib reinvents the original functionality of vFire as a RESTful interface, allowing the direct upload of fuel load, wind, and initial fire data to initialize the simulation. The simulation finishes by creating the resulting time of arrival map. This map provides data about what time each cell will set on fire and is returned from a REST call. This data can be used in any application. The project Harris *et al.* [9] developed a useful web interface, providing tools to modify and simulate various wildfires using modern browsers An example of the interface can be seen in Figure 3, which is a pixel map of the land the simulation is ran on. Two of the larger issues with vFireLib was that the simulation and web interface did not allow for fire spotting and the simulation was also rather slowed down by the addition of the interface.

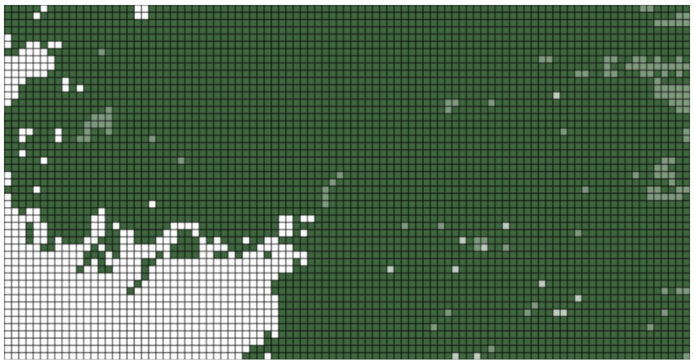
Fig. 3. The visual interface, in VFireLib, corresponding to the area the simulation was ran

In order to rectify the issues of vFireLib, and provide the user using the web service or Unity application, the ability to modify and run fire simulations, vFireLib.v2 was created by Garcia [10] and influenced by [11]. vFireLib.v2 was rewritten to increase the speed of the sequential algorithm by implementing a parallel algorithm that was processed by the GPU. The simulation also included fire spotting computation. Implementing spotting required computing where an ember would emerge once a fire had reached a hot enough intensity, and where that ember might land considering wind, gravity, moisture, and fuel data to determine if a new flame would ignite.

D. *VR Simulators*

VR Simulators can be found in many different forms. Much of the research in this area is in surgical applications. A unique example of this is by J. M. Albani *et al.* [12], which details the work of VR assisted robotic surgery simulations throughout the years. It also incorporates one of the current commercial surgical system, the da Vinci^{TM}robot. The article also describes possible future applications of the simulations and the remaining challenges that need to be fixed.

Another VR simulator that fits into the field of surgery can be found in an article by A. G. Gallagher *et al.* [13], which describes the times in which a VR simulation is actually helpful in teaching and the reasoning on what makes it useful. It describes the use of simulations for minimally invasive surgery only, but it could be used as a training tool for other types of surgeries as well. Their conclusion and results specifically make the case that VR simulators are only impactful when integrated into an already good education or training program that involves actual technical skills.

A fire simulator in virtual reality also exists, it was created by M. Cha *et al.* [14]. This simulator uses computational fluid dynamics to calculate certain quantities: toxic gases, heat, smoke, and flames. The paper's focus is on creating a training simulation, so that civilians, members of the military, and new firefighters can experience wildfires second-hand. Overall, the study provides a clean framework for accurately calculating quantities of fires, and inputs into a simulation. There isn't any spreading of the flames and the paper discusses using this framework for building sized situations, like evacuating, so that the fire not accurately spreading, isn't a large deal.

III. Implementation

As seen in Figure 4 there are many components to the overarching simulation of the modern vFire. vFireLib is the starting point, where the actual simulation gets done. The next step, Web Service, is the REST interface that exists to communicate and run vFireLib. From the Web Service, two clients are split off that communicate with the REST interface found within the Web Service. The first being the Web-based Client. This client exists online to communicate with the server. The other being the Unity Client. This client is vFireVI. The reason this client exists is to create a visualization of the simulation that is user friendly and can act as a realistic environment to help bolster the avoidance and preparedness against wildfires.

Due to how vFireLib fire simulation discretizes its simulation data to a grid, the REST interface provides a 2D array of several components of the fire simulation, ranging from wind data to vegetation type and density. vFireVI takes advantage of the highly parallel nature of this forest fire simulation data by offloading most of the work to the GPU through Unity's ShaderLab language and OpenCL Compute Shaders.

A. Unity

Unity is then used to show that data in a 3D environment by stepping through the burn chart, obtained through the Rest Interface, which is obtained by the server running the simulation with new data. Next, the simulation must spawns the players and ticks through time, incrementing through the burn chart until nothing is left. Once the burn chart is empty, the simulation ends. Unity is also used to implement the 3D environment in virtual reality.

Mirror, a plugin for Unity, is use to implement a multiplayer aspect to the simulation. The multiplayer aspect of the simulation was implemented through the use of a client-host model.

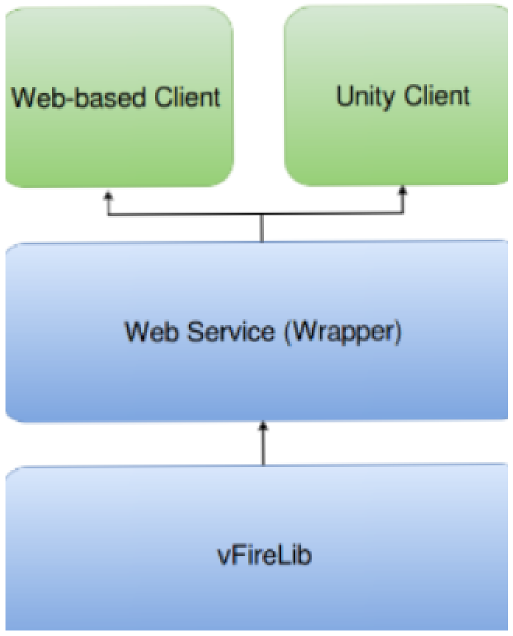

Fig. 4. The connection between projects developed for vFireLib

This is where the first one connected to a multiplayer session acts as a server or host for the rest of the players/clients and is a client themselves. If the simulation has to be changed at all, the host receives the information from the REST interface and updates the host's terrain. Once the terrain has finished updating for the host, that terrain is sent to all of the clients the host is connected to. Once this is done, the simulation continues. All stepping through the burn map is done from the side of the client, so the terrain wouldn't have to be sent every tick. The movement of clients gets sent to the host, the host then updates its models, and sends the updated locations back to all of the clients this creates some latency, but not enough to hinder the simulation. All of this project can be visualized as the Unity Client side of Figure 4 from R. Wui *et al.* [7] which references the connection between this project, vFireLib [6]–[8], and vFireLib2 [10].

B. In Game

vFireVI utilizes 2 different user roles. The first being the user role, ”Runner”. A Runner, as the name implies, runs around the environment and can jump as their only movement and interactive options. The second user role is the ”Overseer”. An Overseer can move more quickly than a Runner, and can fly as their modes of transportation. Another feature of the Overseer is that any Runner can't see the Overseer. This is to allow for an environment where someone can watch a user, from another room, without the user knowing about it. The locomotion and UI for the players were carefully decided through examination and review of multiple research papers, such as M. Nabiyouni *et al.* [15] and M.M. Davis *et al.* [16]. It was settled to use a radial menu and for locomotion, the motion of pulling yourself along the terrain for the Overseer and Teleportation for the runner as they provide, generally, the best mix between possibility of motion sickness and usability.

As for the situation deployment, the program starts with a low opacity 3D interface and basic terrain in the background. This user interface has options that allow the user to join multiplayer, change aspects of the fire simulator, and change the terrain through various tabs, windows, and input fields. After the user completes setup and starts the simulation, the custom data selected by the user gets sent to the REST server to be processed by the server and then the burn map is returned back. Then, Unity runs scripts to load the terrain and the burn line from the available data. Both types of players, Overseers and Runners, spawn in after they load into the simulation and the 3D environment has been built.

C. REST Data Parsing and Terrain Generation

The REST interface reinvented by vFireLib provides plain text files containing data of the simulation via an array of integers; however, a 2048 by 2048 cell simulation has over 4 million entries to parse. To speed the parsing stage up, a compute shader is used to split the data into lines and parse each row into the buffer that will contain the final array of integers. From there, we can pass the final data back to the CPU or transform the array into a texture that can be used in rendering the 3D environment. This texture is then passed to the Terrain Generator.

Terrain generation is primarily handled using Unity's built-in terrain system, which provides functionality for setting height data, trees, and various other details. For the height of the terrain, the parsed data is transformed into a 16 bit integer array like in Figure 1, which is then passed to the terrain system. While there isn't a clear way to directly replace the height map contained in the terrain system, Unity provides functions to replace the height map with an array of integers via C# scripts.

For the terrain material, Unity uses what is called a splat map, which is a texture that defines what ground material is used at which point in the terrain using the RGBA channels to represent the strength of each ground material. A custom shader was written to take advantage of this existing functionality by using the RGBA channels to describe generic variable attributes of the terrain. An example of the splat map can be seen in Figure 2, where blue controls unburnable areas, green controls the fuel density of the area, and red controls the fuel moisture.

D. Forest Generation

For forest generation, the project uses a 16 bit integer map generated from the previously parsed data. The data holds indexes that represent a terrain type. Terrain types are defined by the data obtained from U.S. official land surveys. The indexes in the data are compared to a user defined dictionary that pulls the fuel load density and tree type, and then spawns an instanced tree at the location on the terrain relative to where it was sampled on the index map. This system allows for variable forest types, allowing it to simulate a wider range of flora.

Forest generation requires different types of vegetation *e.g.* shrubs, trees, and grass. Thus, a great deal of importance is placed on allowing for these different types, and having example figures for each. vFireVI has a model for each type of tree specified from U.S. official land surveys. This means that any new vegetation type can't be read into the simulation until a new model is created and tied to the specific representation of the vegetation data.

E. Burning Simulation

The Data for when the fire arrives is calculated by vFireLib and is passed to our visualization through the REST API. A sample of the burn map (which shows the time of fire arrival) for Kyle Canyon can be seen in Figure 5

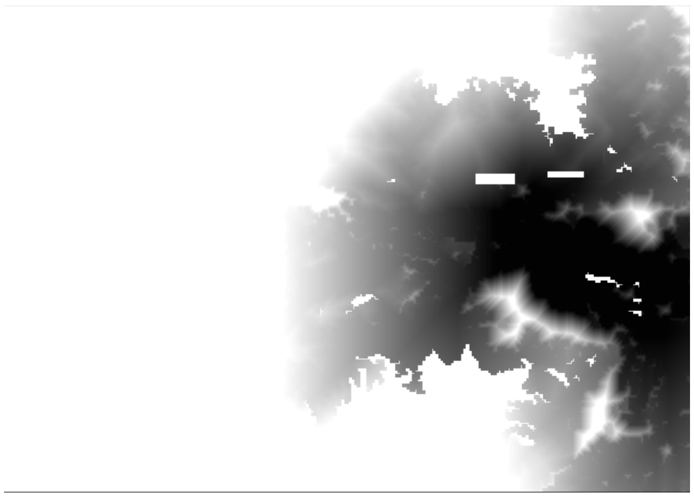

Fig. 5. The burn map generated from a successful run of vFireLib's simulation. float values are passed in and saved into an exr format to preserve the 32bit floating point time of arrival information.

The visual burning of the forest is handled through a set of custom shaders built on top of the existing Unity shaders that effect the rendering for all visual elements in the scene. For all the custom shaders, a map called time-of-arrival, which contains data pertaining to the exact time each cell sets on fire, is referenced in order to determine the visual state of the fire simulation. A sigmoid function is used on the sampled texture to convert the float value of an area into a value that represents its current burning state given the current simulation time. This function is then used to determine the visual aspects of the area.

For trees, the burn state function is used to visually set the tree on fire, and fade away small branches and leaves based on its current burn state. To support varying burn times of different forest types, a secondary map is used to store the burn color and burn time for each tree, assuming trees don't migrate. The code for the rendering is split into two different shaders handling the tree trunk and leaves respectively.

For terrain, the burn state function is used to visually show the fire line at a given time, represented by a glowing line. The thickness of the line is determined by the speed at which the area sets on fire. The color of the fire line changes based on the type of fuel being burned, as well as other user defined traits. The terrain texture will change based on the burn progress as well, where burned areas change from their original material to a burned rubble material.

IV. Example Simulation

An example data set was given from the aforementioned vFireLib simulator so that this project could continue where it left off. This data set is of Kyle Canyon, Nevada as seen in Figure 6. Along with the location data we were given terrain height data as seen in Figure 1, local vegetation data as seen in Figure 2, and the vFireLib simulator itself had moisture and wind data contained within it.

Fig. 6. A satellite view of the simulated area in Kyle Canyon, pulled from Google Maps.

vFireVI ran the simulator and took in the data received from the simulator to create the 3D representation of Figure 5. This 3d representation is shown in Figure 7. The strong bright yellow line represents the current burn line. On the left side of this figure there are representations of already burnt out trees, on the right there are representations of the not yet burned trees. There are also places on the left side with non burnt trees. This is due to there being terrain data that represents areas that can't be burnt overlapping with vegetation data that shows vegetation in that area. This is due to poor data rather than imperfections in the simulation.

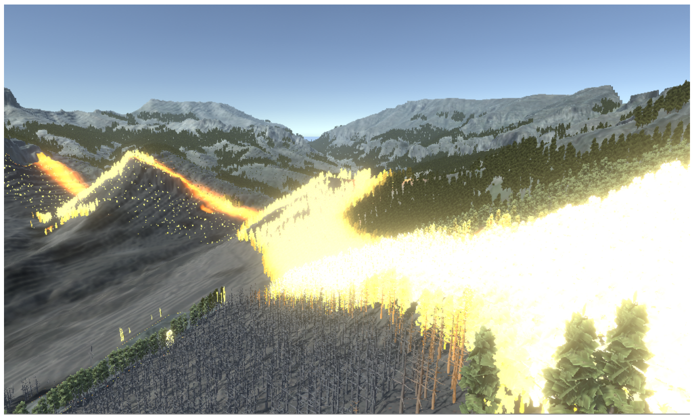
Fig. 7. The fire line of the simulated area in Kyle Canyon, visualized by the Unity simulation.

V. Conclusions & Future Work

A. Conclusions

vFireVI is a virtual reality based visualization environment of a fire simulation. The simulation can be communicated through a REST interface server from vFireLib which allows for quick changes to the environment and rapid simulations. Virtual reality makes this simulation, and others like it, viable options for training of all sorts, at all levels. The physical analog provides little safety, especially compared to the minimal risk of the head mounted displays. vFireVI also allows for responsive multi-user simulations. The virtual reality element tied in to the multi-user simulation allows for collaborative training which makes it even more effective.

B. Future Work

Dynamic Collection of Data: The way vFireVI gets data currently is using presets that already have correct data stored within it. These data sets could be scraped from a terrain database website and then formatted to allow for vFireLib to parse the data.

Fire Textures: vFireVI allows for dynamic texture swapping; however, there are no accurate textures available for fire or smoke representation within the simulation. The current textures are just seen as glowing and then the terrain textures are swapped from ”not burnt” to ”burnt”.

Role Functions: Currently, vFireVI has two roles, the ”Overseer” and the ”Runner”, these roles only really change the movement modes of the user. These roles could be expanded upon to allow for meaningful differences between the two. There could also be an increased game element to the simulations which might cause the user to be more invested in the simulation.

Better Multi-User: Unity recently removed support for multiplayer games on their platform. vFireVI was made using a Unity plugin called ”Mirror”. This causes the multiplayer to be a little slow and it is peer to peer. It would be better, in some cases, to create a server to peer system to allow for less lag and less issues with server or peer updates. It would also be better to use a multiplayer system that is native to the game engine environment.

Editing Data: vFireVI uses data given by vFireLib to run. There are two ways this could be improved. The first is letting a user edit the data directly in the Unity user interface. Another is to allow the user to edit data live during the simulation, perhaps as the ”Overseer” role. Data that could be edited includes, fire start point, additional fire points, vegetation, rain fall, and wind. This would provide a good way to allow friendly user editing of the simulation without having to look at the raw ASCII files.

VI. Acknowledgment

We would like to thank Rui Wu, Connor Scully-Allison, and Andy Garcia for their help in operating their past implementations of the web interface for vFire, and providing us with their simulation and simulation data.

This material is based upon work supported in part by the National Science Foundation under grant number IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] M. A. Finney, I. C. Grenfell, C. W. McHugh, R. C. Seli, D. Trethewey, R. D. Stratton, and S. Brittain, “A method for ensemble wildland fire simulation,” *Environmental Modeling and Assessment*, vol. 16, pp. 153–167, October 2011. https://www.fs.usda.gov/treesearch/pubs/39311, Last Accessed (1/2/2020).

[2] H. Xue, J. Ho, and Y. Cheng, “Comparison of different combustion models in enclosure fire simulation,” *Fire Safety Journal*, vol. 36, no. 1, pp. 37 – 54, 2001. https://doi.org/10.1016/S0379-7112(00)00043-6.

[3] H. S. He and D. Mladenoff, “Spatially explicit and stochastic simulation of forest landscape fire disturbance and succession,” *Ecology*, vol. 80, pp. 81–99, January 1999. https://www.fs.usda.gov/treesearch/pubs/12251, Last Accessed (1/2/2020).

[4] R. V. Hoang, M. R. Sgambati, T. J. Brown, D. S. Coming, and F. C. Harris Jr, “Vfire: Immersive wildfire simulation and visualization,” *Computers & Graphics*, vol. 34, no. 6, pp. 655–664, 2010.

[5] R. V. Hoang, J. D. Mahsman, D. T. Brown, M. A. Penick, F. C. Harris, and T. J. Brown, “Vfire: Virtual fire in realistic environments,” in *2008 IEEE Virtual Reality Conference*, pp. 261–262, March 2008. doi:10.1109/VR.2008.4480791, Last Accessed (1/2/2020).

[6] J. E. Smith, “vFireLib: A forest fire simulation library implemented on the gpu,” Master’s thesis, University of Nevada, Reno, Reno, NV 89557, December 2016. https://www.cse.unr.edu/~fredh/papers/thesis/064-smith/thesis.pdf (Last accessed: 5/2/2019).

[7] R. Wui, C. Scully-Allison, C. Carthen, A. Garcia, C. Lewis, R. Siedrik Quijada, J. Smith, S. M. Dascalu, and F. C. Harris, Jr, “vFirelib: A GPU-based fire simulation library and fire data visualization,” *Submitted*, 2019.

[8] Rui Wu, C. Chen, S. Ahmad, J. M. Volk, C. Luca, F. C. Harris, and S. M. Dascalu, “A real-time web-based wildfire simulation system,” in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pp. 4964–4969, Oct 2016. doi:10.1109/IECON.2016.7793478, Last Accessed (1/2/2020).

[9] F. C. Harris Jr, M. A. Penick, G. M. Kelly, J. C. Quiroz, S. M. Dascalu, and B. T. Westphal, “V-FIRE: Virtual fire in realistic environments,” *The 4th International Workshop on System/Software Architectures*, vol. 1, pp. 73–79, 2019.

[10] A. M. Garcia, “An advanced wildfire simulator: vFirelib.v2,” Master’s thesis, University of Nevada, Reno, Reno, NV 89557, December 2018. https://www.cse.unr.edu/~fredh/papers/thesis/074-garcia/thesis.pdf (Last accessed: 5/2/2019).

[11] J. Smith, L. Barfed, S. M. Dasclu, and F. C. Harris, “Highly parallel implementation of forest fire propagation models on the gpu,” in *2016 International Conference on High Performance Computing Simulation (HPCS)*, pp. 917–924, July 2016. doi:10.1109/HPCSim.2016.7568432, Last Accessed (1/2/2020).

[12] J. M. Albani and D. I. Lee, “Virtual reality-assisted robotic surgery simulation,” *Journal of Endourology*, vol. 21, pp. 285–287, April 2007. https://doi.org/10.1089/end.2007.9978, PMID: 17444773, Last Accessed (1/2/2020).

[13] A. G. Gallagher, E. M. Ritter, H. Champion, G. Higgins, M. P. Fried, G. Moses, C. D. Smith, and R. M. Satava, “Virtual reality simulation for the operating room,” *Annals of Surgery*, vol. 241, pp. 364–372, Feb. 2005. doi:10.1097/01.sla.0000151982.85062.80, PMID: 15650649, Last Accessed (1/2/2020).

[14] M. Cha, S. Han, J. Lee, and B. Choi, “A virtual reality based fire training simulator integrated with fire dynamics data,” *Fire Safety Journal*, vol. 50, pp. 12–24, May 2012. https://doi.org/10.1016/j.firesaf.2012.01.004, Last Accessed (1/2/2020).

[15] M. Nabiyouni, A. Saktheeswaran, D. A. Bowman, and A. Karanth, “Comparing the performance of natural, semi-natural, and non-natural locomotion techniques in virtual reality,” in *2015 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 3–10, March 2015. doi:10.1109/3DUI.2015.7131717, Last Accessed (1/2/2020).

[16] M. M. Davis, J. L. Gabbard, D. A. Bowman, and D. Gracanin, “Depth-based 3d gesture multi-level radial menu for virtual object manipulation,” in *2016 IEEE Virtual Reality (VR)*, pp. 169–170, March 2016. doi:10.1109/VR.2016.7504707, Last Accessed (1/2/2020).