Material Based Fire Propagation and Degradation in Real-Time Video Games

A Thesis Submitted to the Faculty of the Interactive Design and Game Development
in Partial Fulfillment of the Requirements for the
Degree of Master of Fine Arts in Interactive Design and Game Development
at
Savannah College of Art and Design

Sanandanan Somasekaran

Savannah

© August 2015

The author hereby grants SCAD permission to reproduce and to distribute publicly paper and electronic thesis copies of document in whole or in part in any medium now known or hereafter created.

Aram N. Cookson, Committee Chair

SuAnne Fu, Committee Member

Charles Shami, Committee Member

ACKNOWLEDGEMENTS

Thesis Chair

Aram N. Cookson

Thesis Committee

SuAnne Fu

Charles Shami

Technical Advisory

Ben Esler

FX Advisory

Raed Alamoudi

Special Thanks

Keith Guerrette

Will Fagan

Everett Gunther

Esther Nho

Brandon Swan

Josh Parker

Matthew Currey

TABLE OF CONTENTS

LIST OF TABLES 1
LIST OF FIGURES 1
ABSTRACT 3
1 INTRODUCTION 4
2 KEY CONCEPTS 16
2.1 Components of Fire 16
2.2 Pyrolysis 17
2.3 Assumptions 19
3 IMPLEMENTATION 20
3.1 Vertex Positioning vs. Cellular Automata 23
3.2 Spline based Propagation - Sequential Growth 23
3.3 Intelligent System 24
3.4 Propagation System 25
3.5 Degradation System 28
Material Transitioning 28
Observations 30
Material Degradation Results 31
3.6 Fire system 40
Construction of Flames 42
Observations (Real World) 49
Observations (Thesis Demo) 50
Particle System Results 52
4 POSTMORTEM 57
4.1 Alternative to Vertex Based Propagation 57
4.2 Propagation on Skeletal Meshes 58
4.3 Structural Damages to Burning Meshes 58
4.4 Orientation Based Propagation 58
4.5 Layered Propagation & Degradation 59
5 CONCLUSION 59
Appendix A 61
Constructor 61

Essential Editable Presets 61
Advanced Editable Presets 62
Advanced Presets (Not Editable) 65
Appendix B 67
Appendix C 75
Material Transitioning 84
Procedural Transitioning 90
Works Cited 94

LIST OF TABLES

Table 1 Summary of the pros and cons of various fire propagation methods 12
Table 2 Summary of the pros and cons of various fire propagation methods 14
Table 3 Examples of ignition point temperatures for different common materials..... 17
Table 4 Comparison of the degradation results between real-world and thesis visual project..... 31
Table 5 Chart listing flame colors with corresponding temperatures 41
Table 6 Summary of outputs of the various types of flames in the real world based..... 49
Table 7 Summary of outputs of the various types of flames in the thesis visual project..... 51

LIST OF FIGURES

Figure 1 A scene from the video game Alone in the Dark.. 6
Figure 2 Another scene from Alone in the Dark. 6
Figure 3 An interactive fire object capable of spreading fire to other combustible objects. 7
Figure 4 Fire propagation in *Far Cry 3*..... 8
Figure 5 Barbarian setting the floor spilt with oil on fire..... 9
Figure 6 Burning armor clad dead body shows no sign of fire or heat damage. 10
Figure 7 Main character escaping a burning down environment..... 11
Figure 8 Taxonomy diagram comparing and categorizing current fire propagation methods..... 21
Figure 9 Environment for the visual demonstration..... 22
Figure 10 Setting up a basic dynamic spline using Blueprint in Unreal 4 game engine..... 24
Figure 11 Bird's eye view of how the fire propagation tool in the visual project for this thesis works 27
Figure 12 Burnt state of demo environment 29
Figure 13 Comparison between burned plastic object in real-world" and thesis system.. 33
Figure 14 The images compare the burned concrete walls..... 34
Figure 15 Burnt Cardboard comparison. 35
Figure 16 Burnt Rubber comparison..... 36
Figure 17 Burnt Wood comparison..... 37
Figure 18 Burnt Marble comparison..... 38
Figure 19 Comparison of degradation results of fire retardant fabric.. 39
Figure 20 Drip effects of burning plastic..... 52
Figure 21 Spark and ember effects when wood burns. 53
Figure 22 Heat wave comparison. 54
Figure 23 Comparison between real-world chemical based fire 55
Figure 24 Comparison between burning rubber in real world 56
A1 Actor (mesh) with basic parameters 61
A2 Create ray-tracing box to check for actors (meshes) near burning mesh 63
A3 Store nearby meshes in an array 63
A4 Ray-trace boxes wrapping meshes 64
A5 Place a bounding sphere with the mouse click location as its center of origin..... 65
B1 Visual output for the growing sphere 68

B2 Mesh vertex locations and collision box 69
B3 Visual feedback for vertices of a burning mesh 70
B4 Check if growing bounding sphere has covered the corners of a mesh's bounding box 70
B5 Increase Radius of the bounding sphere 70
B6 Add a flame emitter to a vertex 72
B7 Removing flame particle emitters from vertices that have no fuel 73
B8 Updating vertex state and flame properties 73
B9 Checking for the closest vertex on the closest mesh 74
B10 Checking for a vertex on the closest mesh within ignitable distance and igniting that vertex 74
C1 Master Material Class 75
C2 Base Color material expressions 76
C3 Emissive mask material expressions for combustible surfaces 77
C4 Different values for the Heat parameter to control emissive color of heated objects 77
C5 Emissive mask material expressions for metallic surfaces 78
C6 Growth mask for emissive color 78
C7 Forward/Reverse mask for emissive color 79
C8 Normal maps material expressions 80
C9 Material expression for creating World Position Offset for displacement effects 81
C10 Expressions to limit displacement along vertical (Z-axis) direction only. 81
C11 Material expressions to accept input for roughness maps 82
C12 Material Expressions for Opacity Masking 83
C13 Cardboard box (paper based) burning. 83
C14 Location Based Opacity Mask 85
C15 Pre-burnt to mid-burnt to post-burnt states 85
C16 (Left) without cloud edge turbulence with cloud edge turbulence 86
C17 Edge Glow texture sample 86
C18 Material Network showing creation of glowing edges as objects burn. 87
C19 Creating the edge glow. 87
C20 Burnt state of demo environment 88
C21 Setting up the cooling sphere in blueprint 89
C22 Incrementing the cooling sphere 89
C23 Conditional statement for ``cooldown'' 90
C24 Procedural Controls 91
C25 Effect on material when base color value is set to 0.97. 91
C26 Effect on material when opacity transition value is set to 0.75 92
C27 Integration of Master Material Class with Procedural Controls and Blueprint 92
C28 Color and Opacity Transition controlled procedurally 93
C29 Complete burning and charring of carpet material. 93

Abstract

Material Based Fire Propagation and Degradation in Real-Time Video Games

Sanandanan Somasekaran

August 2015

This thesis focuses on how to improve current fire propagation techniques. Current fire propagation systems in real-time video games lack material based interaction as fire spreads. These systems rely on scripted events and predefined paths that are not only time consuming to implement but are also not robust enough to adapt to design changes within a game level. These scripted events can also cause burning environments to appear very homogenous. The appearance and effects of fire on environment and textures seen in games do not capture dynamic surface degradation of the surrounding environment. This results in limited degradation effects of different materials in video games. This paper explores a fire propagation and material degradation system that will allow game developers to improve real-time fire propagation techniques and aesthetic fidelity of material degradations in video games. It explains and builds upon the core concepts of how fire interacts with different surfaces, introducing the term Material Based Propagation and Degradation.

Keywords: Fire Propagation, Material Degradation, Real-Time, Video Game Fire, Fire Damage

1 INTRODUCTION

This thesis explores a fire propagation methodology to create variations of material degradation as fire spreads and comes in contact with different surfaces. The process involved developing a tool in the Unreal Engine 4 to build an adaptive intelligent fire propagation system. The tool includes procedural controls that allow the game designer to control various thermodynamic properties of the scene objects in a level. With an understanding of basic thermodynamic properties of flames and flammable objects, designing procedural techniques and tools will improve real-time simulation of fire in game environments by allowing game artists and designers to rapidly deploy aesthetically appealing flames with greater control.

Examples of games that have used fire propagation to create burnt environments are *Alone in the Dark,*[1] *Far Cry 3,*[2] *Unchartered 3*[3] and *Ryse: Son of Rome*.[4] Based on my observation, these games implement a universal ``fireplace" technique where depending on the environment, the size of the fire is either scaled up or down. The details or subtle nuances involved in an inferno are not adequate with these techniques. For example, fire on burning wood would produce lots of crackling sound, embers and sparks. These effects would not work on a plastic object or rubber object as they burn differently. As such, most fires and the damage effects not only look homogeneous, creating art fatigue for the viewer, but also break the believability of a burning or burnt down environment.

[1] In Episode 1 Scene 5 and Episode 2 Scene 3
[2] Fire is used as a weapon against enemies and also to set the wild brush of Rook Islands aflame
[3] In the video game Chapter 7: Stay in the Light
[4] In the video game Chapter 1: The Beginning

In *Alone in the Dark*, the player is able to set any object in the environment on fire and the fire would spread across the surface of the object and burn it down (see Figure 1, Figure 2 and Figure 3).[5] As the fire spreads across the surface, the object's textures change from unburnt to burnt state. This texture change, however, is not a smooth transition but instant, A to B. Furthermore, the textures appearing on burnt objects in the game are highly tiled and repetitive. The ignitable objects found in *Alone in the Dark* game are predominantly wood. As these objects burned, they would break and fall apart. However, not all objects are made of the same type of wood or have the same wooden surface finish. Wooden objects can be made of softwood, hardwood, lacquer finished or varnished, plastic coated, oak, teak, and plywood etc. Each of these woods have different chemical composition which would burn at different rates and appearances. However, in the game, all the flames have the same visual similarity with no differences. They had similar heights, colors, brightness and turbulence. At the time of the development of this game, having a generic looking flame model and propagation sequence that fits in most environments was efficient as far as visual effects production pipeline was concerned. However, the flipside to this was should there be a change request to the flame model, the entire asset and related scripts would need to be modified or replaced.

[5] *Alone in the Dark - Fire*. Digital image.

Figure 1 A scene from the video game *Alone in the Dark*.[6] Objects set on fire by a flamethrower controlled by the player.

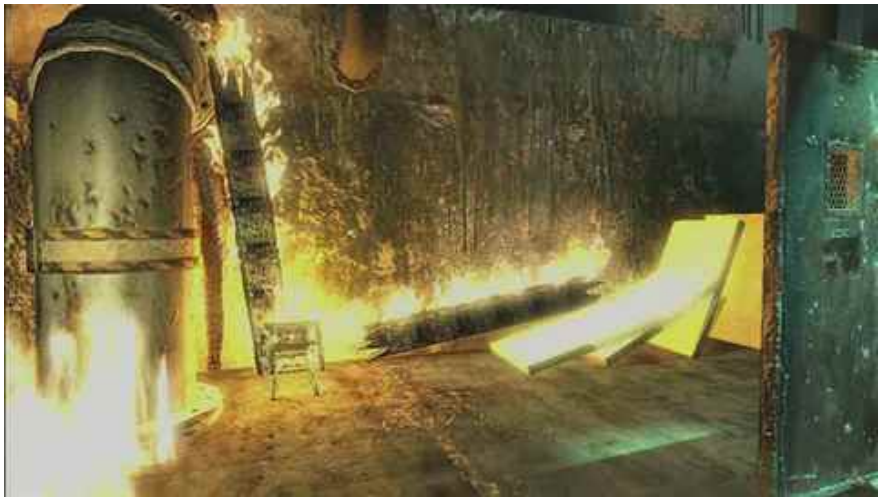


Figure 2 Another scene from *Alone in the Dark*.[7] Burning objects propagate fire to nearing objects and their surface degrades as they are burnt. Observe the tiled textures on burned objects.

[6] *Alone in the Dark - Fire*. Digital image.
[7] Ibid

Figure 3 From the video game *Alone in the Dark*, an interactive fire object, e.g. the chair, is capable of spreading fire to other combustible objects.[8]

Far Cry 3 implemented a very dynamic fire propagation system. Jean-Francois Levesque, the architect and programmer behind the fire propagation system, used an algorithm called Cellular Automata to spread fire across objects and environments. One of the many applications of Cellular Automata is simulating forest fires in the real world to study how wild fire spreads across vast landscapes. In Cellular Automata, a space is broken into a grid of cells. Each cell has a finite number of states (0 or 1). The state transition is governed by a rule or mathematical function that runs every clock cycle. Thus, the state of every cell is updated at the same time. A cell's state value of zero would mean it is not on fire and a value of one would mean it is on fire. A burning cell will affect its immediate adjacent cells. This technique is also explained in Chapter 3 to see how it has been applied in fire simulation. The propagation system in the game was hailed very realistic and held gamers in awe at the fire effects. The players were able to wield a flamethrower gun or explosive charges that can set aflame an environment (See Figure 4). The system also took into consideration also how environmental factors such as wind will affect the

[8] Burning object can interact with environment. Digital image.

direction in which the fire will spread. Again, as with *Alone in the Dark*, one of the main observations was the similarities in the flames burning on organic surfaces like bushes and plantations and flames burning on inorganic surfaces like houses.

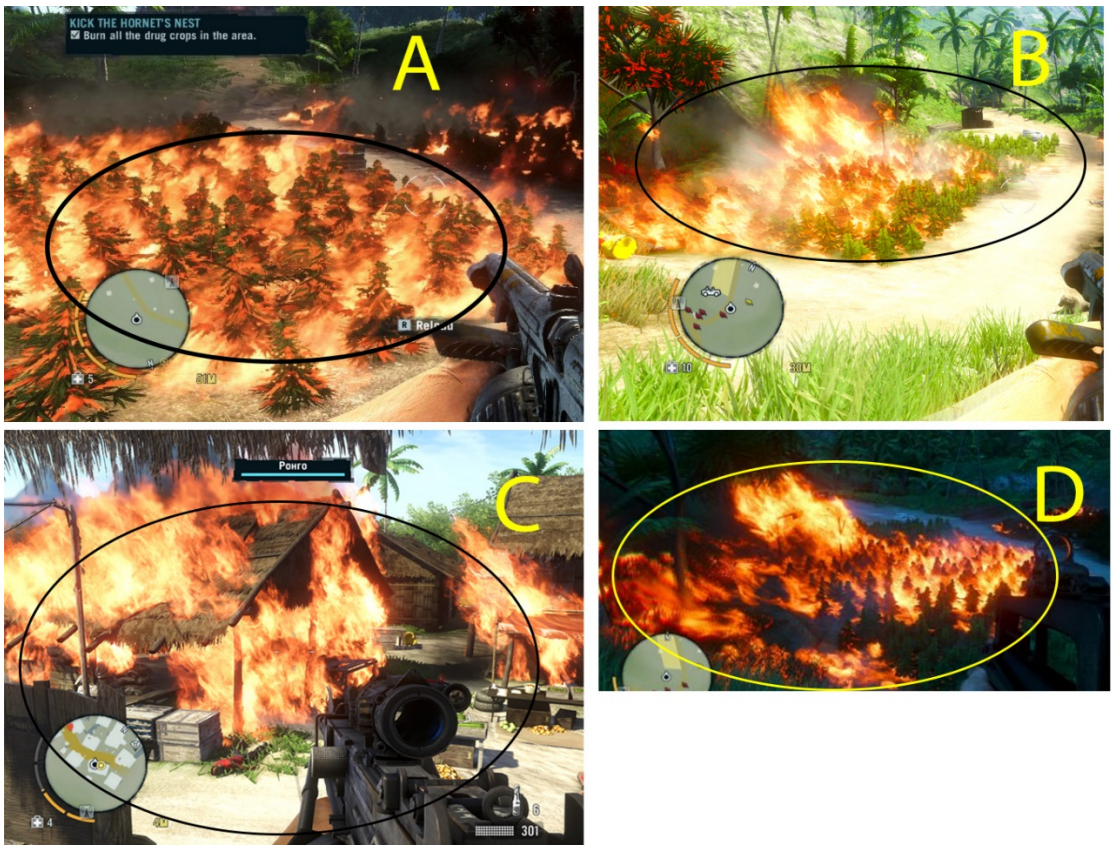


Figure 4 **A**: Grid like fire propagation.[9] All flames have uniform height with minimal variation. **B** and **D**: Minimum visual variation in the height of the flames burning the crops. [10,11] **C**: Fire burning on different areas of the hut appear the same in height, color, shape and turbulence.[12]

In the first chapter of *Ryse: Son of Rome*, the player will battle a barbarian warlord. In a brief cinematic sequence preceding the battle, the barbarian will use a burning torch to light up

[9] Plantation on fire. Digital image. *Far Cry 3* Walkthrough [GUIDE].
[10] Crops burning in *Far Cry 3*. Digital image.
[11] Levesque, JeanFrancois. "Far Cry: How the Fire Burns and Spreads."
[12] *Far Cry 3* Screenshots for Windows - MobyGames. Digital image.

the marble floor spilt with flammable oil (See Figure 5). The patch of oil catches fire surrounding the player and the enemy in a ring fire.

Figure 5 Barbarian setting the floor spilt with oil on fire. The assumption made here is that a scripted fixed path propagation with splines has been used.[13]

As the barbarian swings his torch around to hit the player, drips of burning oil hits the ground and a nearby dead soldier who is armor clad. The drips continue to burn even after hitting the ground. It is being assumed that the propagation of the fire is scripted with a possibility of the use of Splines. Splines are curves created along points plotted in three-dimensional space. They can either be auto generated real-time or pre-defined by a video game level designer. Splines are explained further in Chapter 3.2 Spline based Propagation - Sequential Growth. What is captivating is the realism and quality of the flames, the burning oil drips, and how these drips continue to burn even after they hit the ground. However, there two issues that needs to be pointed out. First, using manual scripting and splines are useful for controlled events such

[13] *Ryse Son of Rome Gameplay Walkthrough Part 1 - The Beginning (XBOX ONE)*. *YouTube*.

cinematic sequences but not necessarily in a larger player interactive environment. Secondly, it was observed that despite the intensity and scale of the flames, none of the objects including the dead soldier was damaged by the effects of heat and flames. There was no material interaction and degradation with heat and fire.

Figure 6 Burning armor clad dead body shows no sign of fire or heat damage.[14]

In Chapter 7 of the video game *Uncharted 3: Drake's Deception*, players will find themselves trying to escape a burning chateau, which was set aflame by enemies (see Figure 7). In a conversation with Keith Guerrette, former visual effects artist at Naughty Dog, company which made the video game, propagation of fire in the level was heavily hand scripted. The environment was primarily wooden. The appearance of fire and how it spreads were triggered by events such as players entering invisible trigger boxes or completing a certain tasks. The level was also interspersed with few cinematic sequences that had scripted fire events. The wooden environment had intense looking material effects and degradation such as glowing embers and structural damages.

[14] *Ryse Son of Rome Gameplay Walkthrough Part 1 - The Beginning (XBOX ONE)*. *YouTube*.

Figure 7 Main character escaping a burning down environment. Enemies set the place on fire and the main character has to escape the imminent danger.[15]

The following tables discusses the above games from production standpoint and player experience and compares them against the system developed for this thesis. These pros and cons are based on observing the gameplay mechanics, in-game effects, player interactions and the various assumptions drawn on how the fire propagation worked in the games.

[15] *Uncharted 3: Drake's Deception - Chateau Pt 2: Burning Up Gameplay Movie (PS3). YouTube*.

	Production	
	Pros	Cons
Far Cry 3	• Efficient due to fire being represented with one constant effect, spawning a few randomly sized particles. • Hair transplant methodology used[16]: Particle emitters are removed from areas not within camera view and placed into areas within player view. Saves allocated memory space. • Interactive o Fire propagates from point of contact with ignition source (player input).	• Requires designers to manually change parameters to achieve desired fire effects.
Ryse: Son of Rome	• Fixed path based propagation setup is quick to implement (It is assumed that Splines have been used for fixed path propagation). • Great for cinematic sequences as they may not need real-time interaction. o Since the art director has complete control of how the fire looks and spreads. • Burning oil drips collides with ground and continues to burn on the ground adding visual appeal.	• Fixed path based propagation methods can be limiting because if there is a need to scale the environment up or down, the propagation path may also need to be altered. This makes it not robust to design changes. • Each fixed path has to be manually placed causing longer development time.
Alone in the Dark	• Dynamic propagation • Interactive o Fire propagates from point of contact with ignition source (player input). o Makes the system capable of dynamic propagation o The game's physics engine alters the physical properties of a burning asset. • Single effect looking flames makes development easier and faster.	• Lacks material based fire propagation and degradation. • Lacks material based flame variations.

[16] Levesque, JeanFrancois. "Far Cry: How the Fire Burns and Spreads."

Uncharted 3	• Manual tweaking and placement of flames and trigger driven events allows greater level "hand-crafting" touch to a pyro level. • Ensures high level of gameplay and framerate optimization • Emitters not within immediate surrounding of player interaction and view are not rendered.	• Very time consuming to implement. • Requires a lot of manual placement and timed event triggers to spawn fire emitters.
Thesis System	• Can be used in cinematic sequences • Minimum scripting o Interactive objects that propagate fire based on underlying material. o Saves time on designing levels requiring pyro sequences. • Interactive o Fire propagates from point of contact with ignition source. o Makes the system capable of dynamic propagation • Procedural controls allow designers to set various thermodynamic values to create desired effects in a game level. • Once optimized, the system can support larger environments.	• Manual construction of propagation points (Vertices) • Density of flames is partly dependent on number of edge-loops and vertices on a mesh. • Single source of propagation • System is not currently optimized for framerate efficiency, memory management and particle count. o Needs more optimization to be game ready. • Current system has limited procedural controls but can be expanded for more comprehensive support. • Currently suitable only for small environments e.g. bedrooms. • Designer or artist has to manually create each type of material/flame which adds to production time.

Table 1 Summary of the pros and cons of various fire propagation methods used in different games in terms of production standpoint

	Player Experience	
	Pros	Cons
Far Cry 3	• Hair transplant methodology produces high framerate experience. • Fire is used as an important gameplay mechanic o Used as weapon (offensive and defensive)	• Minimal Visual Variance o Limited or no change in details over time • As fire is represented with one constant effect, the pattern and visual behavior becomes predictable after the first few iterations and becomes repetitive.
Ryse: Son of Rome	• Wow factor in cinematic sequences • Flame models and pyro sequence	• Based on observation, current propagation method would break dynamic
Alone in the Dark	• Interactive o Fire propagates from point of contact with ignition source. • Destructible environment (due to fire) • Fire is a core gameplay mechanic	• Art fatigue o As fire is represented with one constant effect, the pattern and visual behavior becomes predictable after the first few iterations and becomes repetitive.
Uncharted 3	• Meticulous placement of particle effects creates edge of the seat experience in pyro sequences. • Timed fire effects ensures surprise elements and excitement. • Optimized framerate and gameplay	• Game scene appears to be predictable for the fire propagation on replay of the scene.
Thesis System	• Improved Aesthetic Appeal • Diverse flame visuals keeps the player engaged and immersed • Opens opportunity for new gameplay mechanics or puzzles involving different materials burning or different rates of burning.	• Frame rates may drop when too many objects are burning concurrently.

Table 2 Summary of the pros and cons of various fire propagation methods used in different games in terms of player experience standpoint

Based on the observations made from these games, three ideas formed to address 1) how games can include different fire damage effects on environment based on different material constituents, 2) how fire should look, burn and spread as it interacts with different materials and finally 3) can a solution be developed to do away with manual scripting allowing fire to spread ``intelligently''. These ideas culminated to the following thesis statement.

This thesis argues that *adding material based propagation & degradation to current real-time fire propagation techniques will break the homogenous appearance of flames by adding dynamic visual variances and improving aesthetic believability of the effects of fire in video game environments.* It comprises of a visual project that demonstrates the concepts discussed in this thesis paper. The main goal of this visual project is to develop a tool that will greatly assist video game artists to develop environments subjected to fire. Further, designers and programmers will also stand to gain an understanding of material degradation that will allow them to create material based fire propagation techniques. While the core concepts of this project are demonstrated using the Unreal 4 game engine,[17] they are technology independent to allow game developers to adapt the methodologies explained herein to any game engine or rendering technique. The *Unreal 4* game engine uses a node-based visual scripting system called Blueprints to create interactive games. It allows designers to rapidly prototype and implement any gameplay elements. The construction of the visual component of this thesis is discussed in detail in this paper.

[17] *Unreal Engine 4*. Computer software. *What Is Unreal Engine 4*. Vers. 4.8. Epic Games, 2014. Web. Fall 2014.

All objects react to fire and heat differently, be it organic or inorganic. The core idea behind material based propagation & degradation, hereinafter referred to as MBPD, is to capture the propagation of fire and the degradation effects of an object or its surface details based on the underlying material when exposed to heat or open flame. The concepts are built upon how materials interact with fire in the real world.

In the next chapter, the concepts of how fire breaks down an object and why it is important to understand the stages involved in the decomposition will be explained.

2 KEY CONCEPTS

This chapter discusses important concepts required to understand how fire spreads and affects its environment. There are three basic elements required for fire to exist: 1) fuel 2) heat 3) oxygen. All three need to be present with the right mix for fire to exist. Even if one of these elements is absent, fire will cease to exist.

2.1 Components of Fire

Fuel

Material that stores chemical energy needed to be supplied to produce heat energy.

Oxygen

Second reactant in the chemical reaction, the amount of oxygen supplied determines quality of combustion.

Heat

Required energy to be supplied into the fuel to raise sufficient temperature for chemical reaction to begin. Fire cannot start without sufficient heat. An object has to be heated to a certain temperature before it catches fire. This temperature is called the ignition temperature. Ignition Temperature ``is the temperature at which the rate of heating in the substance being tested exceeds the rate of heating induced by the external source of heat and has visible combustion in the form of a glow or flame as an end result.''[18] Here are few examples of ignition point temperatures for different common materials[19]:

Material	**Ignition Temperature(°Celsius)**
Paper	218-246
Wood-based products	330-375
Combustible Fabric (Cotton)	250
Fire Retardant Fabric	>400
Rubber	260-316
Non-Fire Retardant Plastics	270-360
Metal (Steel)	1100-1600

Table 3 Examples of ignition point temperatures for different common materials

The next section describes what happens to materials when they are exposed to heat and what happens after it is applied.

2.2 Pyrolysis

When an object is subjected to heat or fire, its surface is exposed to chemical decomposition which occurs in a sequential order, from solid to liquid to gas. This is known as

[18] Graf, S.H. "Ignition Temperatures of Various Papers, Woods and Fabrics." March 1949, Bulletin No. ed.: 7.

[19] Cafe, Tony. "Physical Constant for Investigators."

pyrolysis. Pyrolysis is "the chemical decomposition of a material into one or more other substances due to heat alone. All solid combustibles must undergo pyrolysis in order to generate gaseous fuel vapors for flaming combustion."[20] It is very important that game developers understand the process of pyrolysis when creating burnt environments as look development needs to capture the details of various stages of pyrolysis to create a sense of believability in an environment exposed to fire.

There are three stages in pyrolysis: solid, liquid and gas. Flames are seen when an object has decomposed to an extremely hot gaseous state. During this process, objects exhibit three types of deformation.[21] They are 1) Intumescence, 2) Charring, and 3) Melting.

Intumescence is the bubbling and warping of surfaces as seen on painted/tinted plastic objects. Charring is the blackening of surfaces due to buildup of carbon. Finally, melting is the transition objects such as plastics undergo prior to becoming gas. In each of these states, the textures and materials of degrading objects appear very differently. It is crucial to capture these effects. The relevance of the pyrolysis concept will be elaborated further in Chapter 3.9 when this paper describes how the system developed for this thesis applies material transitions to a surface.

The fields of fluid dynamics and thermodynamics research have high accuracy computations to quantify each of the elements and compute the chemical reactions. However, such level of computations can tax the game engine immensely thereby compromising

[20] Stroup, David W., and Daniel Madrzykowski. "Section 2: Flammability Hazard of Materials." *Fire Protection Handbook, Volume 1*. N.p.: National Fire Protection Associates, n.d. 2-32. Web. Spring 2014.
[21] Ibid 2-33

performance. To simplify the process of combustion, a few assumptions were made without compromising believability. These assumption are discussed in the next section.

2.3 Assumptions

Firstly, for the purpose of this thesis demonstration, presence of ambient air (or oxygen) supply was assumed to be in infinite supply. Therefore, oxygen was not included in any calculation. Secondly, types of fuels were limited to common, everyday materials encountered in a living room as a player can most easily relate to them. Examples include the following:

1) Fire-retardant Fabric
2) Combustible Fabric
3) Wood
4) Metal
5) Plastic
6) Paper
7) Hard paper
8) Rubber
9) Concrete
10) Ceramic/Marble

The following section describes in detail the actual development of the visual system to demonstrate this thesis argument.

3 IMPLEMENTATION

The visual portion of this thesis demonstrates the concept of material based propagation. Using the *Unreal 4* game engine, a fire system tool was built with a goal to relieve the designer of having to manually script a propagation path. The tool will allow the designer to drag and drop a script into the game level, and the script will run independently. The script then allows the designer to assign a mesh and a material to the mesh. Based on its thermodynamic property, the mesh will react to fire and spread the flames to surrounding meshes who inherit the same script behavior. The system simulates a material based propagation of fire across the environment based on the user input. Simply put, material based propagation of fire is the manner in which fire will spread through a game level based on its interaction with its surroundings. The purpose is to create variations in the flames and the surfaces they come in contact with.

A choice had to be made between emulation and simulation. A taxonomy chart (see Figure 8) categorizes where current fire propagation methods fall in terms of emulation and simulation. In reference to fire propagation, an emulation system would involve the recreation of every aspect, particularly thermodynamic properties, of fire and its behavior accurately. On the other hand, simulation meant recreating the basic behavior of fire by simplifying various properties of thermodynamics. Complete physical accuracy seemed impractical due to the complexity of thermodynamics calculations. In real life, it may take hours for an environment to burn down as each object undergoes different chemical decompositions. Players cannot be expected to spend that amount of time in a video game. Thus, real-time constraints refer to player feedback within specified time constraints in games. It is essential to keep players in awe by having a fire propagation system to mimic real-world phenomenon while keeping the

entertainment value of the game. Accurate calculations of thermodynamics would increase computational intensity of a game system. In order to improve performance, it is important to simplify the design and stylize the artistic aspects of the propagation system. Thus, the visual project designed for this thesis simulates the real-world phenomenon of fire propagation.

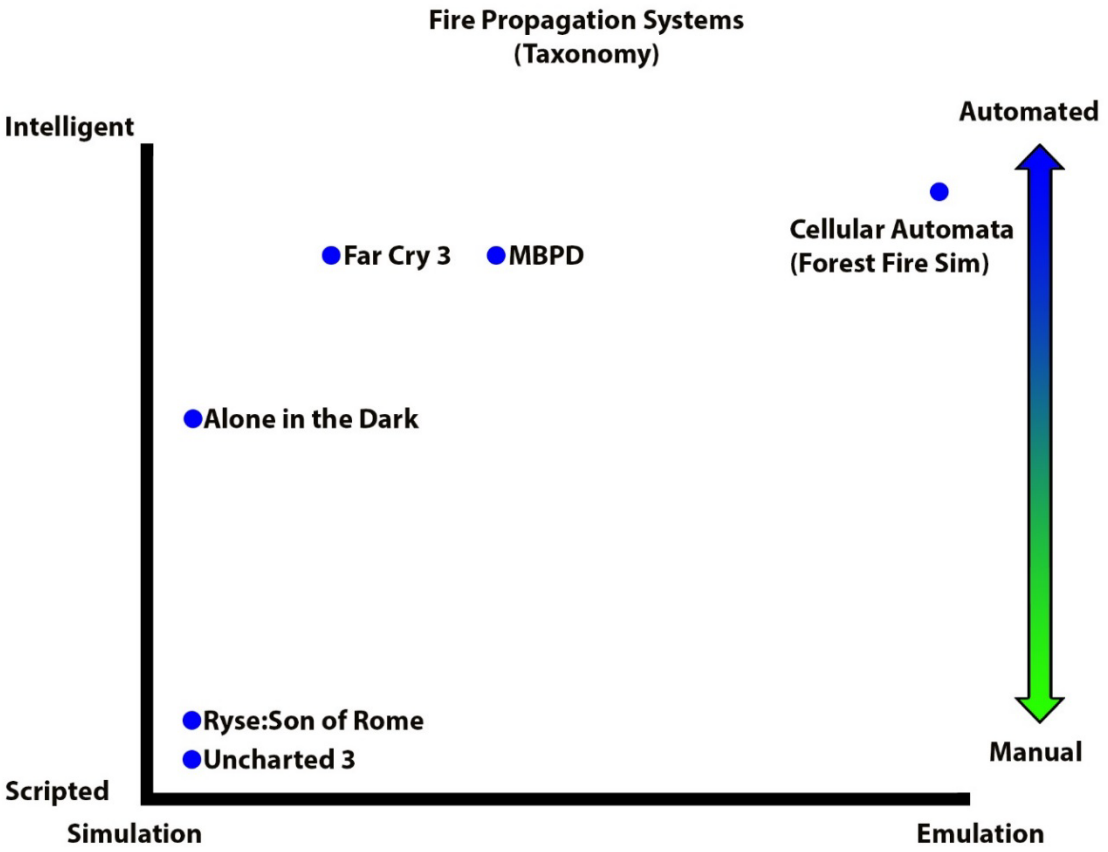


Figure 8 The above taxonomy diagram compares and categorizes where current fire propagation methods fall. The categorization is based on my observation and assumptions of the methods used in the mentioned games and applications. The horizontal axis ranges from simulation to emulation. An emulator recreate every aspect, particularly thermodynamic properties, of fire and its behavior accurately. Simulator recreates the basic behavior of fire by simplifying various properties of thermodynamics (i.e. make belief). The vertical axis depicts how pyro effects in games range from scripted (i.e. manual hard coding) methods to intelligent self-propagating methods.

Figure 9 Environment for the visual demonstration

The propagation technique being developed is based on Jean-Francois Levesque's *Far Cry 3* method[22] of spreading fire across an environment and objects. Levesque uses a cellular automata technique to spread fire. As discussed briefly in Chapter 1, Cellular Automata is used in the real-world to simulate and study forest fires. In this method, a grid is simulated across a surface with individual cells possessing a state attribute. This state attribute is of a finite range; in the case of Levesque's method, the values are either zero or one. Each cell's state is affected by its own state as well as those of adjacent cells. A common rule such as a mathematical or logic function is used to determine the state transition. While cells may each be in a different state, their transitions will happen at the same iteration. For more complicated meshes, an Axis-Aligned Bounding Box (AABB) is used to surround an object. If, for example, the mesh is a chair, the chair is given a bounding box or a collision box. The box is then split into smaller, evenly spaced cubes.

[22] Levesque, JeanFrancois. "Far Cry: How the Fire Burns and Spreads." N.p., 6 Dec. 2012. Web. Spring 2014.

Through an iterative algorithm, each cube is tested for collision against the mesh to determine whether a cube is empty or is currently overlapping with the parts of the chair. A cube that has a positive collision is a propagation point. Collectively, the cubes will represent the shape of the mesh and is now ready for propagation.

3.1 Vertex Positioning vs. Cellular Automata

In cellular automata, spawning positions for flame emitters are generated dynamically at runtime. In this thesis's visual demo, an alternate approach was implemented to create the spawning positions at pre-compile times to optimize performance. The spawning positions were vertices constructed when a scene object was modeled using a 3D modeling software. Depending on required flame density, the mesh would contain corresponding edge-loops to create vertices. The denser the amount of vertices, the denser the object's burning would appear.

The game engine of choice is Unreal 4 due to its comprehensive list of visual nodes that can be used to construct fully functional game mechanics. In order to access a mesh's vertex data, a plug-in called *Rama's Victory Plug-in*[23] was used. Using this plug-in, it was possible to get the vertex locations of an imported mesh.

3.2 Spline based Propagation - Sequential Growth

Before using *Rama's Victory Plug-in*,[24] the initial method for spreading fire was to use spline based propagation. A spline is a curve between two points. The spline method involves creating a path using the spline components in Unreal Blueprint. At construction, the designer

[23] Rama. "Rama's Vertex Snap Editor Plugin." *Epic Wiki*. Epic Games, 2014. Web. Fall 2014.
[24] The ``Victory Plug-in'' is a zip file containing custom blueprint nodes and C++ codes that allows developers to access vertex information of a mesh. It is written by a programmer called Rama who works with the Unreal 4 engine.

would create a path using splines. A spline has a minimum of two points, Start and End, with an option to create many more in between or after the last point. The points can be retrieved and their locations computed and used to spawn particle emitters.

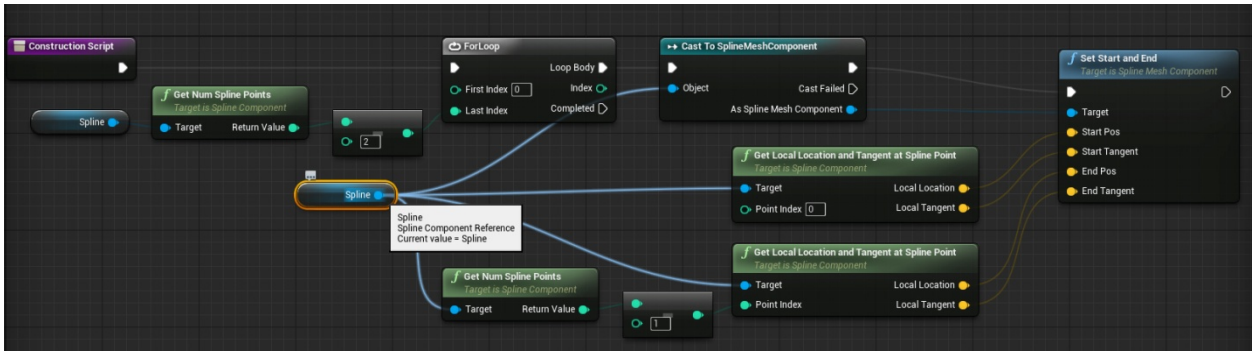


Figure 10 Setting up a basic dynamic spline using Blueprint in *Unreal 4 game* engine

The limitation with spline propagation was that the level designer would have to plan and manually create a spline path, which can create an issue of complexity and confusion if the number of spline paths increase. Furthermore, a spline propagation is very linear and sequential. That means when an event triggers fire, the fire particle emitters would spawn in the order the spline points were created. In addition, should there be a change in the level design, the designer would need to remap all the spline paths in the scene. Given these limitations, the spline based propagation was considered inappropriate for building a dynamic intelligent fire propagation system.

3.3 Intelligent System

The design approach to an intelligent fire system is Object-Oriented. In object-oriented design, the actor class would hold essential thermodynamic attributes. When dragged and dropped into a level, that actor is ready to interact with fire. Depending on the actor's preset parameters, fire will spread and react with the actor accordingly. The preset parameters enable

the initialization of the actor, which is basically an object or mesh that will interact with heat and fire. Appendix A details the construction of this actor's preset parameters.

3.4 Propagation System

After the initialization, an object is ready to be interacted with. Interaction is via a user input, e.g. left mouse click. The location of the mouse click is stored as the source point for fire to spread throughout the object. Following the mouse input, there are two operations are carried out: 1) two invisible expanding spheres, first one called bounding sphere and a second one called cooling sphere are spawned at the source point and 2) a ray trace box enveloping the object checks for overlapping ray trace boxes belonging to other objects of the same class. The use of the cooling sphere is explained in section 3.5 under Material Transitioning.

During the first operation, the expanding bounding sphere checks for vertices falling within its radius. Using a First-In-First-Out approach, the enveloped vertices are added to a data structure array list called "propagation list". Each element in this propagation list contains the location of a vertex, corresponding vertex temperature and fuel values, a particle emitter component, and two Boolean states to check if a vertex has been triggered to spawn a fire emitter and if the vertex has burnt out. Each element in the array list is also a propagation data as it tells the system where to spawn a fire emitter and what to do with it. The invisible sphere continues to grow until all the object's vertices have fallen within its radius and been added to the propagation list.

In the second operation, the selected object's ray trace box checks for any overlapping ray trace boxes other objects of the same class. Any overlap means that there is an object within range for fire to spread to.

At every frame interval, the system checks if there is an element added to the propagation list. If there is a new element, it spawns a particle system component (fire emitter) and adds it to corresponding data structure element in the propagation list. The vertex is also marked as triggered. Once added, at each frame interval, the vertex temperature value is incremented while the vertex fuel value is decremented. The vertex temperature contributes to the color of the flames while vertex the fuel value determines how long a flame will continue to burn at the vertex location. When the fuel value reaches zero, the vertex is marked as burnt and the particle system component is removed from the data structure to free allocated memory space.

A burning vertex constantly checks the distance between itself and the vertices belonging to the objects with overlapping ray trace boxes. Basically, it is checking for the closest vertex of the closest mesh. The closest vertex has to be within a certain distance for it to catch fire. If not, the mesh remains unscathed. If the closest vertex falls within the specified distance, that vertex is ignited and the whole process of propagation is repeated. See Figure 11 for a bird's eye view of how the propagation script works.

Appendix B explains the construction of the propagation script in the *Unreal Engine 4* and how this class works in detail. Next, the paper discusses the degradation process that occurs when an object is on fire.

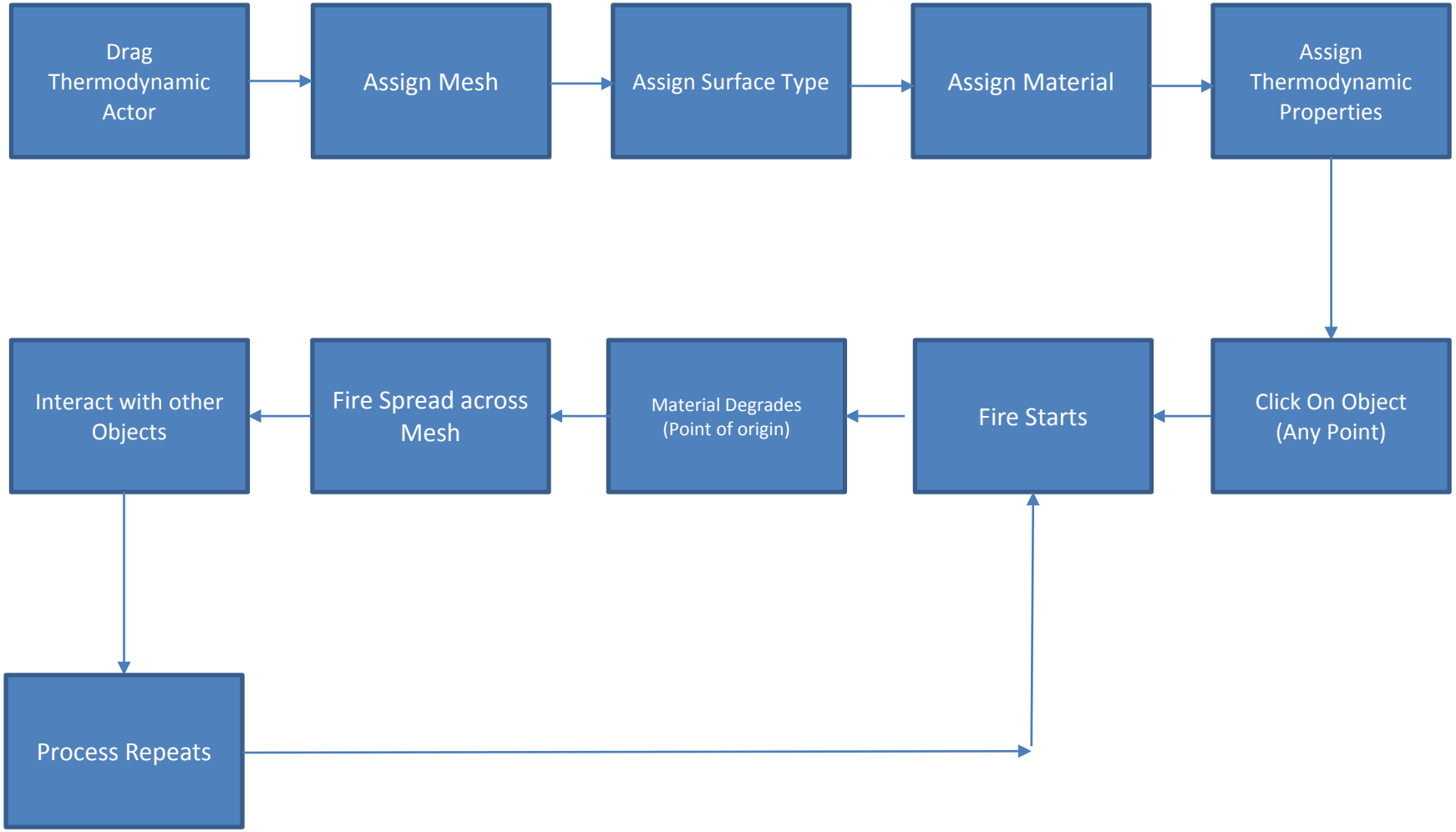


Figure 11 The above flowchart provides a bird's eye view of how the fire propagation tool in the visual project for this thesis works

3.5 Degradation System

Objects go through material degradation when they are exposed to heat or fire. Using real-time shaders, it was possible to achieve the desired degradations effects of a burning object. A Master Material Class was constructed from which a material instance will be created for each scene object. An instanced material allows procedural modification of an object's appearance. It enables manual control over the burning states of the objects. This also allows designers to test every parameter that affects the appearance of an object. The material instance is integrated with the dynamic propagation system so that the parameter controls are taken over by the system. The highlight of the degradation system is the way in which the material transitions from a pre-burnt state to a burnt state. Appendix C discusses in detail the construction of the material degradation system in the *Unreal Engine 4* and how they are integrated with the propagation script.

Material Transitioning

Material transition is a process in which texture A changes to texture B based on either a scalar parameter or another texture, which could be used as a pattern. The node or expression that drives the transition is the Linear Interpolation (Lerp).

Heating Up

When an object is setup and initialized, its materials and textures are also set. These include its unburnt state, burnt state and an optional mid-burnt state. The key to revealing the underlying texture is using an expanding mask, preferably a spherical mask.

This mask has a center and a growing radius that determines how a burning object's surface reveals its underlying texture. When the player clicks on an object, the point of click

becomes the center of the mask. The mask's radius is derived from the radius of the bounding sphere explained in section 3.4. The bounding sphere is used to check for overlapping vertices to spawn particles at each vertex location. At every frame, as the bounding sphere's radius is incremented, the sphere mask's radius is also updated. This masking technique reveals the burnt texture under an un-burnt texture. Enhancements to the mask would include adding turbulence to the edges or borders of the mask to give the transition more randomness.

Figure 12 Burnt state of demo environment

Cooling Down

The previous sub-section described the material transition involved when objects were burning. This sub-section describes what happens when the objects have finished burning and are cooling down. When a user clicks on a mesh, both a bounding sphere and a cooling sphere are spawned at the mouse click location. That location becomes the center of the spheres' radius. When a mesh has more than 90%-95% of its vertices completely burnt, the cooling sphere begins

to expand. As the cooling sphere expands, the areas falling within its radius begin to cool down. Visually, this is represented by the reduction in the amount of emissive color given out by that region previously burning. During the propagation stage, the material changes to a burnt state and emits red and red-orange glows. During the cooling phase, the emissive colors transition to zero.

Observations

The following table compares the degradation results of this thesis's visual component with real world degradation of corresponding materials. It is followed by the visual observation and comparison of the results.

Type	Real World	Thesis Result
Plastic	1. Melting and decomposing material break into chunks 2. Chunks drip while still burning in the air 3. Burning object deforms and warps	1. Melting and sputtering chunks 2. Chunks drip while still burning in the air 3. Burning object deforms mildly and warps in some areas
Paper	1. Areas exposed to heat turn brown 2. Areas on flame disappear as fuel burns	1. Areas exposed to heat turn brown 2. Burnt areas become transparent 3. However, the flames do not align correctly with the disappearing edges. As a result, even some areas spawn flames after they become transparent.
Rubber	1. Melts under extreme heat and disintegrates. This process in real world. 2. Disintegrating areas break away	1. Warps and deforms 2. Heavy charring
Wood	1. Heavy charring 2. Glowing Embers 3. Breaks down and crumbles to smaller chunks 4. Loss of structural integrity	1. Burning process produces lots of crackles and sparks 2. Surface chars and turns ash grey in many areas 3. Hot ember glows after fire dies
Fire Retardant Fabric	1. Very slow charring and degradation 2. Burning areas and receding fabric tips glow bright. 3. Depending on chemical composition, varying degree of decomposition occurs. 4. Shrinks from heat 5. Black colored beads that cannot be crushed	1. Burns very hot but slowly 2. Edges burn away slowly 3. Some surface decomposition occurs
Combustible Fabric	1. Material burns completely till its reduced to crumbles and ashes 2. Material degradation is very fast 3. Small areas of glowing embers 4. Fine gray ash residues	1. Burns very quickly but not as hot as other materials 2. Object does not decompose completely. 3. Displacement maps cause object to have some deformation

Concrete	1. Turns brown or chars at extreme temperatures 2. Structure integrity is compromised	1. Turns brown or chars at extreme temperatures
Ceramic/Marble	1. Some ceramics or marbles crack if heated to extremely high temperatures and rapidly cooled down 2. Soot from other burning objects settles on the surface 3. May deform like any other metamorphic rocks subjected to extreme heat	1. Turns brown or chars at extreme temperatures to give the illusion of soot depositing on surface. Artistic liberty was taken in degrading marble surface.
Metal	1. Glows red in high temperatures. 2. At extreme temperatures, different colors varying from blue to red form. 3. Extreme temperatures will also start melting the surface	1. Glows red at high temperatures.

Table 4 Comparison of the degradation results between real-world and thesis visual project.

Plastic

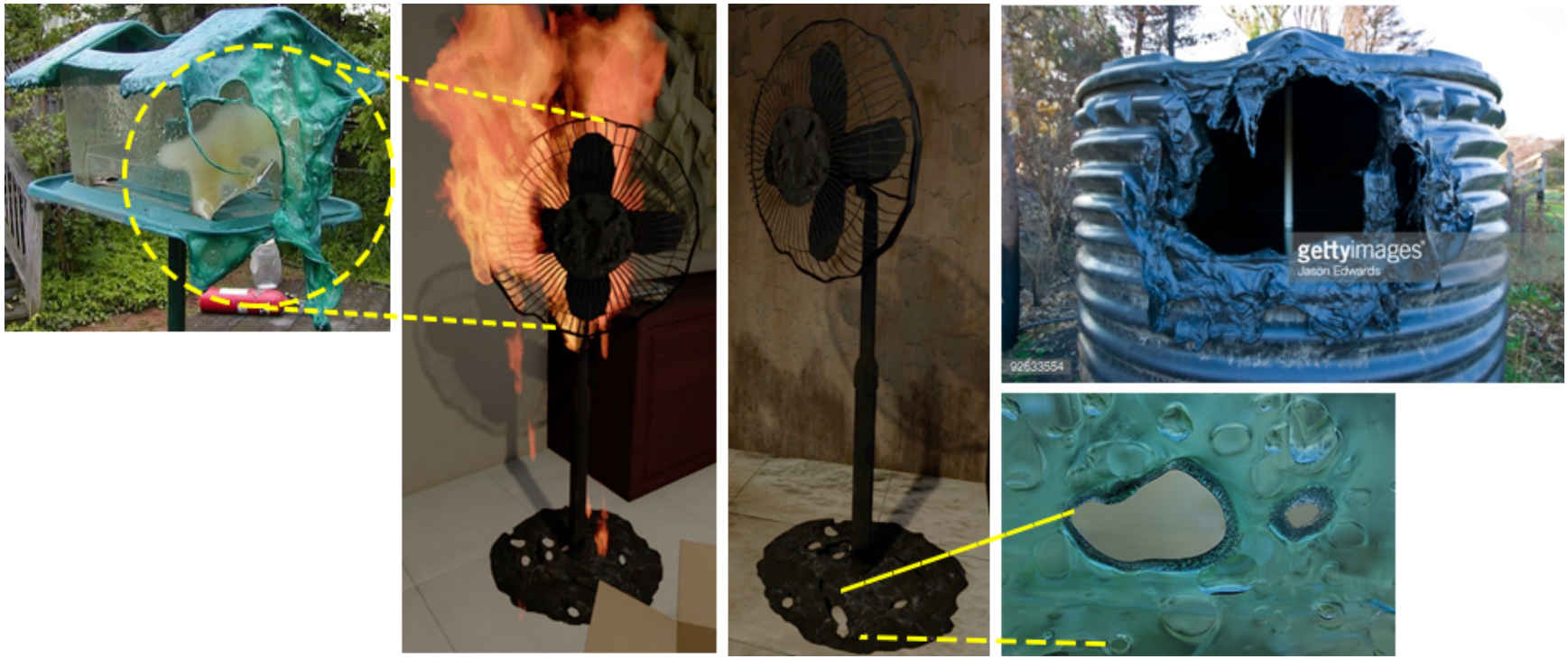


Figure 13 Comparison between burned plastic object in real-world[25,26,27] and thesis system (bottom). As explained in Section 2.2, plastics undergo a 3 stage pyrolysis: Solid -> Liquid -> Gas. When exposed to heat, plastic begins to melt. During this stage, intumescence, which is bubbling and warping of the surface can be observed on a plastic surface. The melting also causes severe deformation of a plastic object's shape. Using displacement maps (See under section 3.5 Degradation System), it was possible to achieve the effects of intumescence on burning plastic.

[25] Burned and melted Plastic. Digital image

[26] Rizzo, Francesca. *Burnt Plastic*. Digital image

[27] Edwards, Jason. *A Plastic Water Tank Melted in the Extreme Heat of a Forest Fire.* Digital image.

Concrete

Figure 14 The images compare the burned concrete walls in real-world (top and bottom left)[28,29,30] and thesis visual project (bottom right). The goal was to achieve a scorched effect on a concrete wall as can be seen in real life. The images on top are the result of arson.

[28] Datta, Arko. *In Pictures: Gujarat's 'houses of Death'* Digital image.
[29] *Jerusalem - Police Arrest Suspects In Mosque Arson Attack*. Digital image.
[30] *Urbex: Abandoned, Burned, Semi-Demolished Emge Foods Meat Processing Plant*. Digital image.

Paper

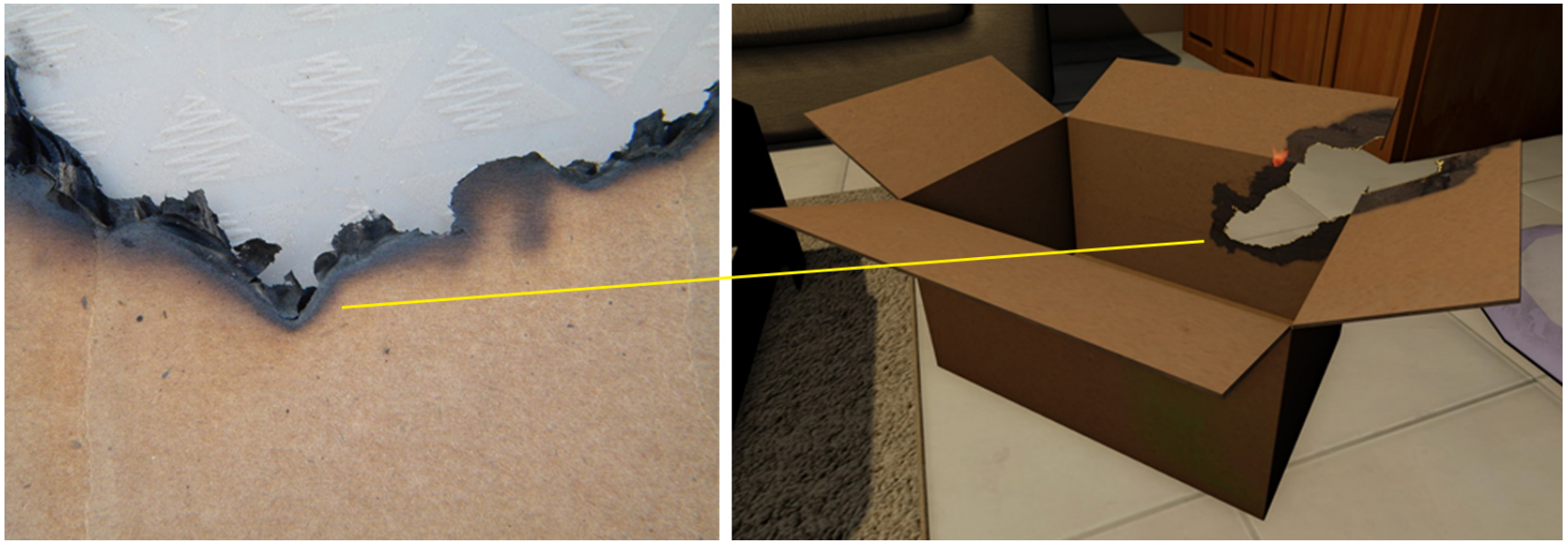


Figure 15 Burnt Cardboard comparison between real-world (left)[31] *and thesis system (right).* The Master Material Class allows designers to plug in textures for pre-burnt, mid-burnt, and burnt states for a burning object. In addition, for paper, the designer can switch on an optional opacity mask that controls which areas should be zero opacity. As paper burns, the areas close to the heat source first turn dark brown. This is achieved by plugging in a mid-burnt state texture. As the material transitioning sphere expands[32], the opacity mask also expands causing burnt areas to disappear.

[31] Smith, Andrew. *Burnt Cardboard*. Digital image.

[32] See Heating Up under Material Transitioning in Section 3.9

Rubber

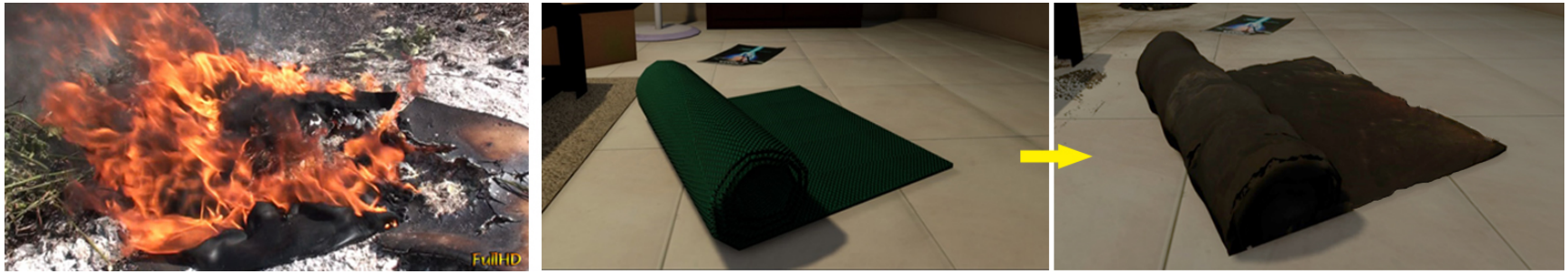


Figure 16 Burnt Rubber comparison between real-world (Left)[33] and thesis system (Right). Right shows pre-burnt and burnt states of a rubber yoga mat. The image on the left shows a rubber tube or tyre on fire. Areas not on fire but close to the flames exhibit surface deformation. In the thesis system simulation, the rubber mat warps and deforms. It melts as it undergoes the following pyrolysis stages: Solid -> Liquid -> Gas. Rubber melts under extreme temperatures and then catches fire.

[33] *Burning Rubber*. Digital image. *Envato Market*.

Wood

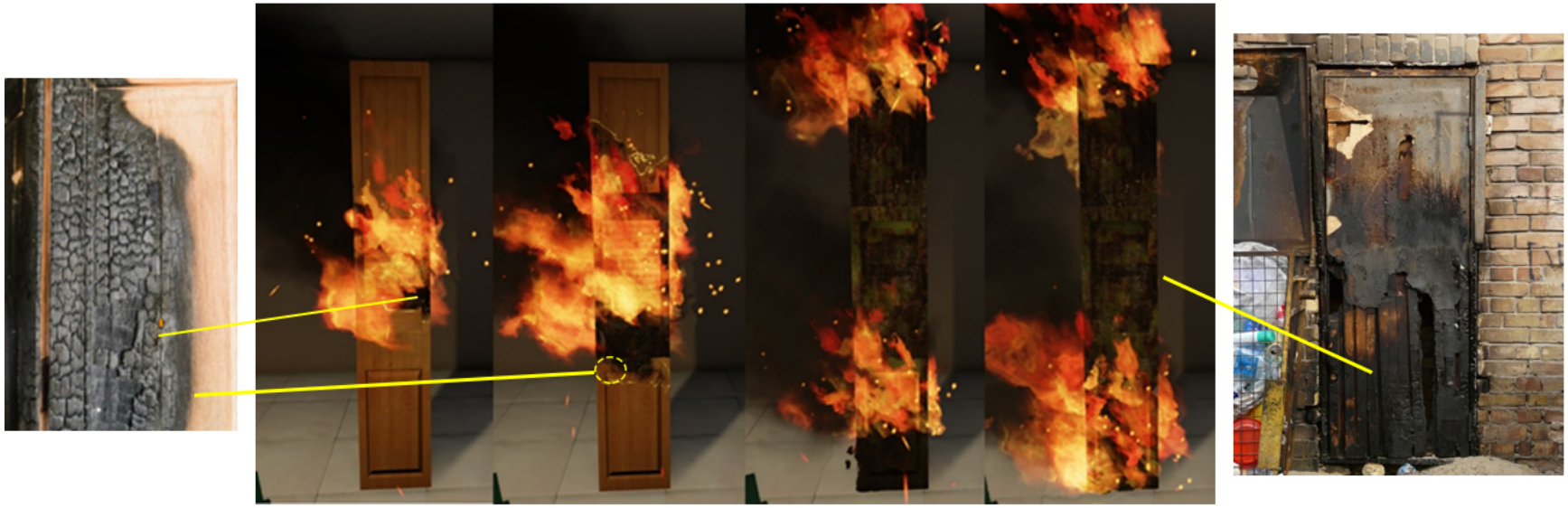


Figure 17 Burnt Wood comparison between real-world (sides)[34,35] and thesis results (center). The image in the center shows successive damage to a wooden wardrobe door. As wood is exposed to heat, the edges close to the heat or fire darkens before catching fire and then chars. To achieve this effect, an additional mid-burnt state texture was used.

[34] *Burn on Wooden Door Stock Photo 54838856 - IStock*. Digital image. *IStock by Getty Images*.
[35] *Free Photoshop Patterns and Textures of Wood and Metal*. Digital image. *SitePoint*.

Marble

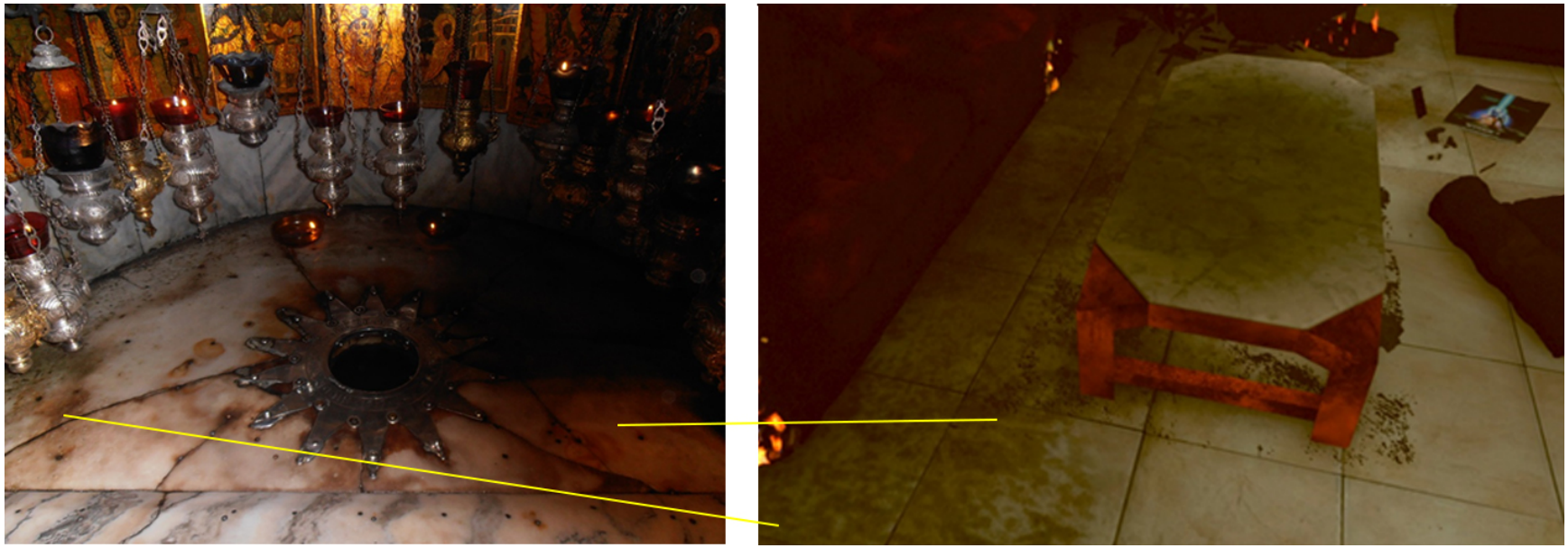


Figure 18 (Left) Shows oil and heat stained marble surface at an altar.[36] (Right) Marble/Ceramic tiling and tea-table top exposed to high temperatures. The effects shown in the visual component of this thesis has been exaggerated to give an elevated sense of damage to an environment.

[36] 14-pointed Silver Star set into the marble floor and surrounded by silver lamps. Digital image. *Pilgrimage in the Holy Land*.

Fire Retardant Fabric

Figure 19 Comparison of degradation results of fire retardant fabric. Example used is a couch. (Top) shows real-world burnt couch.[37] (Bottom) shows couch in thesis demo system. Although the top image is not cited to be a fire-retardant material, it has been used as a reference for deriving some artistic liberty in simulating a burning/burnt fire-retardant couch. Fire-retardant fabrics are made of polyester material. In this simulation it is assumed the couch is made of part polyester and part cotton.

[37] Channell, Justin. *Burnt Couch*. Digital image. *Flickr*. Yahoo!

3.6 Fire system

In this section, we examine in detail how the fire system was constructed. The fire system basically controls the different types of fire and their appearances. Each material type (e.g. plastic, paper etc.) in the thesis demo has a separate particle system to create visual variations. Construction of each of these particle systems was primarily based on visual references from footage and photographs of real world materials burning. The particle systems were built on top of existing fire templates available on the Unreal 4 marketplace. It is stressed here that while the simulation of these flames were based on basic concepts of fire and visual references from actual footage and photographs, it is primarily artistically driven. Liberties were taken to stylize the visual output for entertainment value. The stylization includes exaggeration of these visual outputs.

There were six factors considered as key elements in creating visually believable flames. These factors form the basis for constructing different types of flames. They are Effects, Smoke, Color, Turbulence and Size.

Smoke ``designates the by-products of combustion, including fire gases (carbon dioxide, water vapor, carbon monoxide, and other products of a combustion reaction) and particulates such as carbon and unburned particulates that are transported by convected currents.''[38]

Turbulence refers to how wild and chaotic flames are. ``When the flow of fuel is at a faster rate than the air can be mixed into the reaction, the mixing process occurs in whirls, which form

[38] Gorbett, Gregory E., and James L. Pharr. *Fire Dynamics*. Upper Saddle River, NJ: Pearson, 2011. Print. 78

turbulent diffusion flames. Turbulence occurs when the fuel gases released are less constant, which results in whirls in the zone where fuel and air mixtures support combustion.''[39]

Observable flame colors are determined by ``flame temperatures and the types of chemical radicals produced in the reaction. Changes in the percentage of oxygen available for combustion and fuel composition are also factors in determining flame color.[40] In the thesis system, oxygen is assumed to be in infinite supply and is therefore not factored in determining flame color. Thus, only temperature and fuel composition are considered. Fire retardants ``are intended to reduce either the ignitability of combustibility of a substance. Most often, these materials are chemical additives that are either added during the production of materials or applied after the material is in its final form.[41]'' Thus, when designing the flames for a fire retardant material, some artistic liberty was taken into deciding the colors, sizes and heights of the flames. The following table lists the different color temperatures of heat (in °C): [42]

Dull Red	500-600
Dark Red	600-800
Bright Red	800-1000
Yellow Red	1000-1200
Bright Yellow	1200-1400
White	1400-1600

Table 5 Chart listing flame colors with corresponding temperatures

Effects refer to material specific reactions that take place when a fuel is burning. These effects were included in the simulated flames based on real world observations. For example, when wood is burning, it produces crackles and sparks. When plastic is burning, it melts and drips

[39] Gorbett, Gregory E., and James L. Pharr. *Fire Dynamics*. Upper Saddle River, NJ: Pearson, 2011. Print. 75
[40] Ibid 76
[41] Ibid 127
[42] Cafe, Tony. "Physical Constant for Investigators."

in burning chunks. The subsection that follows describes the construction of the flames for the thesis system.

Construction of Flames

This subsection provides an overview how the general flames are constructed using *Unreal 4*'s particle system. To conserve development time, the flames were not built from scratch but built upon existing templates sold on the Unreal Marketplace. These templates are both from Unreal's ``*Content Examples*''[43] and Fred Hooper's ``*VFX Fire Pack*'', [44] which will be referred to as Fire Pack hereon. The Fire Pack has different types of fire effects ranging from candle flames to wild turbulent flames. Turbulent flames include campfires and torch flames. Each fire particle system in this *VFX Fire Pack* can be controlled by parameters to suit an environment. Some parameters are dynamic wherein values can be passed from the blueprint and material network and change at run-time. The rest are predefined at construction to set the initial values.

Fire Material

The flame material[45] is the most important ingredient for the construction of flames. It controls the textures, distortions, colors and emissive color. In order to make flames look realistic, a flipbook technique is used. The flipbook holds an array of images that is rendered in sequence. When the last image in the array is rendered, it goes back to the first image in the array and the process is repeated. The flipbook texture used for the thesis system was taken from the *VFX Fire Pack*.

---

[43] *Content Examples*. Rockville: Epic Games, Fall 2014. Unreal 4 Engine Content Examples Package.
[44] Hooper, Fred. *VFX Fire Pack*. Rockville: Epic Games, 22 Apr. 2015. Unreal 4 Engine Particle Systems Package.
[45] A snapshot of the flame material expression network, created by Fred Hooper, is not included this paper due to copyright reasons.

In the particle system, the flame material is plugged into the Emitter's material parameter under the ``required'' module. In the details panel of the required module, under Sub UV category, it is essential to specify the horizontal and vertical values. This allows the particle emitter to traverse the array of images stored in the flip book texture.

Particle Count

High particle count can make a fire or smoke look very dense and thick. Keeping in mind that each vertex on a burning mesh is spawning a particle system, a mesh of even 200 vertices will be spawning over tens of thousands of particles that could kill the frame rate. Getting the right balance between particle count and desired visual output of flames and smoke took several repetitive trial-error steps. It involved manually tweaking the burst rate of particles (amount of particles released per unit time), the lifetime of particles, the lifetime of particle emitters and the time delays which determine when emitters will spawn. Balancing the time delay and lifetime parameters were important because spawning particles at the same instant and having them actively rendered all at once will drastically lower the frame rate.

Light

By default, the particle systems in the Fire Pack renders with light. This gives the illusion of fire casting light on its surrounding. The ``Light'' module in the particle system allows emitters to render with a light attached to each light particle. This module was removed and not used in the thesis system as it was very costly in terms of frame rate.

Color

The color of the flames as explained earlier is mainly driven by its temperature. Fire is hottest near its center and source and as flame particles draw further away from its source, they begin to cool down. The temperature value is derived from the vertex data structure that holds data such as fuel, temperature etc. There were three ways of passing the vertex temperature value to the particle system to affect the color attributes.

The easiest way was through the material editor by creating a parameter called ``temperature'' that gets its value from a burning vertex through the blueprint actor class. This temperature parameter is then connected to a ``*blackbody*'' material node that changes the emissive color of particles based on how hot a particle is. At the time of this thesis, the blackbody method experimentation showed it was suitable for linear color transitions in flames but not for customized flames like fire-retardant fabric flames.

The second method involves directly accessing the Vector3 values of the color attributes of a particle system. A Vector3 value of a color is broken into Red, Green and Blue values. The second method would require a dynamic assignment of a particle color based on the temperature value. Suppose the temperature is below 1000 degree Celsius, the Vector3 value of a particle color would be (1, 0, 0) which means red. If it was between 1200-1400 degree Celsius, the Vector3 value of a particle color would be (0, 1, 0) which is yellow. As the temperature rises, the Vector3 values will interpolate between them to create a smooth transition from red to yellow by creating the intermediary colors such as red-orange and orange-yellow. This method was not used due to the number of steps and logic operations required to computer the color transitions.

The third method was to simply use the graph editor for color modules for each particle system. The graph editor allows control over the color properties that change over time. However, by using this method, the value of a vertex temperature no longer determined the color of the flame. It is taken over by a curve that is manually plotted and key-framed to specify the different colors at different time throughout the lifespan of a particle. Getting finer nuances in the visual output was possible with this approach. With fire-retardant fabric, an additional factor that determines the color is the chemical composition. Through few trial-error experimental runs, this method showed useful for creating the fire for a fire-retardant material, which required a few different color combinations throughout its lifespan. Using a graph-editor to control the colors allowed greater customization and also less requirement for logic operations. The downside is it is not robust enough to accommodate a need for changes in the color outputs. It would need an FX artist to manual tweak the curve values to get the desired output.

Size

The sizes of the flames are not determined through any dynamic parameters but through the graph editor that sets the size properties of the particles from its birth to death. In general, flame particles begin with a size of zero and rapidly expand to their maximum size by one-third of their lifespan and begin to shrink when two-third of their lifespan. Finer nuances and variations are achieved by creating more than one emitter that spawns particles that grow and shrink at different times and rates.

Emitters

A particle is the smallest unit in a particle system. An emitter spawns one or more particles. A particle system is a group of one or more emitters spawning particles. Each emitter spawns particles based on several parameters that determine the appearance of the particles. A single emitter was not sufficient to create the desired look for different fires. Depending on the type of material the fire was going to burn, the number and type of emitters varied for each particle system. The types of emitters included the following:

1) Different sizes of flames
    - To create density in flames at various stages of a flame growth and propagation, more than one emitter was used to create different sizes of flames. These included starting flames, mid-life flames, large flames and finally plumes. Each flame emitter also has different behaviors at various stages of its particles' life. The starting flames of a wood fire, when small, has low turbulence. As the flames grow in size, its turbulence grows making the flame appear wild.
    - The particle sizes are governed by two modules: *Initial Size* and *Size by Life*
    - The initial size module sets the size of the particles at the time of spawning and the size by life module determines the growth and reduction of particle sizes over the span of its life. Values in the size by life module are set using a graph editor that allows greater control over the sizes of the particles throughout their life.
2) Heat Distortion
    - For all fire types, an emitter was created to spawn particles that create the effects of heat distortion. Heat distortion is the effect of hot air rising off a heated surface.

The heat distortion emitter spawns small amounts of particles (less than 10) of sizes roughly quarter to half the diameter of the entire fire. This is to enable them to create slight warping or distortion effects of the flames to give the appearance of hot flames. Their sizes were derived through trial-error method to ensure maximum effect without compromising framerate.

3) Embers/Sparks
    - Wood produces sparks and embers when burning. The particle system for wood fire has a dedicated emitter to create sparks and embers.
4) Drips
    - As plastic burns, it begins to melt and deform first before burning. When this happens, chunks of melting material break of from the burning body and fall. As they fall, they are aflame. The drip effect is created by adding a separate emitter whose acceleration is set to negative values on the Z axis (vertical direction is represented by +Z axis or –Z axis).
5) Smoke
    - All fires have an emitter that produces smoke particles. These smoke particles vary in sizes, colors, density and opacity depending on the underlying material.
6) Different colors of flames
    - The colors of the flames is described in detail in subsection 3.11 under ``*Colors*''.
7) Spread
    - Flame emitters are spawned at vertex locations on a mesh. Vertex locations form at intersecting edge-loops that run parallel to each other. This caused fire to

appear and propagate in a uniform grid-like manner. To avoid this, a sphere module was added to the flame emitters. The sphere module sets the initial location of the particle emitters within an invisible sphere shape. So, while the center of the sphere is a vertex location, emitters now spawn randomly within the sphere's radius. This makes the fires appear more fluid as they spread.

Observations (Real World)

The following discussion summarizes the comparison of outputs of the various types of flames between the real world and thesis demo. The comparison is based on visual observation of videos of different materials burning. [46,47]

Type	Effects	Smoke	Color	Turbulence	Size
Plastic	• Sputters • Melting and decomposing material breaks into chunks • Chunks drip while still burning in the air	Thin, light ash-colored smoke	Fire burns hot in the center and fades quickly to orange and red	Mild - Medium	Medium
Paper	• Fire spreads and burns very fast	Little to small amounts of smoke	Yellow flames with yellow-orange rims and bright white center	Low-Medium	Small-large depending on surface area
Rubber	• Fire burns very slow • Very slow to ignite • Very difficult to extinguish flames • Sputters	Thick and Dark Smoke produced	Fire burns very hot and it is very difficult to put out burning rubber	Medium-High	Medium-Large depending on surface area
Wood	• Burning process produces lots of crackles and sparks	Thick smoke produced as fuel source is low	Bright yellow fire at the center and core of fire that turn red or red-orange near the exteriors	High	Medium- Large depending size of fuel source

[46] Rusch, M.D. "How Different Fabrics and Materials Burn Cotton, Leather Ect." *YouTube*.
[47] "Textile Fibers Burning Test." *YouTube*.

Fire Retardant Fabric	• Slow burning	Thin and Light smoke	Bright and varied colors depending on chemical composition	Low-Medium	Small
Combustible Fabric	• Fire spreads and burns very fast • Tall flames	Light smoke as fuel source gets low	Bright yellow	High	Medium to large depending on surface area size

Table 6 Summary of outputs of the various types of flames in the real world based on visual observation.

Observations (Thesis Demo)

In order to make fires in games more entertaining, artistic liberty was taken in stylizing the different flames while still ensuring visual variances in the flames. This is also to provide a small level of escapism for players from real world fire.

Type	Effects	Smoke	Color	Turbulence	Size
Plastic	• Melting and decomposing material breaks into chunks • Chunks drip while still burning in the air	Thin light ash colored smoke	Fire burns hot in the center and fades quickly to orange and red	Mild - Medium	Medium
Paper	• Fire has a bright yellow body but the center is bright white • Burns quick • Areas exposed to heat turn brown • Areas on flame disappear as fuel burns • Grid like spreading of flames making the propagation less fluid	Little to small amounts of smoke	Bright yellow flames that turn orange at the edges	Mild	Small

Rubber	• Flames burn for a long time while spreading.	Thick black smoke	Very hot and bright flames. Turns from bright yellow to orange over the lifetime of a flame.	Medium - High	Medium
Wood	• Burning process produces lots of crackles and sparks • Burning surface glows hot	• Smoke produced towards the end of a flame's lifecycle • Thickness gradually increases as a flame wanes.	Dynamic color variations from red-yellow-bright yellow-orange-red.	High	Large
Fire Retardant Fabric	• Burns very hot but slowly • Flame heights are small	Thin small amounts of smoke	Due to being chemically treated, flames have some color variations from white, blue to yellow.	Mild-Low	Small
Combustible Fabric	• Burns very quickly, but not as hot as other materials • Flames start small but scale up very quickly • Flame size is very large due to rapid expansion • More fluid in flame expansion and propagation	Smoke forms when fuel is low.	Body of flames is yellow in color while having subtle nuances of orange color as flame cools towards the end of their lifecycle.	High	Large

Table 7 Summary of outputs of the various types of flames in the thesis visual project based on visual observation.

Particle System Results

Plastic Burning

Figure 20 Drip effects of burning plastic from real-world[48] (right) and thesis system (left). An extra emitter was added within the particle system. The emitter fire particles with negative acceleration creating the effect of burning melting chunks.

[48] *Dripping Fire*. *YouTube*.

Wood Burning

Figure 21 Spark and ember effects when wood burns in real-world (Top) and thesis system (bottom). The wardrobe is of wooden material and when subjected to fire, it produces glowing embers and bursting sparks. The flames seen are also wild and turbulent. The core of the flames are bright yellow and as the particles move further away from the center, they turn orange and then red. Red colored flames are cooler than yellow and bright yellow.

Heat Distortion

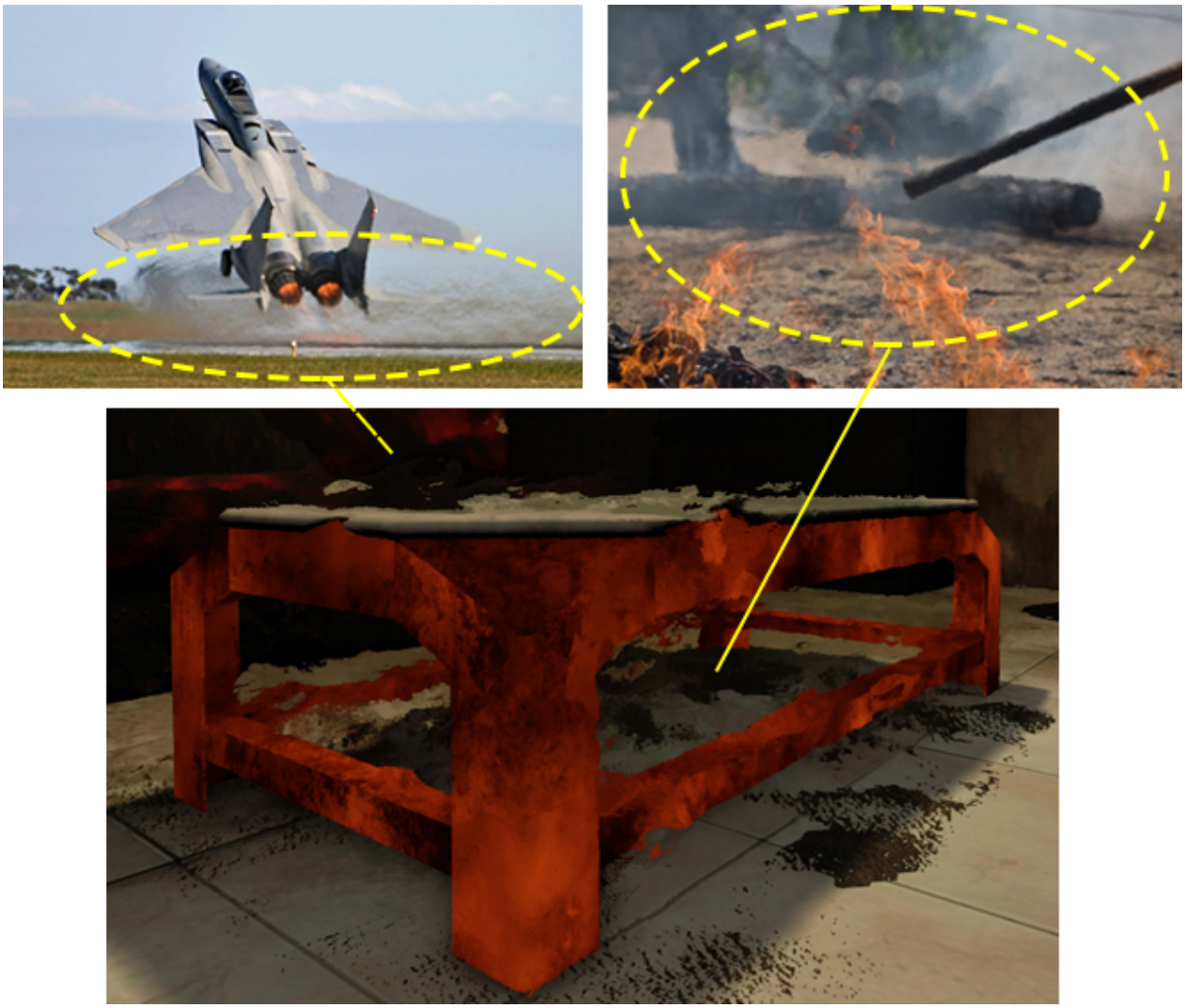


Figure 22 Heat wave comparison between real-world (top)[49] and thesis system (bottom). The selected reference images show heat shimmers caused by light rays entering through an air volume of variable density. For any noticeable heat distortion or heat shimmers, a large surface area subjected to very high temperatures is required. It may not be easy to discern any visible shimmers on smaller burning objects. In a typical game environment when player movement and action sequences are happening very fast, it would require exaggerated heat shimmer effects to capture player attention.

[49] Heat Distortion from Jet Engine. Digital image.

Fire Retardant Flames

Figure 23 Comparison between real-world chemical based fire (top, bottom)[50,51,52] and thesis system chemically treated fabric fire (center). Fire retardant fabric are synthetic materials that may be chemically treated. Some artistic liberty was taken to exaggerate the visuals of such flames. Fire retardant materials do not catch fire easy and if they do they burn hot. Colors of flames change as temperatures and fuel compositions vary. Combustible fabric fire, for the purpose of this thesis, has been assumed to be burning on cotton or feather stuffed material that burn quick and require less temperature to ignite.

[50] "Burning Chemicals!" *YouTube*.
[51] Zavos, Alison. Spectral emissions from different chemicals burning: (left to right) Methane, Calcium Sulfate, Calcium phosphate, Sodium chloride, Potassiam phosphate, Sodium borate. Digital image.
[52] Rusch, M.D. "How Different Fabrics and Materials Burn Cotton, Leather Ect." *YouTube*.

Rubber Burning

Figure 24 Comparison between burning rubber in real world (left)[53,54] and in thesis system (right). Rubber produces very thick smoke when it burns and it is very hard to estinguish its flames. Its flames burn very bright yellow in the center and turns orange and red as the temperature of flames cool. The smoke in the visual project appears darker due to the indoor lighting.

[53] *The Tyres Emitting Hazardous Smoke at New Baneswhor*. Digital image.
[54] *French Goodyear Workers Burn Tyres in Riot against Bosses | The Times*. Digital image.

4 POSTMORTEM

The visual component of this thesis is not without its limitations. These limitations are discussed as follows. Future iterations of the thesis system will include improvements, which are being currently researched to address these limitations.

4.1 Alternative to Vertex Based Propagation

At the time of development, the main limitation was with using vertices as points of propagating fire as they are not dynamic. Each time game designers or artists require a change in the density of flames appearing on a mesh, they would need to edit the mesh in a 3D modeling software by reducing or increasing the number of edges and vertices. This increases production time unnecessarily.

At the same time, cellular automata techniques as seen in *Far Cry 3* require additional computation to check for collision on a mesh to determine the shape and surface of an object to propagate fire to ensure the interiors or non-player viewable areas are not on fire. While analyzing these limitations, an alternate technique was determined. In this technique, the idea is to use an object's UV information and convert them to world space locations. Using only these locations, a grid of cells can be created. These cells will be evenly spaced cubes. The density of the grid can be determined through procedural controls. As UV locations provide only the surface information of a mesh, the need to calculate collision shape of a mesh may be reduced. The grid of cubes will be generated only on the surface of an object, which may represent the shape of the mesh. This grid can now be used for propagating fire. This technique also eliminates the need for calculating vertex positions. This will greatly improve production time as developers need not

modify a mesh's geometry repeatedly to achieve the desired flame density. The implementation of this alternate technique can be expected in the future.

4.2 Propagation on Skeletal Meshes

The second limitation faced at the time of building this thesis visual component was that the *Victory Plug-in* used in the development of the MBPD tool did not allow for calculating vertex locations on skeletal meshes. Skeletal meshes include objects which have morph target information or animation data. The current system is therefore limited to spreading fire only on static meshes. Future development will ensure propagation on skeletal meshes.

4.3 Structural Damages to Burning Meshes

Another feature to expect in the next version would be the breaking or crumbling down of burning objects. When objects are exposed to extreme heat and burn down, their structural integrity is weakened. For example, wooden objects, when burnt, crumble and fall apart. The current system developed for this thesis does not yet have this feature and is considered for future development. The destruction effect is achieved by converting a scene mesh to a destructible actor and create fracture data along a mesh. Fracturing a mesh is done through in-built game engine tools such as Voronoi Fracture, which allows a mesh to be divided into several smaller chunks by creating fault lines (cracks). This is done at pre-compile time. When this object is exposed to a physics impact or damage in in-game, the object crumbles to smaller pieces.

4.4 Orientation Based Propagation

The concept behind Orientation based Propagation is that the direction of the movement of flames, and the orientation of the object on fire determines the rate of fire spreading. Flames

moving upwards burn and degrade objects faster than flames travelling downwards. Take, for example, paper. A burning paper held upside down will burn and degrade much faster than when held upright. This is due to the amount of surface or fuel contact available to the flames. When held upright, the flames move downwards burn slower. In many cases, the fire will extinguish before consuming the entire paper. Distance also plays a very important role here. If a plastic bottle is kept half a meter away from a fireplace, it will not burn but its structure softens and loses its form. However, if it is exposed directly to the fire, it will melt, deform, and catch fire.

4.5 Layered Propagation & Degradation

Many objects are made of more than one type of material. The material construction or constitution could be either modular or layered. Example of a modular construction would be a dresser, whose frame and drawers are made of wood while the handles are made of metal. Propagation of fire along modularly constructed material surface objects are straight forward and achievable in the current system developed for this thesis. Example of a layered constitution could be a study desk whose exterior is coated with a sheet of plastic while its interior is wood. If this desk comes in contact with heat or fire, the plastic surface will burn and degrade differently than the wooden interior. The current fire propagation system constructed for this thesis does not allow layered propagation & degradation, and development of this feature will allow further visual variations in burning effects.

5 CONCLUSION

This thesis has discussed and demonstrated in the visual component a different approach to current fire propagation techniques. Adding an intelligent fire propagation system will help

game developers design pyro levels with greater ease. Adding procedural controls allows designers to test an environment subjected to fire modularly and rapidly. It will reduce the need for manual scripting and hardcoding of fire propagation paths across a game level. Material based propagation and degradation does not discount the current fire propagation techniques practiced in the industry but serves as an improvement or add-on to these methods. Adding material based degradation due to effects of heat and fire along with material based fire propagation will cumulatively increase dynamic visual variations in the appearance of fire and textures in a burning or burnt environment, thus breaking the homogeneity in flames. The combination of material based degradation and propagation will certainly increase player experience.

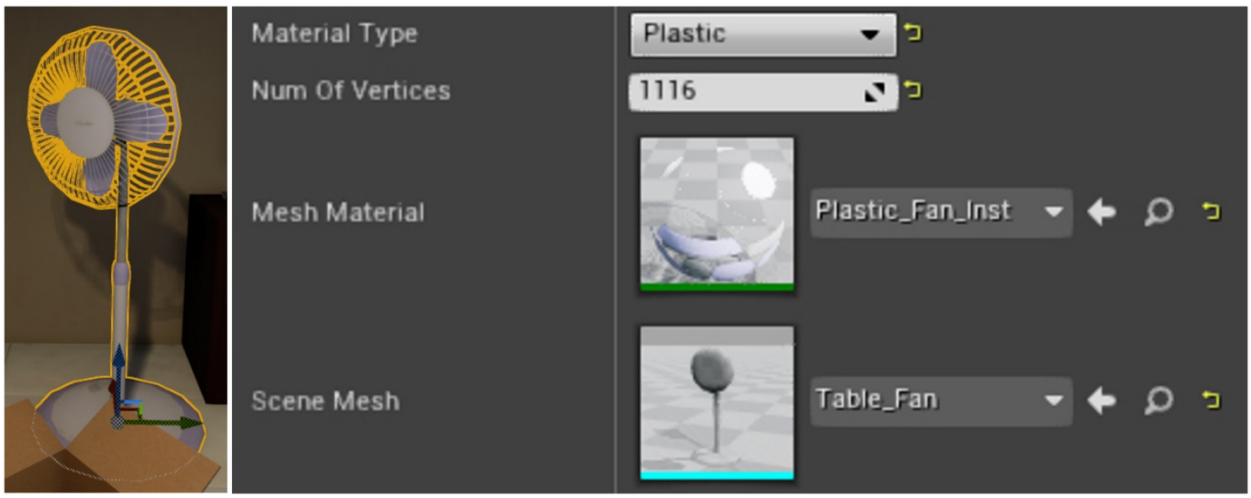
Appendix A

Constructor

The actor class's constructor allows predefining of certain parameters required for calculating propagation and degradation. They are split into editable and non-editable parameters. Editable parameters allow the designer to set custom values for more detailing. They remain static values at run-time, and the engine will not override the values. Non-editable values are predefined and can only be modified by the system at run-time.

Essential Editable Presets

Essential Editable Presets are required for the initial setting up of an object that will interact with fire. It is important to set the values in this preset section correctly as it will ensure proper appearance of the object in the level. The essential attributes of this actor class were determined to be the following:



A1 Actor (mesh) with basic parameters

1) Scene Mesh – The 3D model which is assigned to the actor script. The number of vertices of the model are determined at asset creation in a 3D modeling software like Maya/3DS Max. The Victory Plug-in automatically calculates the number of vertices on the assigned mesh. The number of vertices is non-editable and is derived automatically upon mesh assignment.
2) Mesh Material Type – The type of surface that is going to interact with fire in the level. They are fire-retardant fabric, combustible fabric, wood, metal, plastic, paper, hard-paper, rubber, concrete and ceramic or marble.
3) Mesh Material – The visual appearance of the mesh. The material carries the textures and surface appearances of the mesh. The textures include both the pre-burnt and burnt states and in some meshes, the mid-burnt states. An instance of a Master Material Class is created by the constructor and assigned to the mesh. A more elaborate explanation of the Master Material Class will be provided in subsequent sections. The Master material class drives the degradation of a mesh's surface.

Advanced Editable Presets

The advanced editable presets provide designers greater control over the how fire would spread across an object and how the effects of fire damage will appear across the object's surface. The following highlights the advanced editable presets:

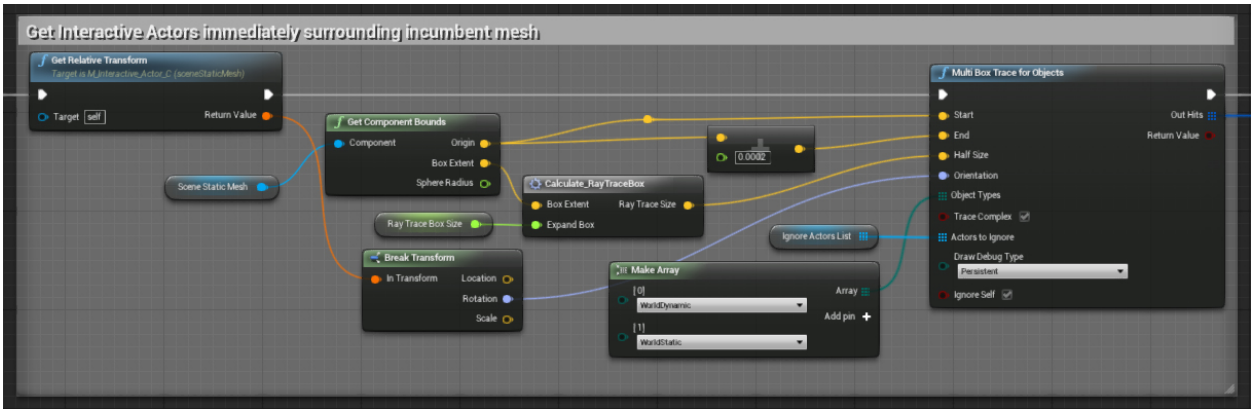
1) Deformation (min/max) – As the object burns, how much should its shape deform? This value is supplied to the displacement map of the mesh. The higher the value, the more the mesh deforms.

2) Heat (min/max) –It affects the heat distortion seen on hot surfaces.

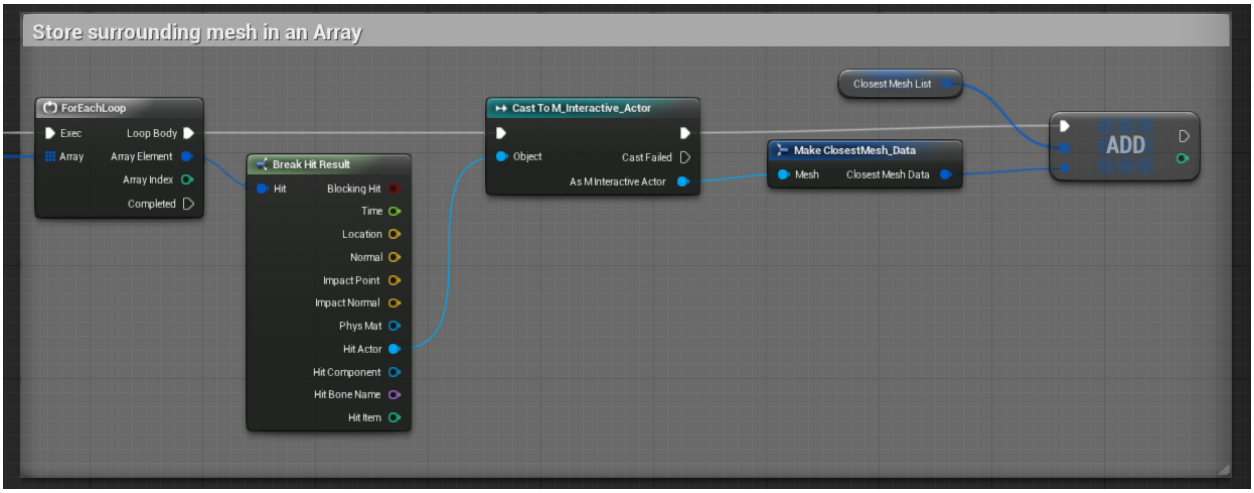
3) Emmisiveness (min/max) - This value determines how much an object glows as it is burning. Higher values result in a brighter glow.

4) Ignitable distance – If an object is combustible, what is the minimum distance between a burning vertex of mesh A and an unburnt vertex of mesh B before Mesh B catches fire?

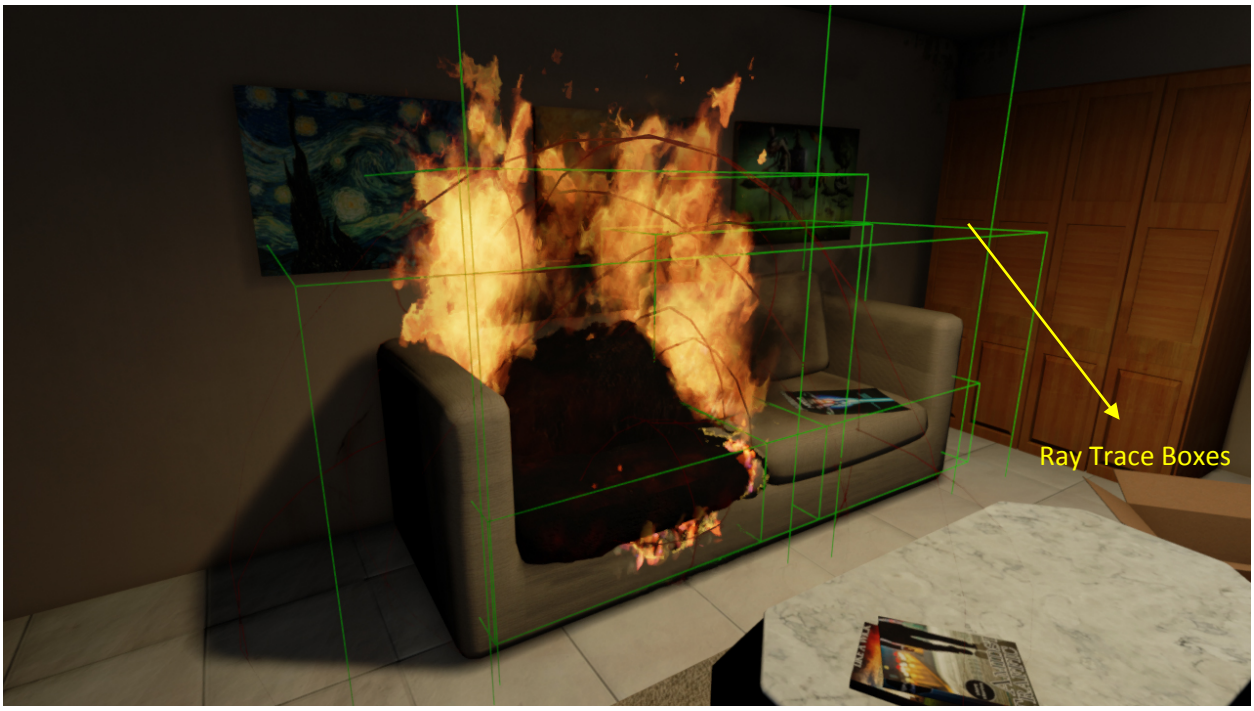
5) Ray-trace box size – Each actor/mesh in the scene has an invisible ray-trace box which is checked to see if there is another mesh's ray trace box overlapping. If there is an overlap, both meshes are checked to see if there are any burning vertices and unburnt vertices within ignitable distance.



A2 Create ray-tracing box to check for actors (meshes) near burning mesh



A3 Store nearby meshes in an array

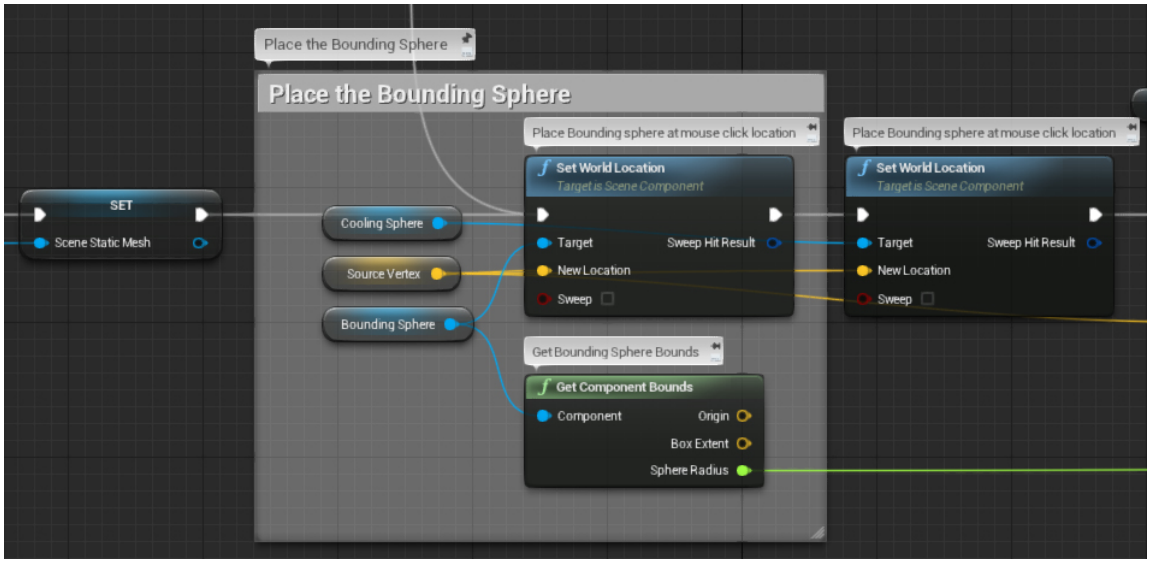


A4 Ray-trace boxes wrapping meshes

Advanced Presets (Not Editable)

Non-editable advanced presets are parameters whose values are set at compile time. These values are derivation from several trial-error runs. If there are any changes required, they have to be manually changed within the blueprint editor.

1) Mesh Fuel Amount – This is the overall fuel for a given mesh of particular surface type. Each surface type is given a different fuel amount. Higher fuel amount means the mesh will burn longer. Based on this fuel amount, a minimum (80%) and maximum (120%) value is derived and allocated to each vertex. At each event tick, the fuel amount for each vertex is reduced at a preset random value range. As each vertex catches fire, they burn and die out at different durations.
2) Sphere Increment & Ignition Delay – In order to simulate auto-ignition of a mesh/vertex, an invisible sphere is spawned at the source of a fire. The source of fire is determined by the player when he/she clicks a surface. The point of click attaches itself to the nearest vertex and spawns the first fire particle at the vertex location.



A5 Place a bounding sphere with the mouse click location as its center of origin

At every event tick, the sphere grows at a predefined value range depending on the type of surface. As the sphere grows, it begins to envelop vertices. Every vertex that falls within the radius of the sphere begins to raise its room temperature. As this room temperature reaches the auto-ignition temperature, a particle emitter is spawned. The rate at which the sphere increments is determined by the ignition delay. Although not physically accurate, this method circumvents the need for complex thermodynamic calculations seen in real-world fire simulation systems.

3) Combustibility – A Boolean value that checks which objects are combustible and which are not. Metals, ceramics and concrete are examples of non-combustible surfaces while plastics, wood, and fabric are examples of combustible surfaces.
4) Flame Template – Each surface type is assigned a particular fire particle system to ensure visual variations. For example, burning plastic objects will depict the characteristics of fire on plastics while the template would be different for wood, paper or fabric.

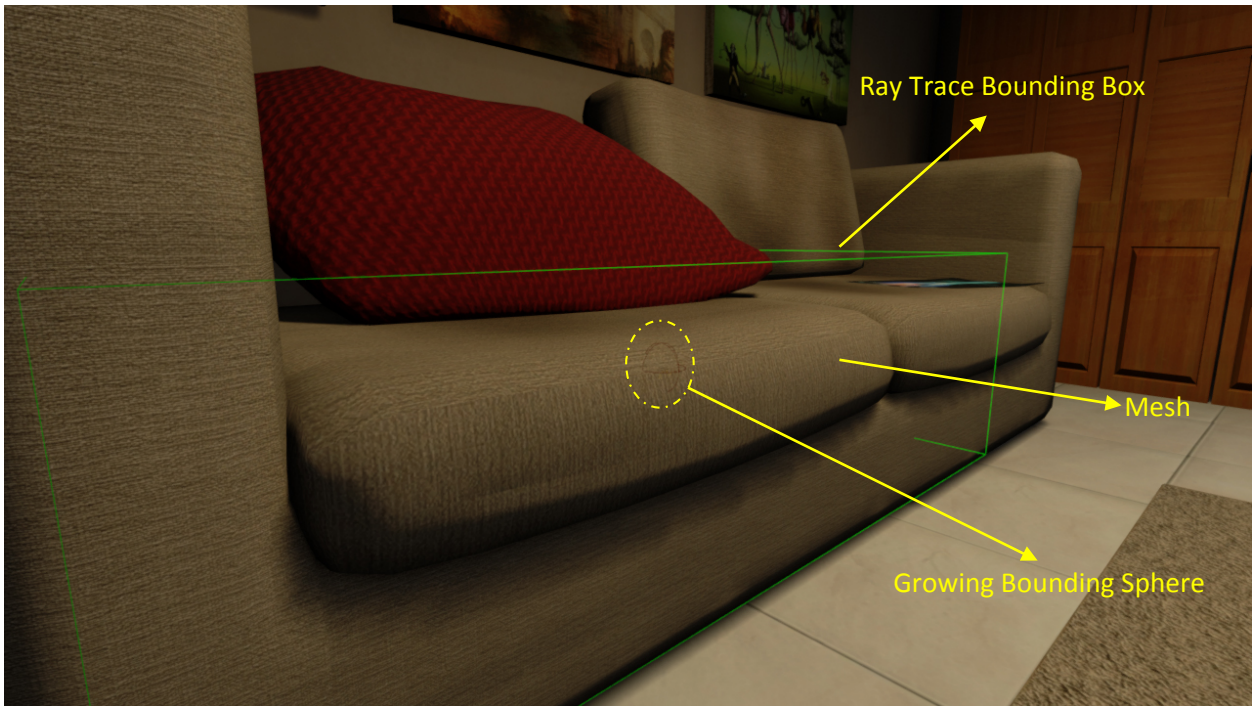
Appendix B

In the visual demo, an actor class called an ``Interactive Actor'', hereinafter referred to as IA, was created to contain a script that runs the propagation and also allow the designer to assign a mesh, surface type information, material/texture and essential thermodynamic properties. This self-contained actor class allows a mesh to interact with fire.

The IA accepts user input with a click of the left mouse button. When an IA is clicked, it cannot be clicked again. While an IA object can in reality be set fire in multiple locations, implementing it posed technical challenges. Thus, a constraint was imposed to simplify the propagation system by allowing only one source of propagation per IA object.

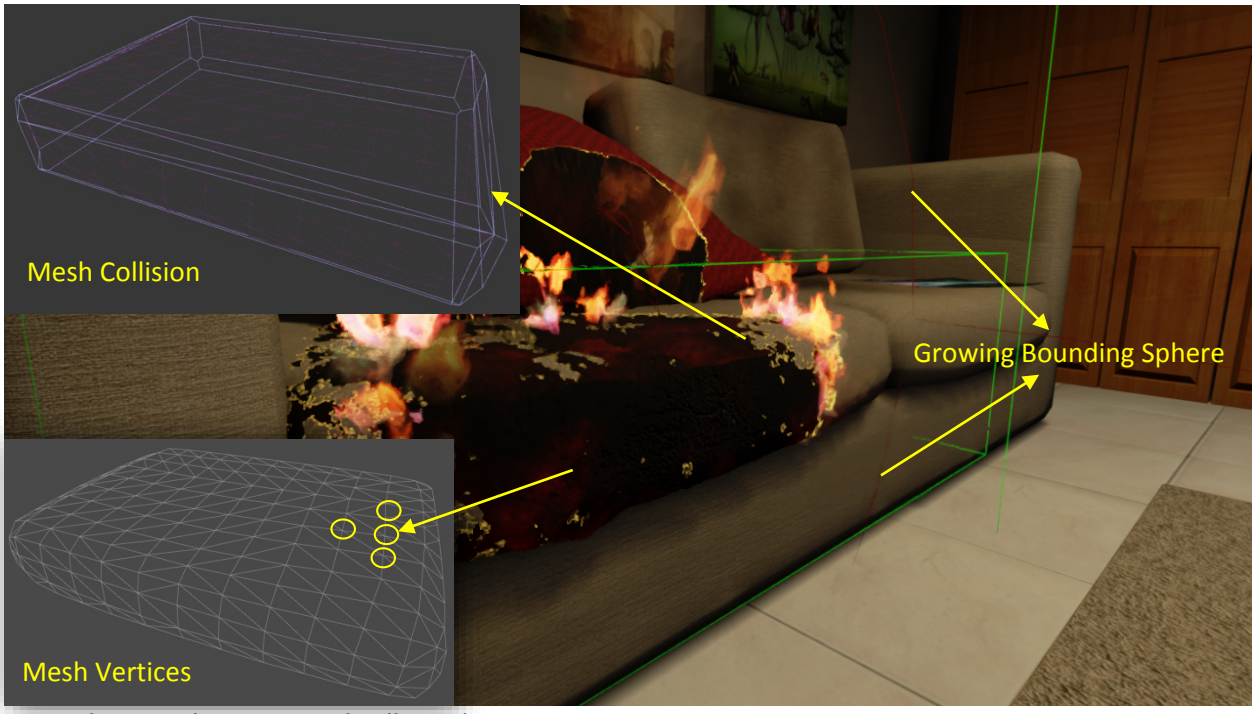
Using the mouse input, two hit results are extracted, namely, “Location” and “Hit Actor”. The location refers to the position of the mouse click at world space and the hit actor is the IA object (or the 3D mesh) the user clicked. This hit actor is stored and referred to as “Scene Mesh”. The scene mesh’s vertex positions are stored in an array called “Original Vertex List”. The system computes the closest vertex to the click on that scene mesh and stores that single vertex’s position as “Source Vertex”.

At this point, an invisible bounding sphere is spawned at the location of the source vertex. This bounding sphere’s radius is incremented by a value called Sphere Increment, which set in the IA class constructor.



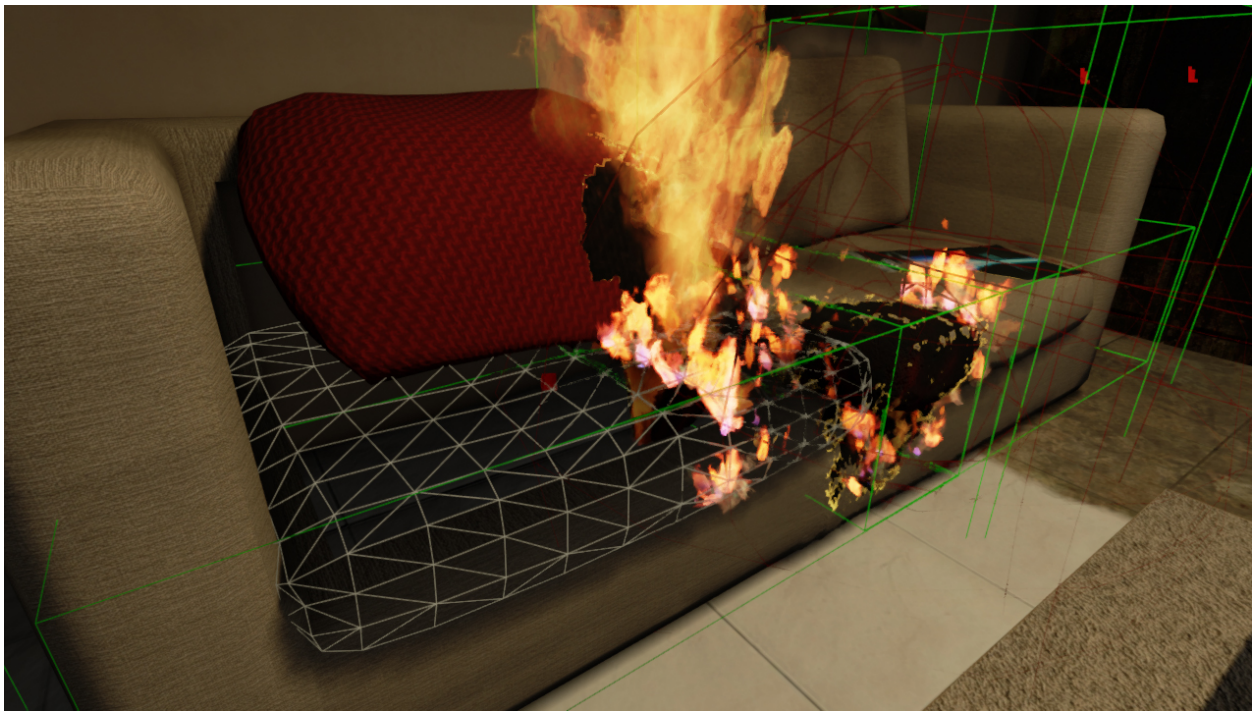
B1 Visual output for the growing sphere

The rate at which the sphere increments at every system tick is determined by the Ignition Delay value. Objects which catch fire easily have higher values for the Sphere Increment & lower Ignition Delay. This means more vertices are engulfed by the Bounding Sphere at a faster pace. On the other hand, a lower Sphere Increment value with high Ignition Delay results in a slower pace of vertices catching fire. Together, these two values simulate the aesthetic "feel" of objects catching fire and burning, either slower or faster.

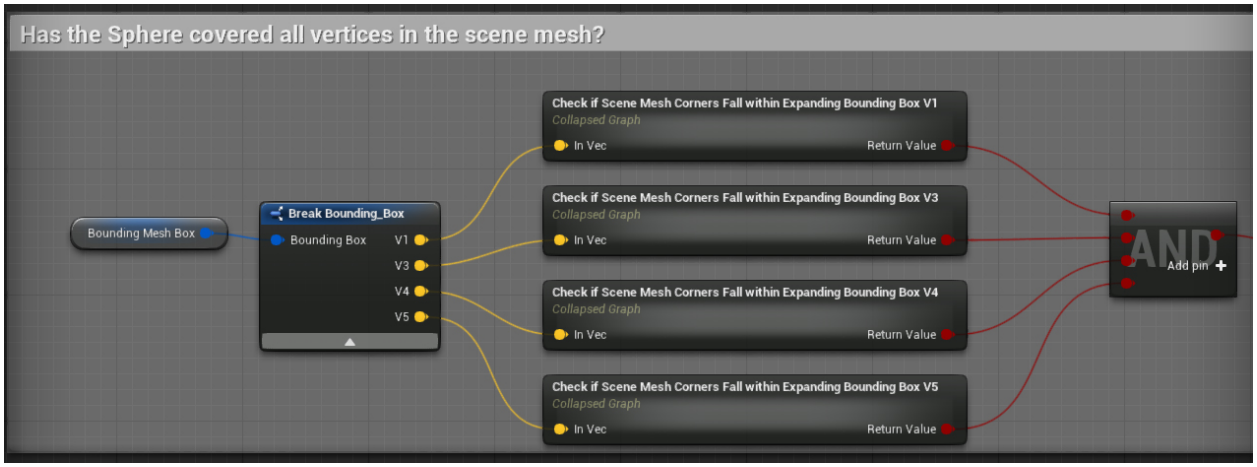


B2 Mesh vertex locations and collision box

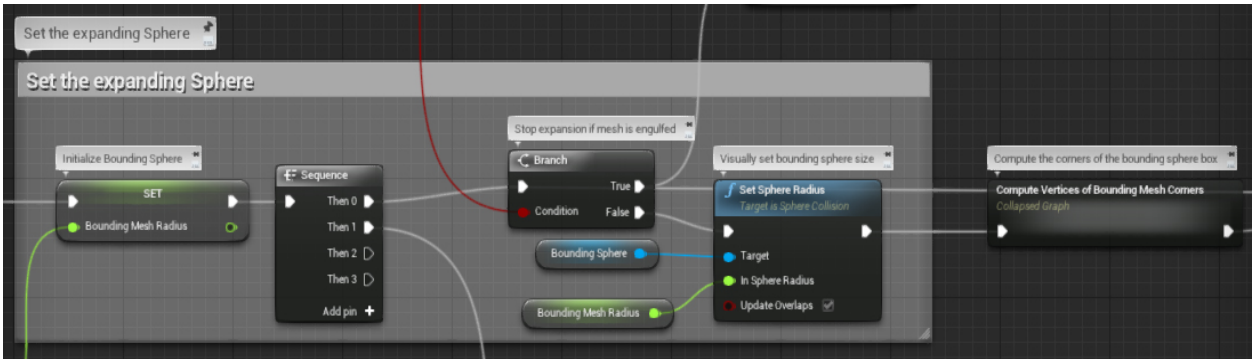
As the bounding sphere grows, it checks to see which vertices on the scene mesh, stored in the ``Original Vertex list'', are coming within its radius. The bounding sphere continues to grow until it envelops the entire mesh's bounding collision box.



B3 Visual feedback for vertices of a burning mesh



B4 Check if growing bounding sphere has covered the corners of a mesh's bounding box



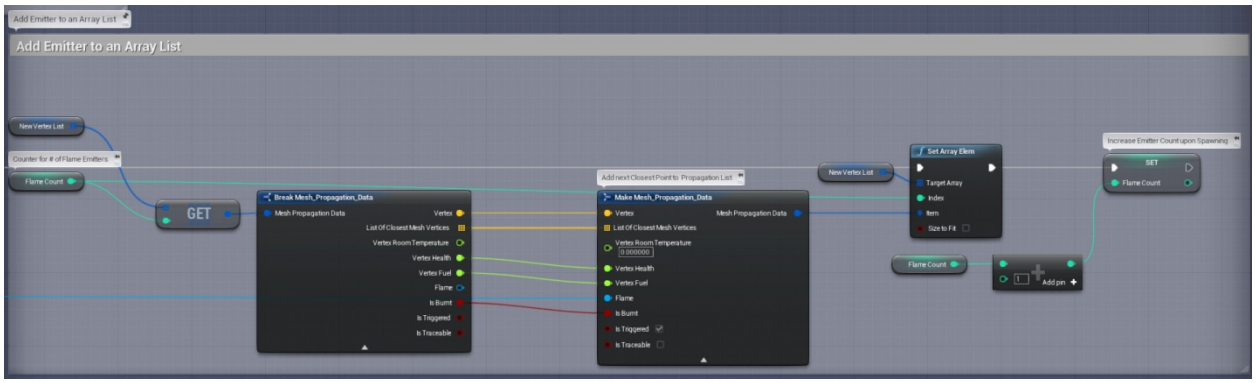
B5 Increase the radius of the bounding sphere

Based on the first-in-first-out principle, vertices are stored in a new array called ``New Vertex List'' in the order in which vertices fall in the bounding sphere's radius. This re-ordering is necessary to prevent the mesh from spawning fire from random vertices. As they are added, each vertex is assigned the following attributes and saved in a data structure called ``Mesh Propagation Data'':

1) Mesh Vertex – The world space position of the vertex
2) Vertex Temperature – The initial room temperature of the vertex. Its range is set to begin between the ranges 24 to 33 degrees Celsius. This temperature drives the color of the flames through a material node called "Blackbody".
3) Vertex Fuel – The starting amount of fuel for each vertex. Its range is set to begin between the ranges 80% to 120% of the mesh's overall fuel amount. This allows variations in the rate at which flames die out.

As vertices are added to the New Vertex List, the system runs the thermodynamics calculations at every Event Tick. At each event tick, operations relating to each vertex and its parent mesh are run in sequence:

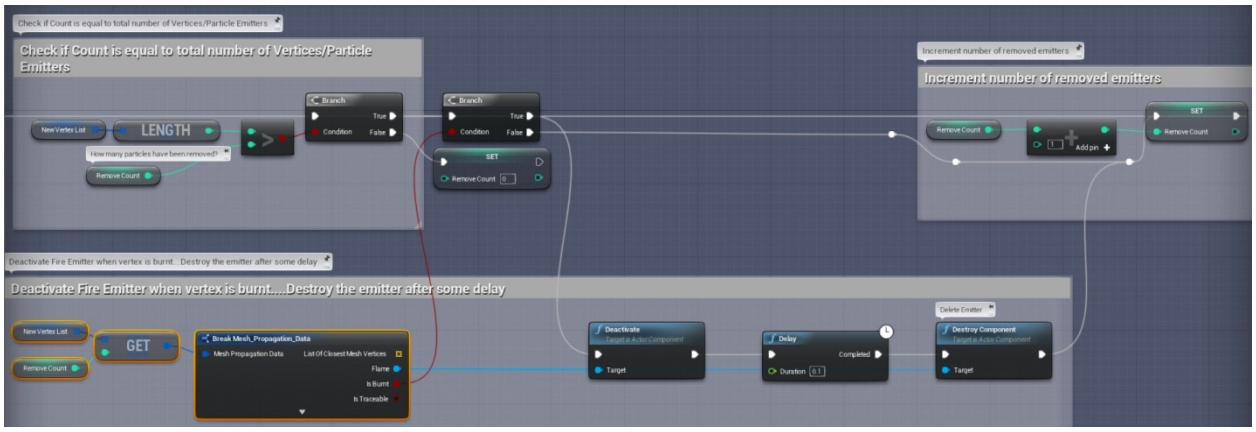
1) Spawning fire – Each vertex in the order in which it is added (First-in First-out) to the New Vertex List is checked to see if it is burning or burnt. If either states are false, the vertex's temperature value is incremented. With each iteration, the vertex's temperature continues to rise. As soon as it reaches the Ignition Temperature value or above, a particle emitter is spawned at the vertex location. Once spawned, the vertex's state is changed to burning and its fuel value begins to reduce. The type of spawned fire is determined in the IA class's constructor.



B6 Add a flame emitter to a vertex

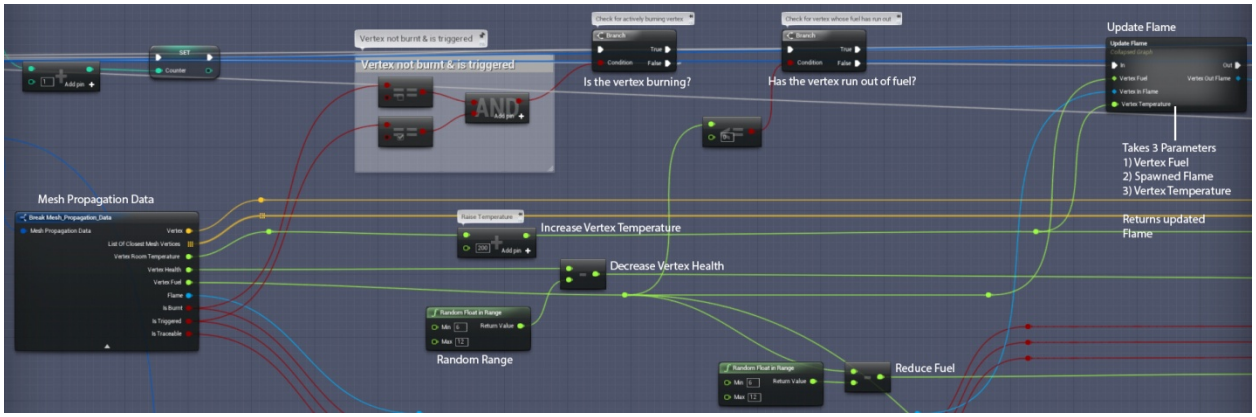
2) Following the spawning of a fire emitter on a vertex, the vertex's state is updated at each event tick. The following describes the various state updates:

- State check - Vertex is checked to see if it has not burnt out. A vertex is flagged as burnt out when its fuel is zero or below zero.
- If it has burnt out, the particle emitter attached to the vertex is removed from the system to clear memory space.



B7 Removing flame particle emitters from vertices that have no fuel

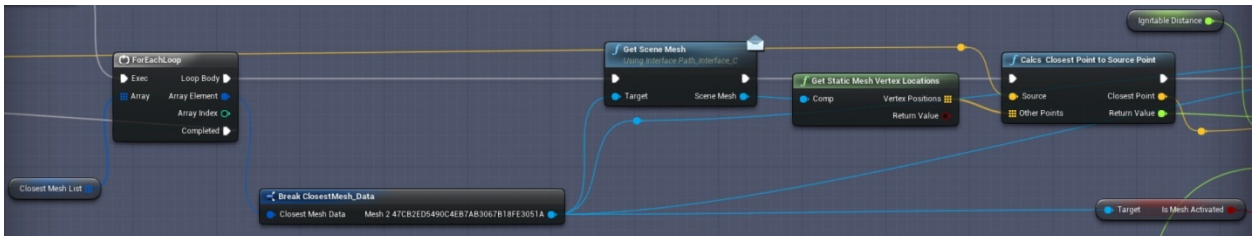
- If it is still burning, the vertex's temperature is incremented at each event tick. The temperature of the vertex affects the color of the flame.



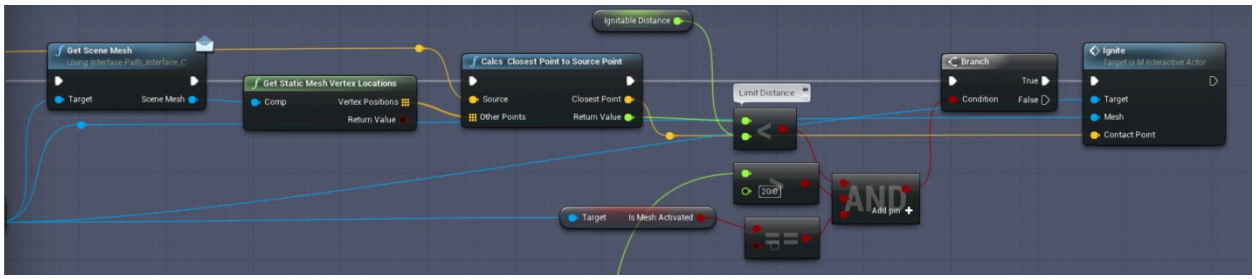
B8 Updating vertex state and flame properties

- An actor (interactive mesh) that is on fire has an invisible ray-trace box surrounding it. The actor actively checks to see if its own ray-trace box is overlapping ray-trace boxes belonging to other meshes. Two actors with overlapping ray-trace boxes indicate proximity between them and that their vertices can ignite each other. Once a vertex is on fire, it constantly checks for vertices belonging to other actors. The vertex with the closest proximity will catch fire. Both the ray-trace box size and the minimum ignitable distance between two

vertices can be set by the designer. If there are no meshes within ignitable range, fire will not propagate to other meshes.



B9 Checking for the closest vertex on the closest mesh

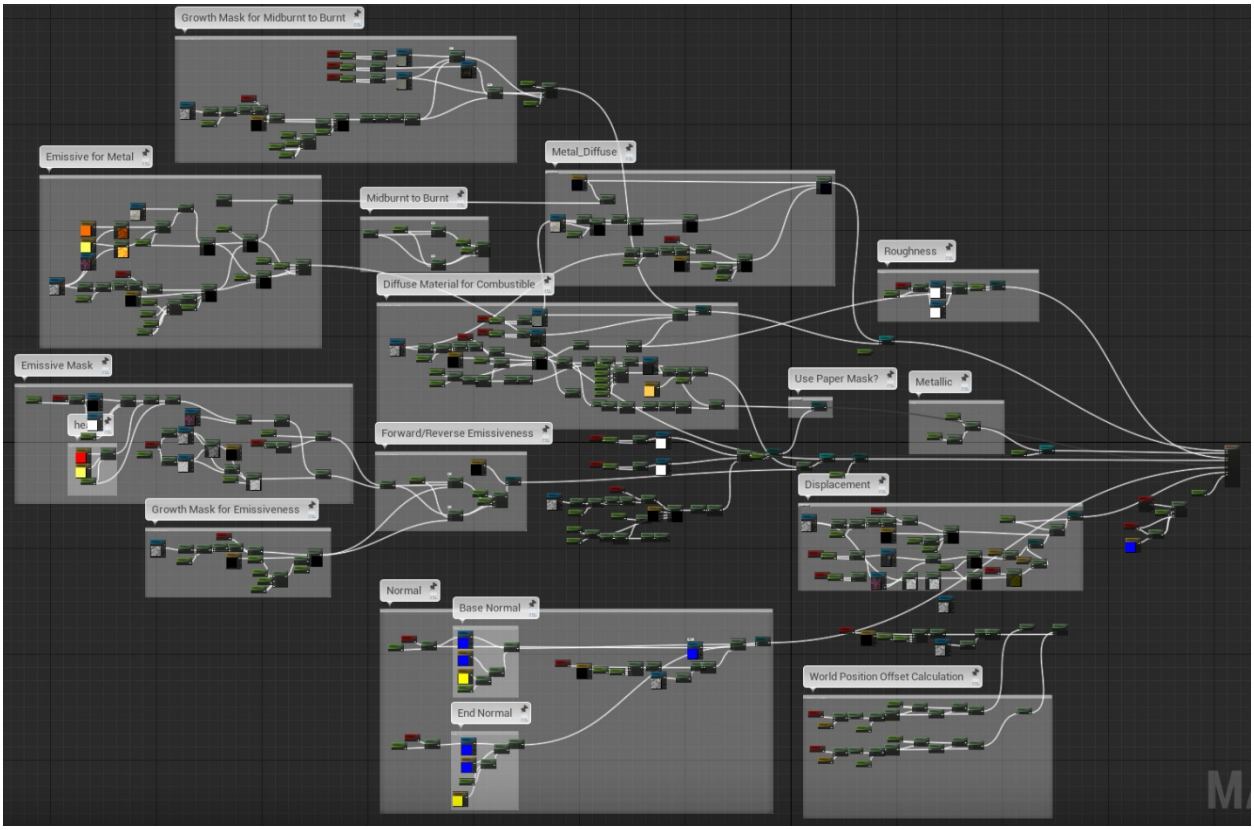


B10 Checking for a vertex on the closest mesh within ignitable distance and igniting that vertex

3) Once an actor (mesh) has caught fire, it is subjected to material degradation. This will be elaborated more in the next section.

Appendix C

The Master Material Class (see C1) is made of several properties to define the surface information of a degradable object. These properties are Base Color, Emissive Color, Metallic, Roughness, Normal, Displacement and Opacity. Details on the construction of these properties follows.

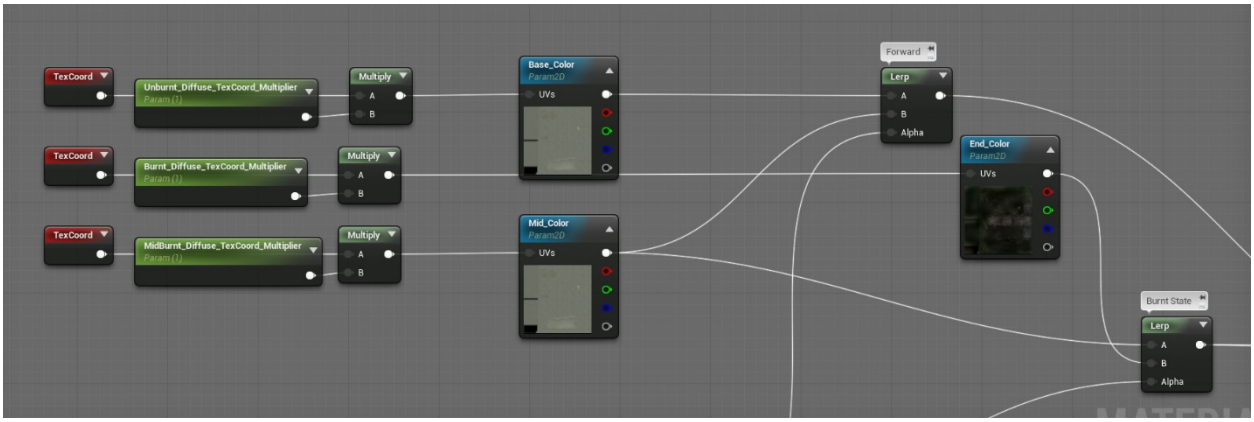


C1 Master Material Class

Base Color

The Base Color drives the overall color information of the material to be applied on an object. It accepts texture samples that allow the user to specify textures from image files (see C2). These textures are 1) pre-burnt state texture, 2) post-burnt state texture, and 3) an optional mid-burnt state texture. They are connected to a ``Lerp'' node, short for Linear Interpolation. The

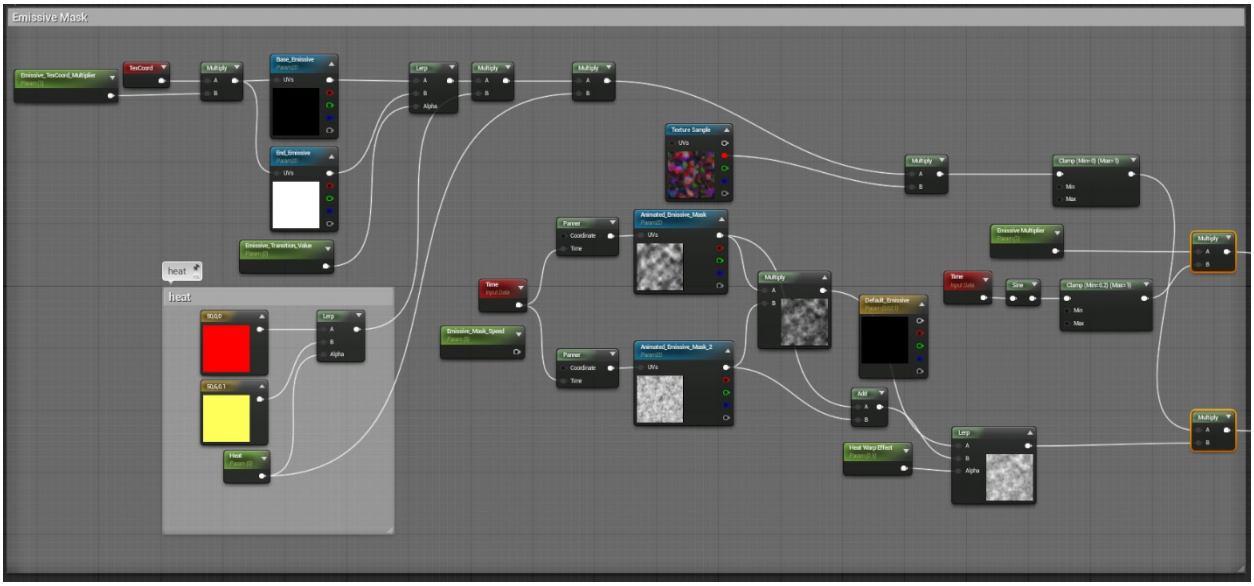
lerp node allows transition between two materials or textures based on the alpha input. The alpha can either be another texture sample or a set of material expressions to calculate how the transition should occur. The lerp node is used in other material properties such as emissive color, normal etc. This allows similar transition between two states, pre-burnt to burnt. More explanation on material transitioning in the system is given in the following sub-section.



C2 Base Color material expressions

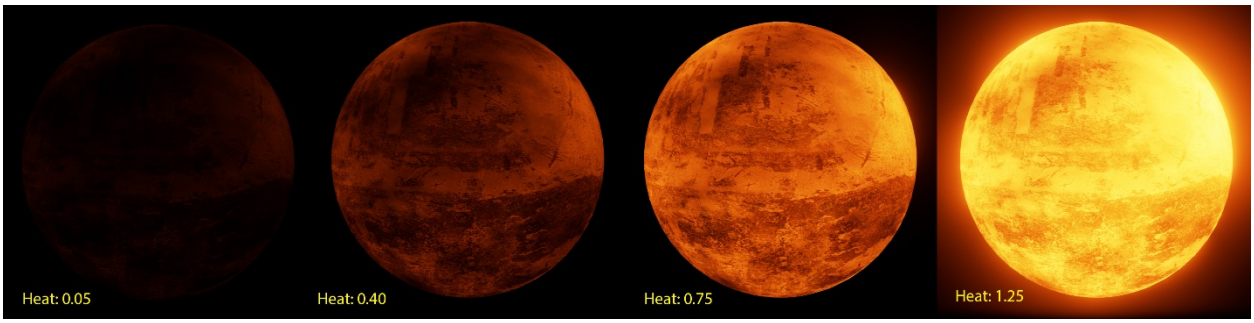
Emissive Color

The emissive color controls how much glow or light emission a burning surface gives out. The construction of the emissive color was very challenging. It was constructed using three groups of material expressions: 1) Emissive mask for combustible objects (See C3), 2) Emissive mask for metallic objects (See C5), and finally 3) Growth mask (See C6) and Forward/Reverse expressions to control the material transition between the unburnt, burning and burnt states of a surface (See C7). The set of material expressions under growth mask allows a burning object's surface to incrementally glow and also fade away as the surface cools down.

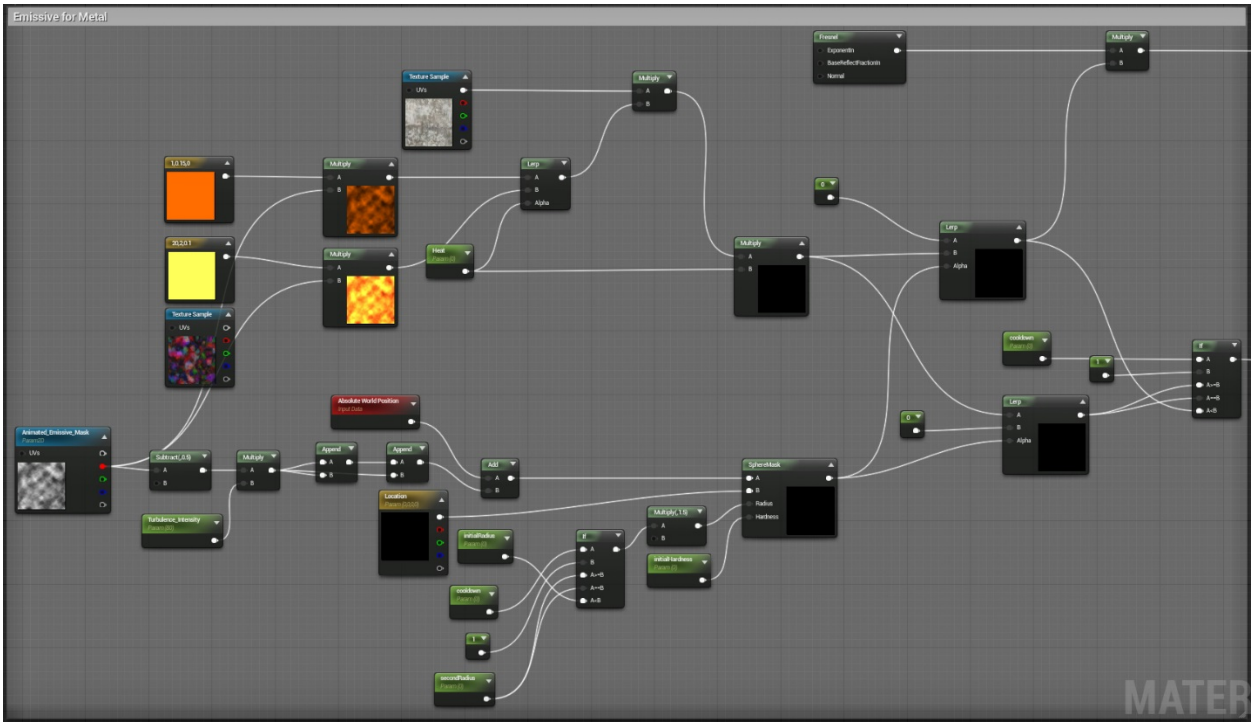


C3 Emissive mask material expressions for combustible surfaces

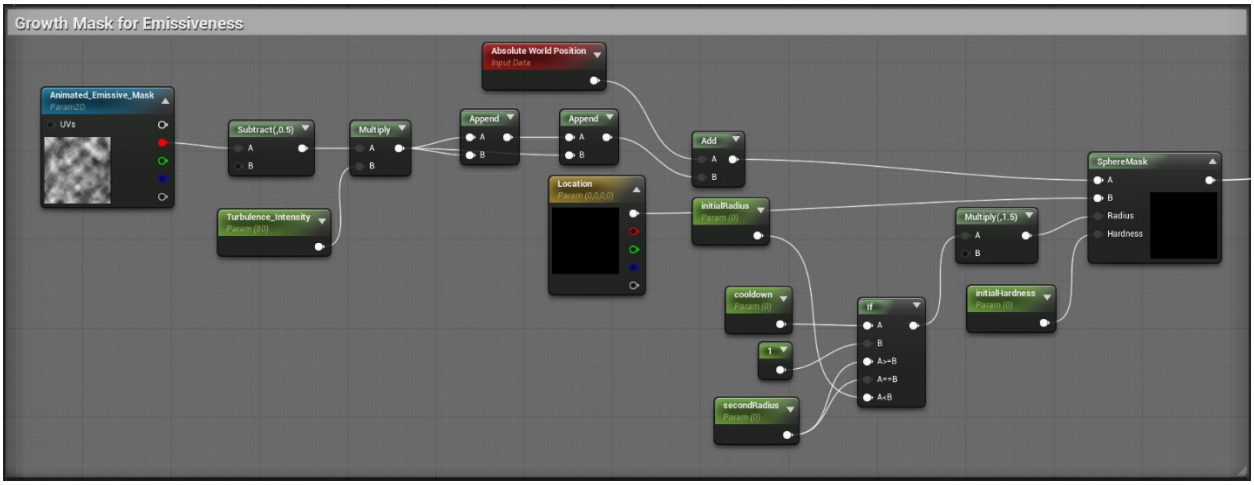
An assumption was made, for simplicity, that, unlike combustible objects, metallic objects do not burn away. They would slowly heat up and glow red (See C4**Error! Reference source not found.**). Therefore, it was easier to create a separate group of material expressions for emission on metallic surfaces despite few repetitions.



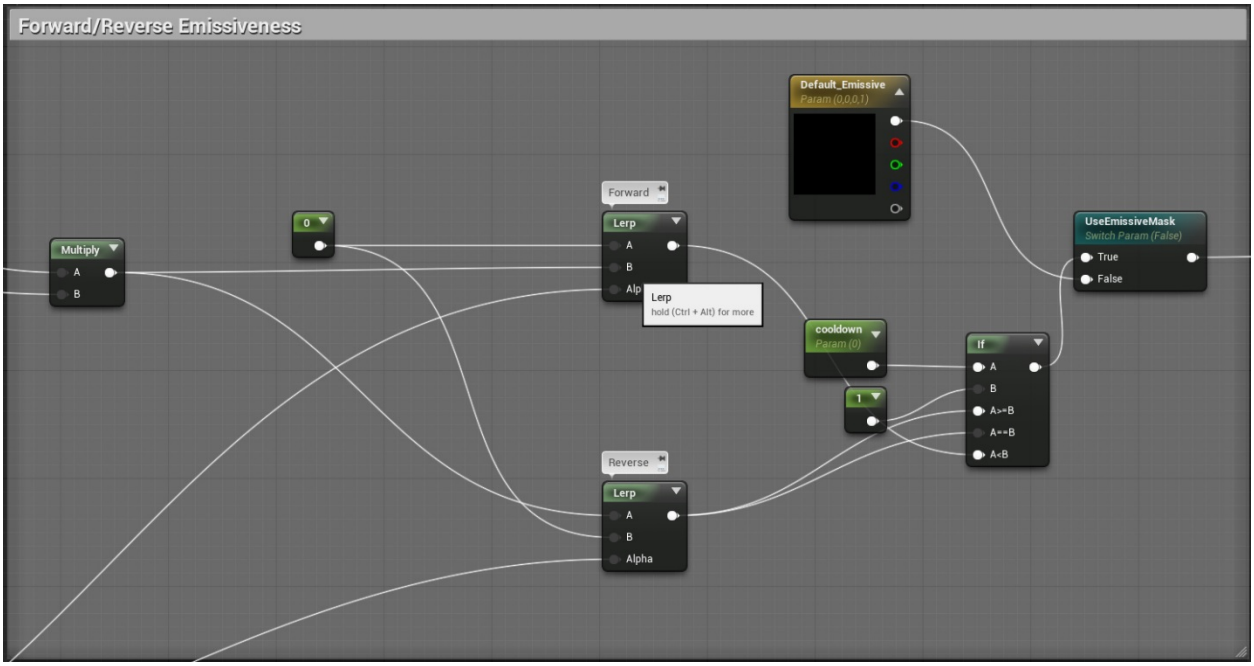
C4 Different values for the Heat parameter to control emissive color of heated objects



C5 Emissive mask material expressions for metallic surfaces



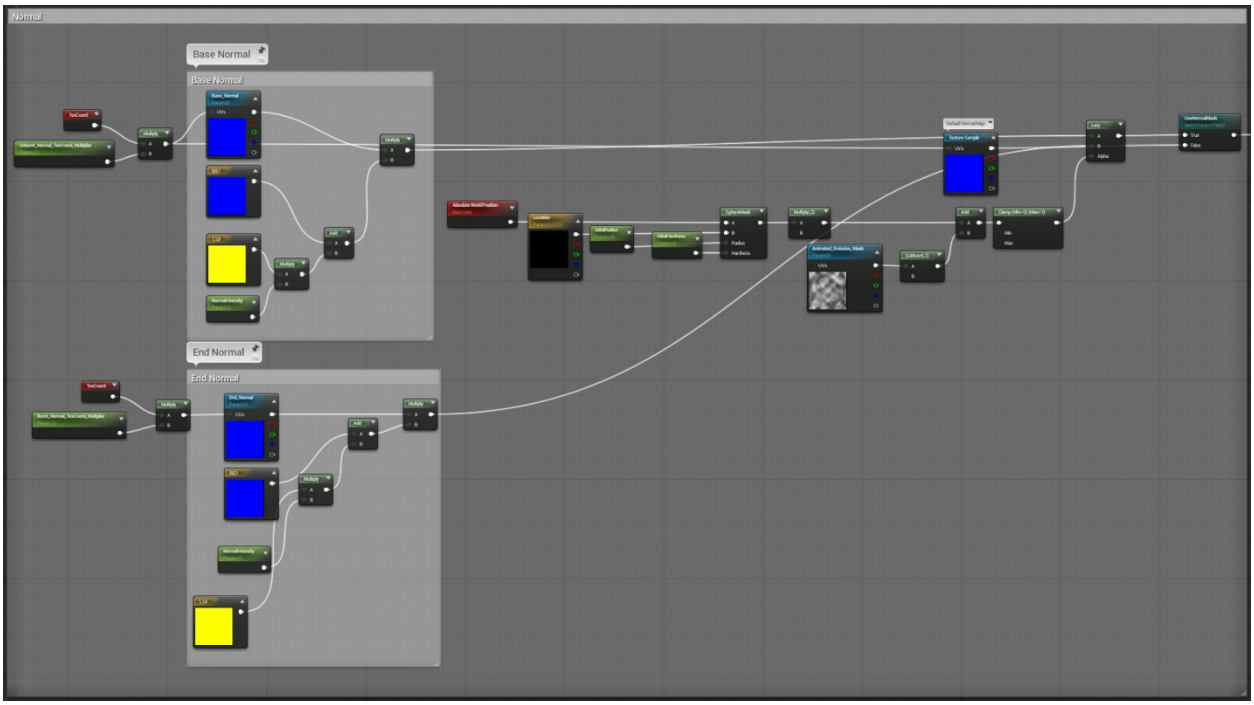
C6 Growth mask for emissive color



C7 Forward/Reverse mask for emissive color

Normal Map

The normal map properties were used to create surface details and illusion of bumps on the surfaces of burning objects (See C8). For example, the bubbles formed on a heated plastic surface and rough bumps on burnt fabric were achieved through normal maps. The normal map property, just like base color input, allows the designer to set both pre-burnt state and burnt state. The system will then transition between the two states as the object is degrading.

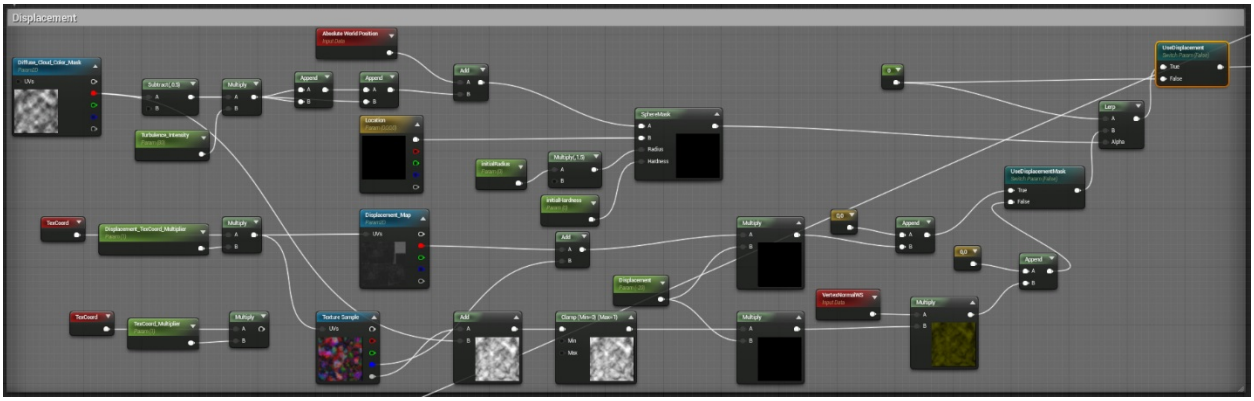


C8 Normal maps material expressions

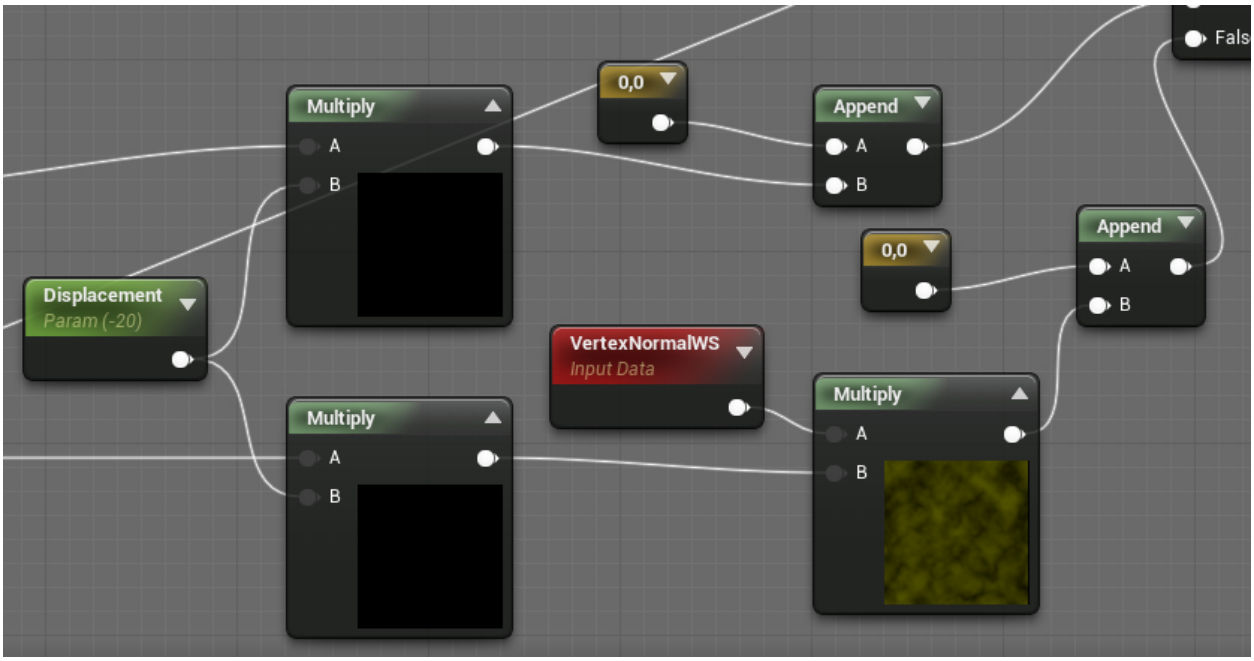
Displacement

One of the limitations of the current thesis demo is that it does not allow fire propagation on skeletal meshes. Skeletal meshes carry morph and animation data. A morph target ``is a snapshot of vertex locations for a specific mesh that have been deformed in some way'' (Epic Games). It allows the deformation of a mesh's shape. As a workaround, displacement maps were used to drive the deformation of a mesh (See C9). Although the results did not meet the deformation quality morph targets could have achieved, displacement maps were fairly convincing to show meshes have been burnt. The displacement network allows usage of custom displacement textures for each object in the scene or a default texture. The intensity of the displacement can be controlled parametrically. Initially, the displacements were causing excessive deformation on the x and y axes of the mesh (sideways). Therefore, meshes appeared very skewed and warped. To counter this, a limit was set to the x and y axes to prevent any

sideway deformation and allow only vertical (z-axis) deformation (See C10). The results were more natural in that as meshes underwent heat and fire, they appeared to be melting and gravitating towards the ground.



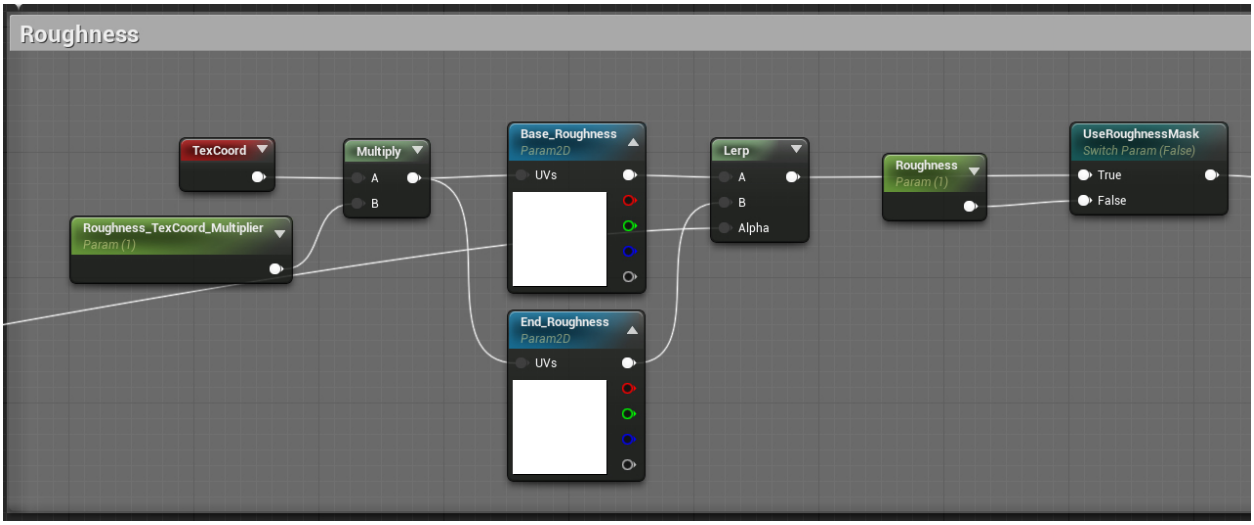
C9 Material expression for creating World Position Offset for displacement effects



C10 Expressions to limit displacement along vertical (Z-axis) direction only. Initial displacements caused excessive sideway deformations. To limit this, expressions were used to ensure the deformation would occur along Z-axis or vertical direction giving the illusion of mesh melting and effects of gravity.

Roughness Mask

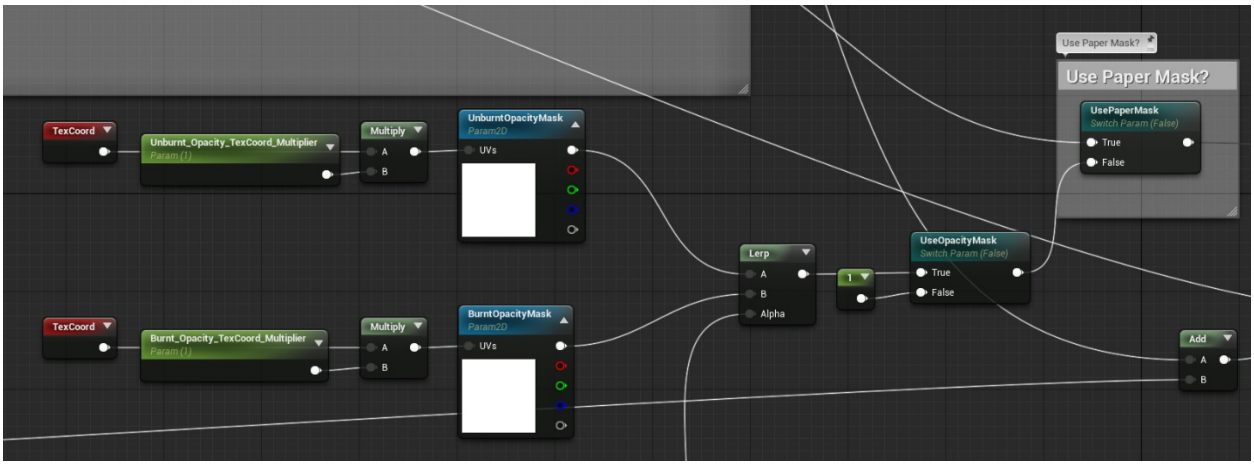
The roughness properties of a material affects how rough or smooth a surface is. The current thesis demo does not focus on the roughness and is considered for future improvement. However, the basic roughness material network allows input of textures for pre-burnt and burnt roughness states (See C11).



C11 Material expressions to accept input for roughness maps

Opacity Mask

The primary purpose of the opacity mask in the visual demo was to allow objects to expose transparency in areas that have burnt. In the thesis demo, this is used in objects of paper material. By default, the opacity masking is turned off as most objects are primarily opaque. If an object has to have opacity masking, an instance of this Master Material class will have its blending mode set to “masked.” The opacity is controlled by two texture sample parameters. By default, they have a complete white texture. White denotes opaque while black is transparent. The material instance allows the user to change the textures to create custom opacity values on a burning object. In the case of paper, there is to be complete transparency.



C12 Material Expressions for Opacity Masking

C13 Cardboard box (paper based) burning. Transparent areas show burnt areas. Darkened areas depict parts that have been heat damaged (Mid-burnt states).

Material Transitioning

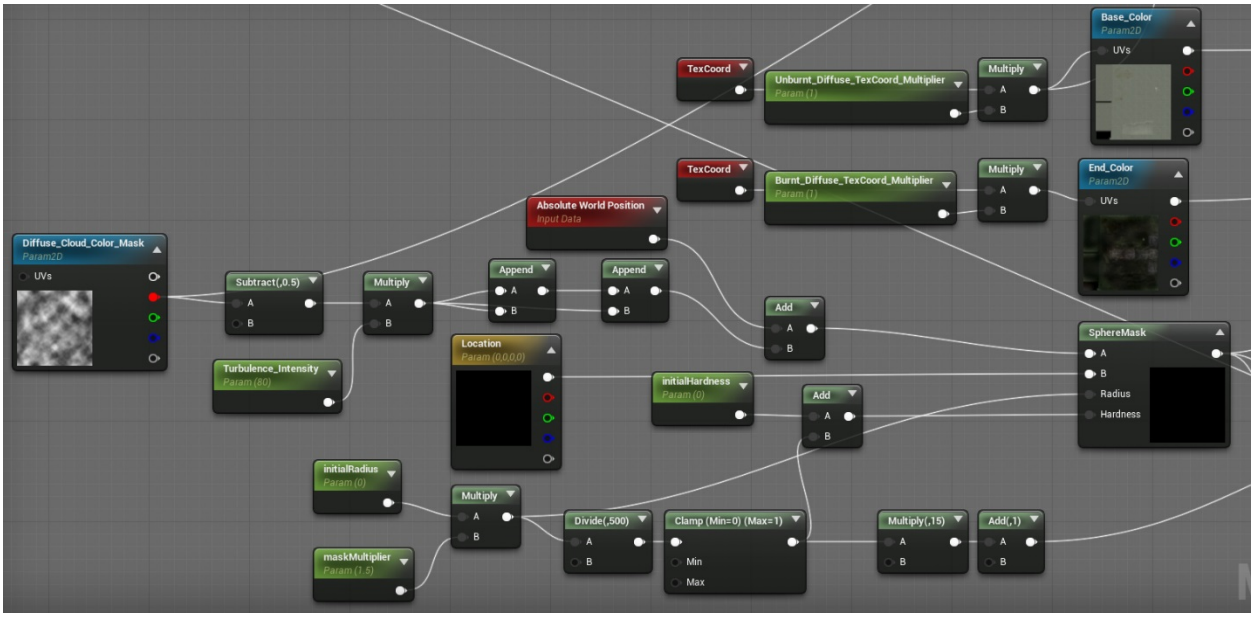
Material transition is a process in which texture A changes to texture B based on either a scalar parameter or another texture, which could be used as a pattern. The node or expression that drives the transition is the Linear Interpolation (Lerp), as explained earlier.

Heating Up

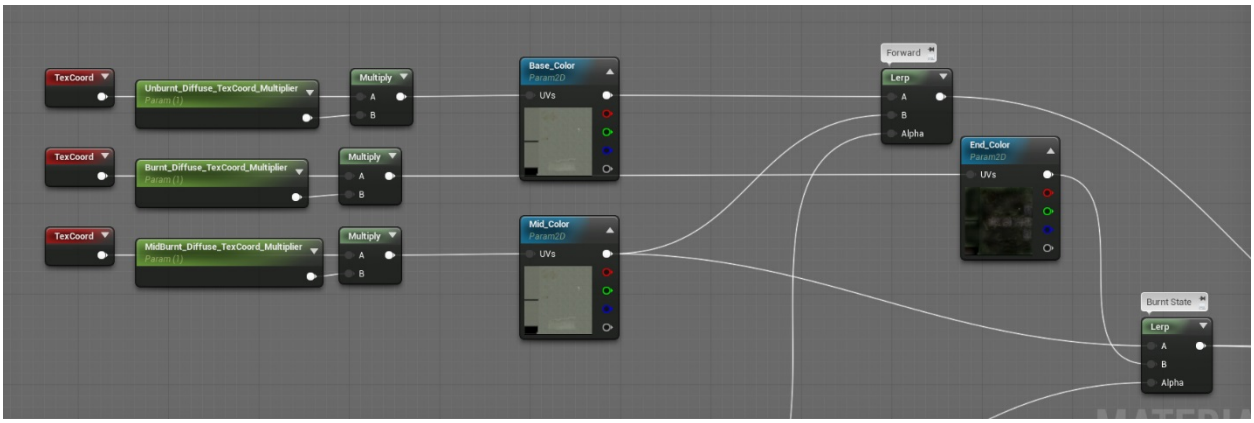
A limitation faced with the Unreal 4 engine, at the time of this thesis, is the inability to change or dynamically modify the vertex colors of a mesh in real-time. The initial proposed solution was to be able to modify vertex colors in real time to ensure the textures of a burning mesh only change where the current vertex is burning. The area in and around the vertex would be dynamically colored, and the colored region would reveal degraded textures.

This limitation was easily overcome with a material node called "Sphere Mask" (See C14) that takes in four parameters: 1) The edge point of the masking sphere, 2) location of the center of the masking sphere, 3) its radius, and 4) the hardness or fall-off region of the mask. The edge point parameter determines how a burning object's surface reveals its underlying texture. Surface edges do not burn in perfect spheres. In order to create an unevenness in the edge, a greyscale texture parameter is used as first input. This parameter accepts a texture which is basically a cloud image rendered in *Adobe Photoshop* using just black and white colors. It is then multiplied with a user editable parameter called "Turbulence" to affect the amount of distortion along the edges. High values of turbulence result in higher distortion. The difference between using and not using the greyscale texture parameter can be seen in C16. When the player clicks on an actor mesh, the point of click becomes the center of the sphere mask. Its radius is derived from the bounding sphere's radius. At each event tick, as the bounding sphere's radius is

incremented, the sphere mask's radius is updated. The sphere mask's radius is also multiplied with another user controllable parameter called ``maskMultiplier.'' This controls the rate of expansion of the sphere mask's radius. Thus, using this node, a "Location-based Masking" or "Location-based Opacity" technique was used to reveal the burnt texture under an un-burnt texture. This Location-based Masking technique was adopted from a tutorial posted on YouTube[55] and modified to fit the degradation system.

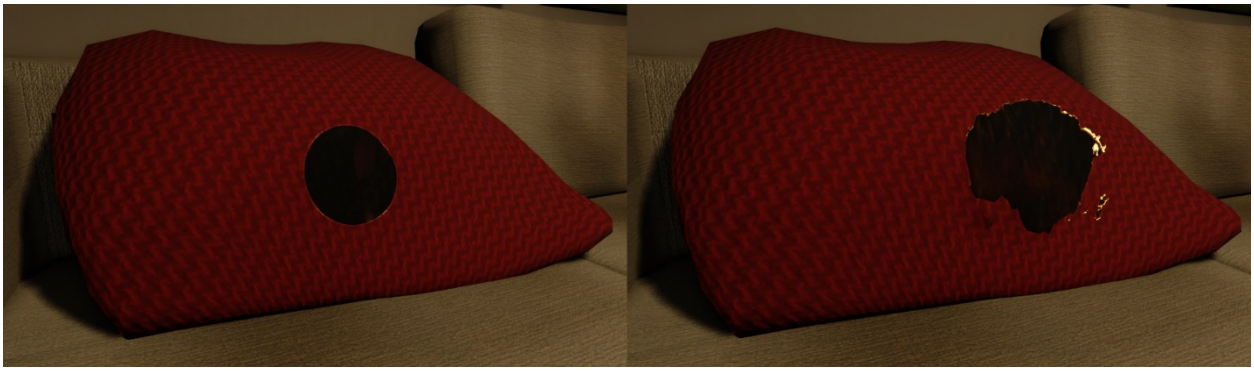


C14 Location Based Opacity Mask



C15 Pre-burnt to mid-burnt to post-burnt states

[55] "Location Based Opacity in UE4 - Part1." *YouTube*. YouTube, 3 June 2014. Web. 15 Apr. 2015.

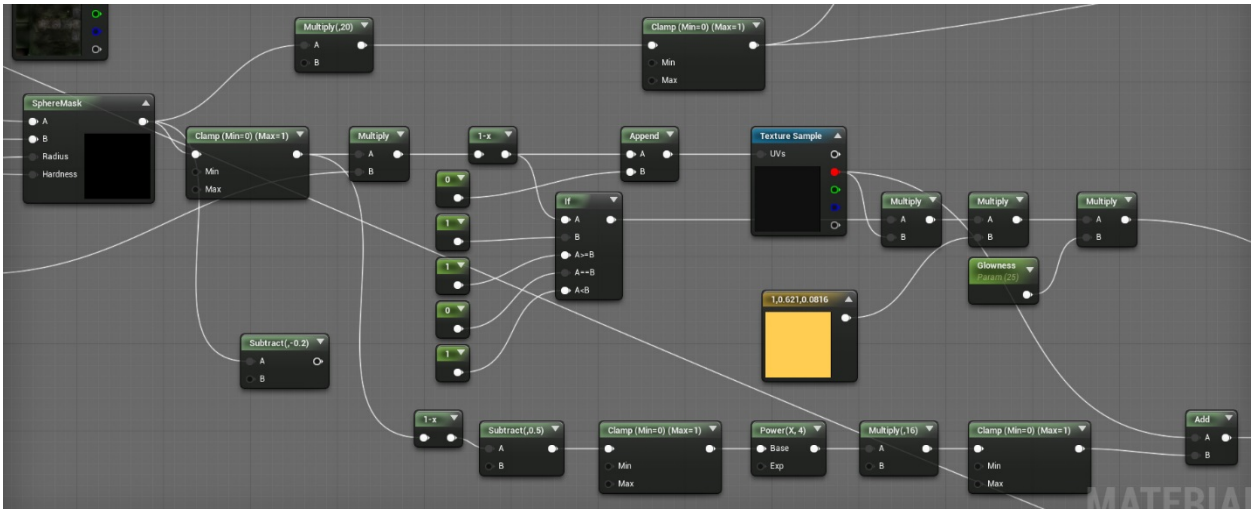


C16 (Left) without cloud edge turbulence (Right) with cloud edge turbulence

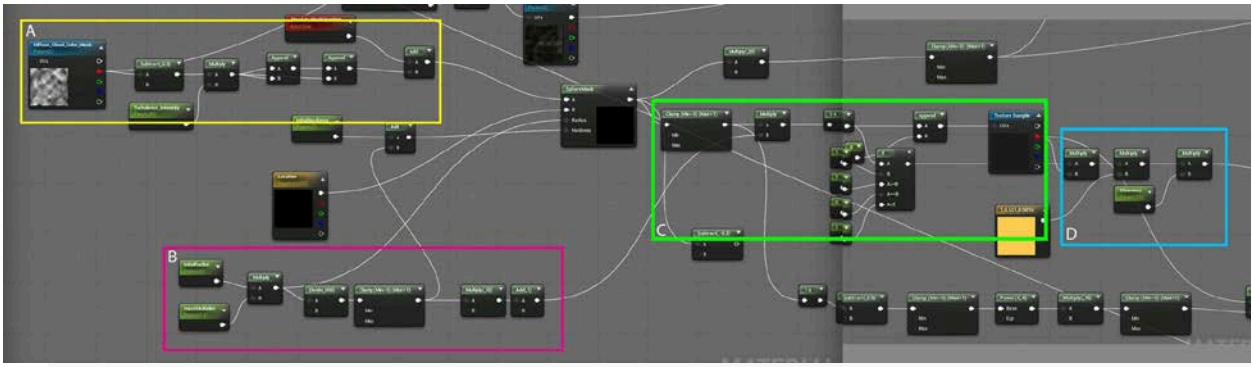
Creating the Edge Glow

The next step involved in creating transition effects was to have glowing edges. Receding edges of burning surfaces glow intensely when exposed to heat and flames. The first step in creating a glowing edge was to create a texture of size 256 pixels by 1 pixel. The texture sample for the same is shown in (See C17). It is primarily a black image with a thin white gradient strip. The white and gradient areas are multiplied with a Vector3 value (RGB) to create a yellow border. The output is multiplied with a parameter called ``Glowness'' to control the brightness. Further details on the creation of the edge glow are described in C19.

C17 Edge Glow texture sample



C18 Material Network showing creation of glowing edges as objects burn. Intensity of the glow can be controlled by the parameter called “Glowness”.



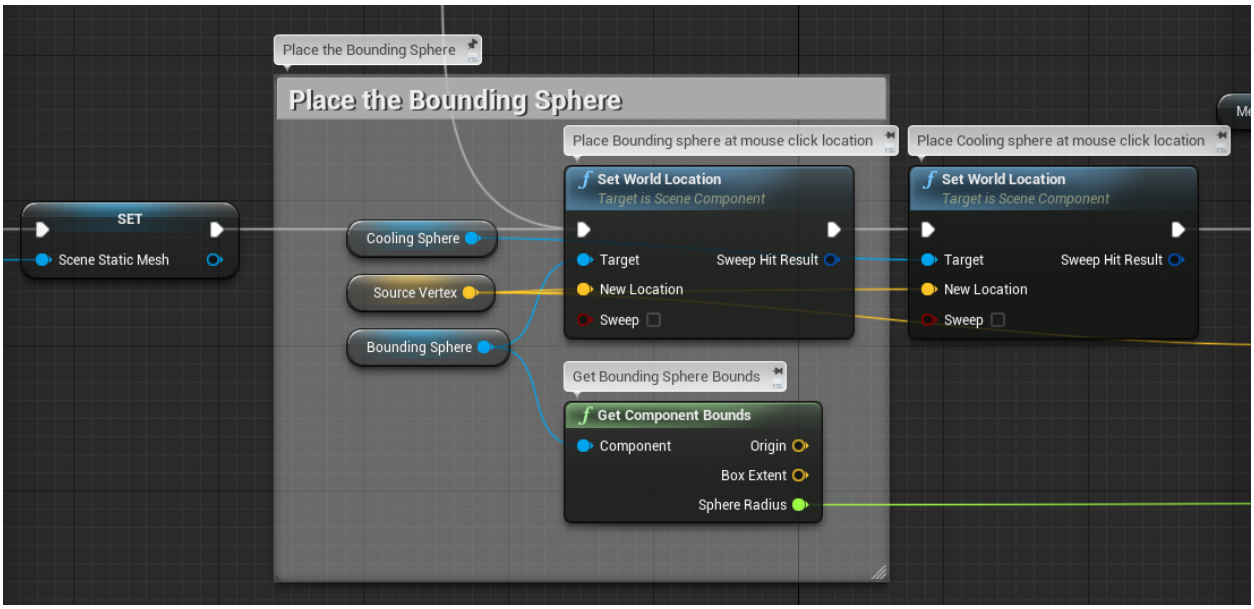
C19 **A:** A single channel output of a turbulent texture is subtracted by .5 to reduce it to a -.5 to .5 range. It is multiplied by a dynamic turbulence intensity value. Then two append nodes are used to turn this single channel into 3 channel RGB data. All of this turbulence is added into Absolute World Position to produce an output which represents a World Position distorted in all 3 axes by texture based clouds. **B:** Radius is multiplied by a maskMultiplier value to artificially shift the mask ahead of the animated radius value. This is used as the radius value in a sphere mask which also takes in the location of the burn point and the distorted World Position from section A. initialHardness is added to the clamped result of the radius divided by 500, forcing the mask to harden gradually over a 500 unit transition. **C:** The output of the spheremask is clamped and multiplied by the hardness adjustment (multiplied by 15 and added to 1) in order to shift the 0 to 1 range of the spheremask out to the outer edge of the sphere volume. The 0 to 1 range is then inverted and 0 is appended to the G channel to create an empty V channel for the texture mapping, since we only need a U channel to map a linear gradient. The IF statement exists to determine if a sample of the U channel is exactly 1, to prevent a border clipping error in the texture sample. **D:** The grayscale output of the texture sample is multiplied by the IF statement and then multiplied by an RGB color to give the glow its color. Then it's multiplied by an intensity value to give it bloom.

C20 Burnt state of demo environment

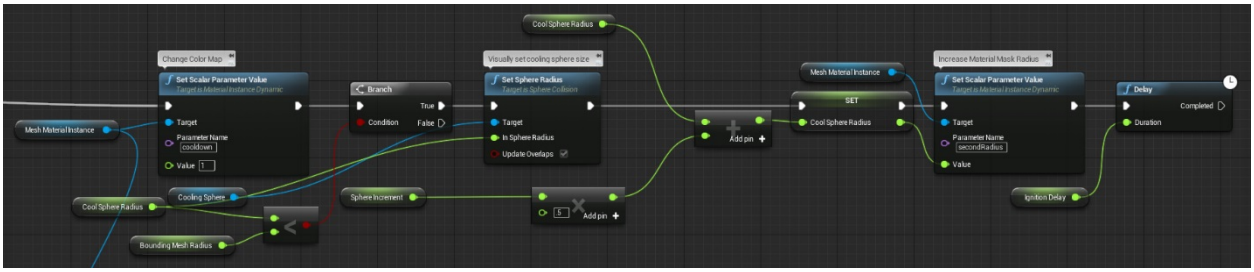
Cooling Down

The previous sub-section described the material transition involved when objects were heating up and catching fire. This sub-section describes what happens when the objects have finished burning and are cooling down. When a user clicks on a mesh, both a bounding sphere and a cooling sphere are spawned at the mouse click location (See C21). That location becomes the center of the spheres' radius. The purpose of the bounding sphere has been explained earlier. The following explains the purpose of the cooling sphere. When a mesh has more than 90%-95% of its vertices completely burnt, a scalar parameter called ``cooldown'' in the Master Material Class is set to '1'. The ``cooldown'' parameter is set to either '1' or '0', where '1' means the mesh is in cooling state and '0' is in heating state. Once set to '1', the cooling sphere begins to expand (see C22). As the sphere expands, the areas falling within its radius begin to cool down. Visually,

this is represented by the reduction in the amount of emissive color given out by that region previously burning.

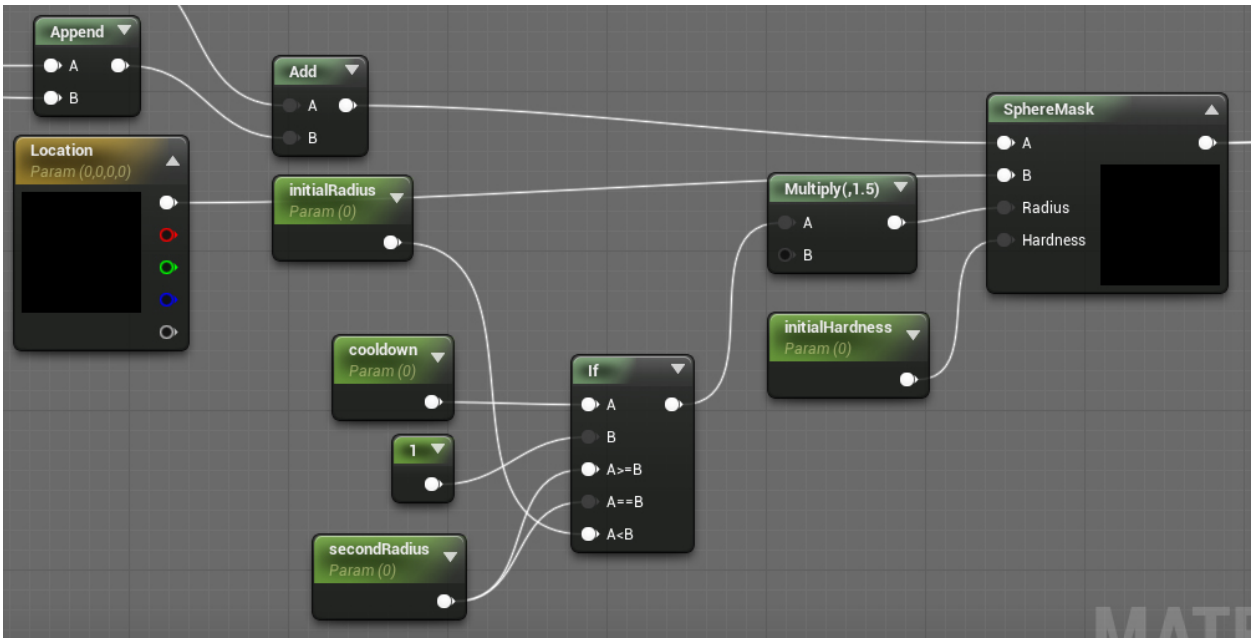


C21 Setting up the cooling sphere in blueprint



C22 Incrementing the cooling sphere

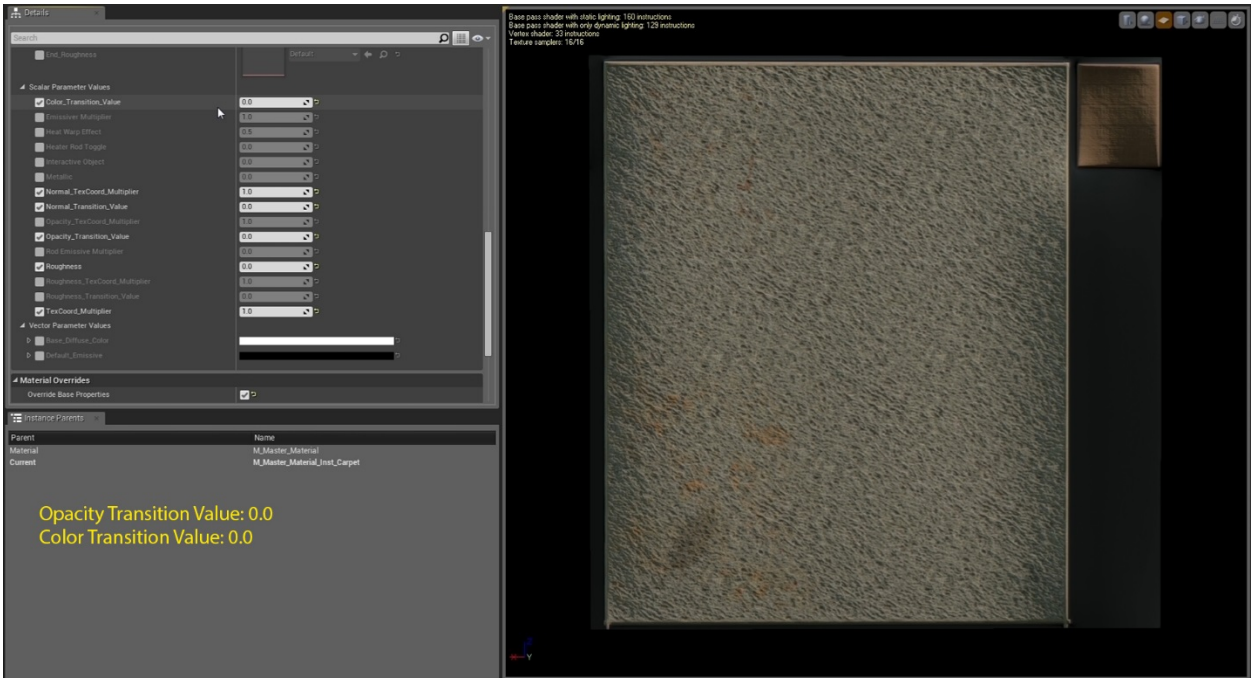
When the cooldown parameter is zero, Sphere Mask utilizes the radius of the bounding sphere that causes fire propagation. As explained earlier, during the propagation stage, the material changes to burnt states and emits red and red-orange glows. When the cooldown parameter is one, Sphere Mask utilizes the radius of the cooling sphere whose corresponding material radius is called ``secondRadius.'. As this radius increases, it reverses the emissive colors to zero slowly (see C23).



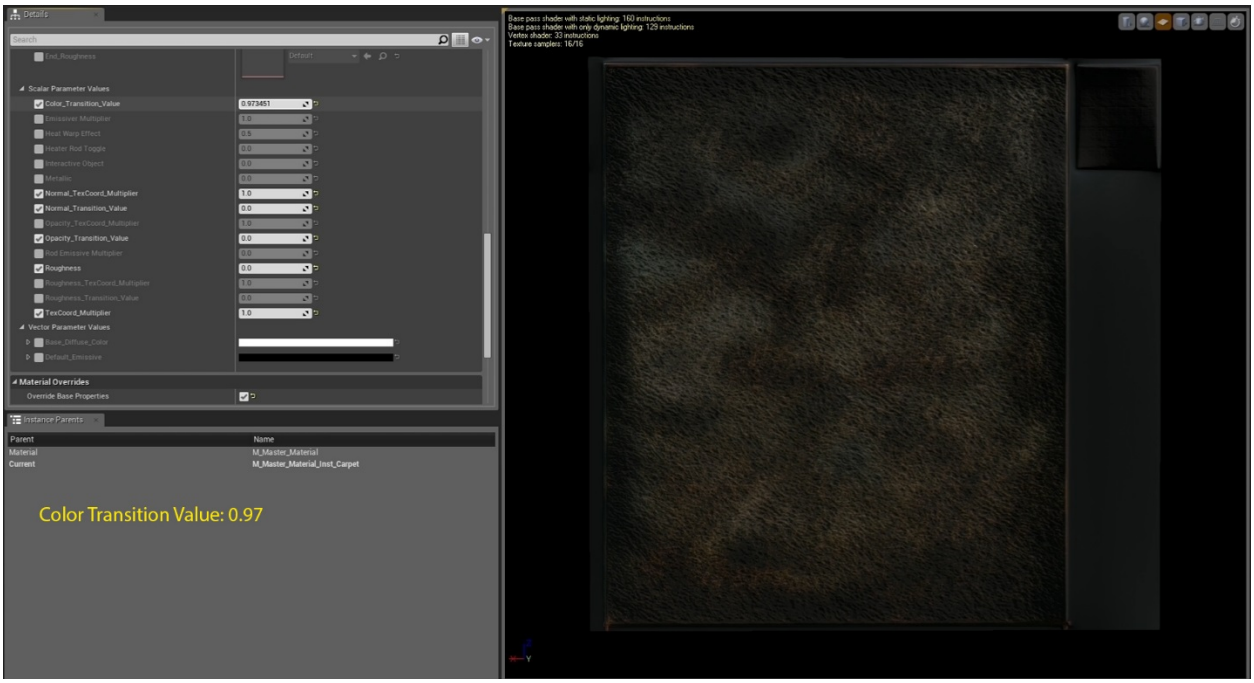
C23 Conditional statement for ``cooldown''

Procedural Transitioning

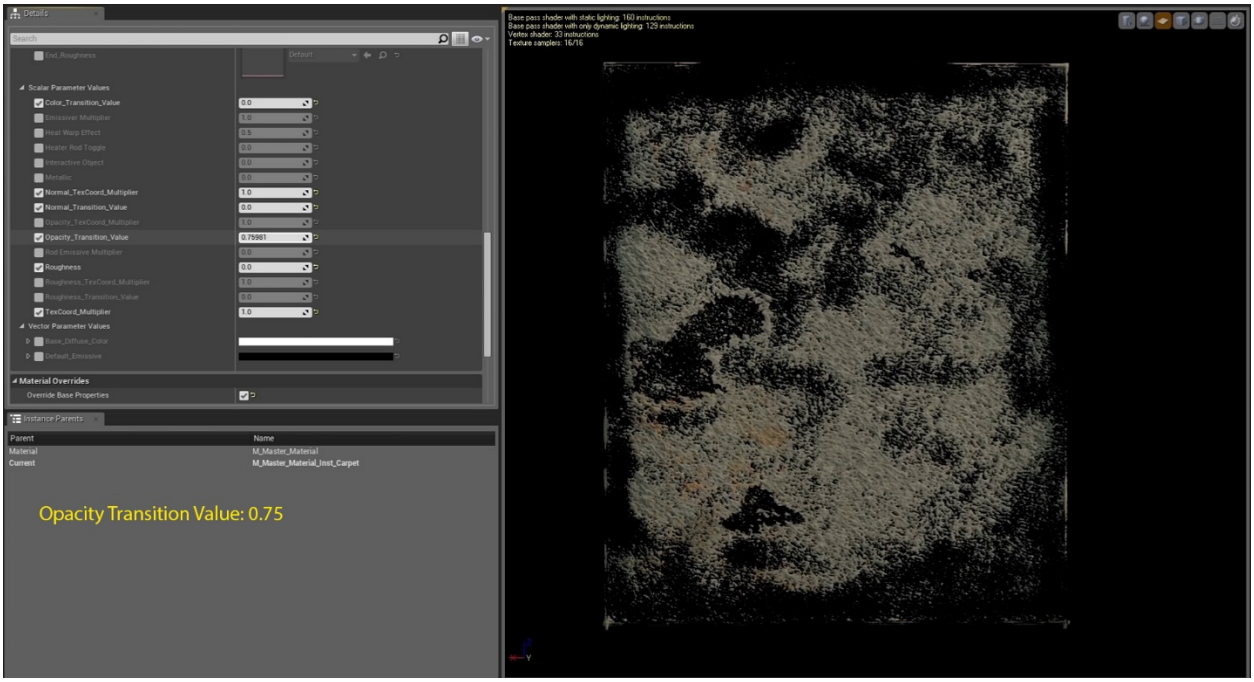
In order to ensure each interactive object's material in the scene is uniquely affected, the Master Material Class is instanced and assigned to each object. This allows procedural transitioning of the material from a pre-burnt state to a burnt state. Since the Master Material Class is integrated with the propagation system, the instanced material's parameters can now be modified either at compile time by the designer or at run-time by the system. The following figures show how as the material parameters are adjusted they affect the object's surface in the scene.



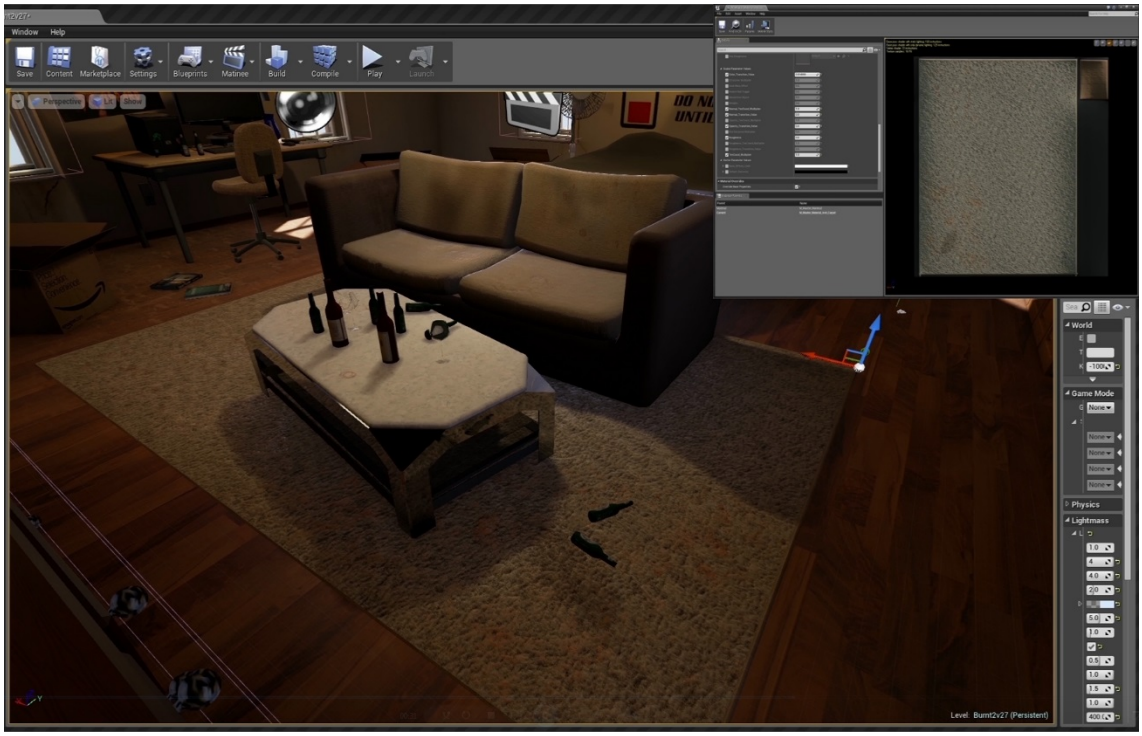
C24 Procedural Controls



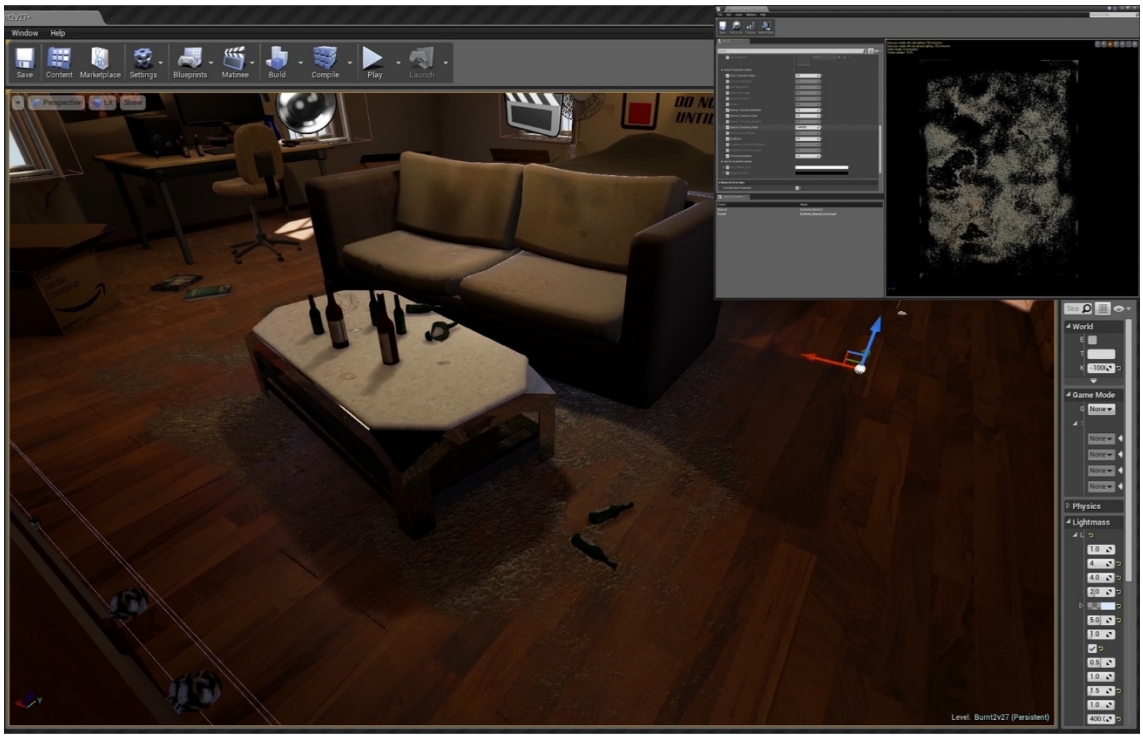
C25 Effect on material when base color value is set to 0.97.



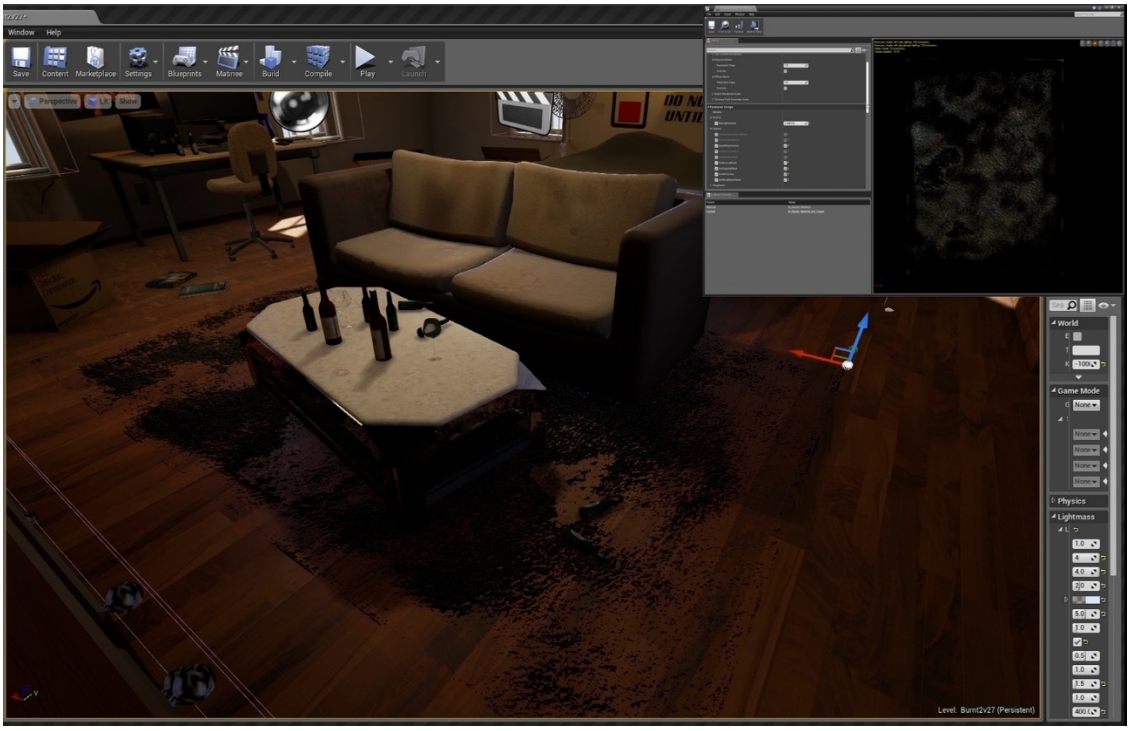
C26 Effect on material when opacity transition value is set to 0.75



C27 Integration of Master Material Class with Procedural Controls and Blueprint



C28 Color and Opacity Transition controlled procedurally



C29 Complete burning and charring of carpet material. The values are controlled and visual output is achieved procedurally.

Works Cited

14-pointed Silver Star set into the marble floor and surrounded by silver lamps. Digital image. *Pilgrimage in the Holy Land*. N.p., n.d. Web. 08 Jan. 2015. <http://wigowsky.com/travels/israel/book/ch9.htm>.

Alone in the Dark® (2008). Computer software. *Atari*. Atari, 20 June 2008. Web. Spring 2014. <https://www.atari.com/buy-games/adventure/alone-dark>.

Alone in the Dark - Fire. Digital image. *Eurogamer.net*. N.p., 18 Apr. 2008. Web. 03 Nov. 2014. <http://www.eurogamer.net/videos/alone-in-the-dark_fire>.

Audet, Matt. "Heat Distortion from Flames." *Blender Guru How to Create Heat Distortion Comments*. Blender Guru, n.d. Web. 16 Mar. 2015. <http://www.blenderguru.com/tutorials/how-to-create-heat-distortion/>.

"Burning Chemicals!" *YouTube*. YouTube, 10 Nov. 2009. Web. 18 Mar. 2015. <https://www.youtube.com/watch?v=li2P2I3tCxw>.

Burning object can interact with environment. Digital image. *TechArena Community RSS*. N.p., 27 June 2008. Web. 03 Nov. 2014. <http://forums.techarena.in/reviews/992949.htm>.

Burned and melted Plastic. Digital image. *Pinterest*. N.p., n.d. Web. 27 Feb. 2015. <https://www.pinterest.com/pin/121667627405730393/>.

Burning Rubber. Digital image. *Envato Market*. N.p., 24 Apr. 2013. Web. 10 Feb. 2015. <http://videohive.net/item/burning-rubber/4565657>.

Burn on Wooden Door Stock Photo 54838856 - IStock. Digital image. *IStock by Getty Images*. Getty Images, 03 Jan. 2015. Web. 15 Mar. 2015. <http://www.istockphoto.com/photo/burn-on-wooden-door-54838856?st=5325daa>.

Cafe, Tony. "Physical Constant for Investigators." 17 June 2007. *T.C. Forensic, Forensic & Scientific Services.* Article. 12 April 2015. <http://www.tcforensic.com.au/docs/article10.html>.

Channell, Justin. *Burnt Couch*. Digital image. *Flickr*. Yahoo!, 10 Mar. 2010. Web. 08 Feb. 2015. <https://www.flickr.com/photos/justinchannell/4472969643>.

Content Examples. Rockville: Epic Games, Fall 2014. Unreal 4 Engine Content Examples Package.

Crops burning in Far Cry 3. Digital image. *Giant Bomb*. Giant Bomb, n.d. Web. 04 Oct. 2014. <http://www.giantbomb.com/far-cry-3/3030-32933/forums/why-is-there-a-skrillex-song-in-this-game-569238/>.

Datta, Arko. *In Pictures: Gujarat's 'Houses of Death'* Digital image. *BBC News*. N.p., 29 Feb. 2012. Web. 12 Jan. 2015. <http://www.bbc.com/news/in-pictures-17184815>. Image 5 of 8

Dripping Fire. *YouTube*. YouTube, 15 Sept. 2007. Web. 15 Apr. 2015. <https://www.youtube.com/watch?v=yypzkz5OzB4>.

Edwards, Jason. *A Plastic Water Tank Melted in the Extreme Heat of a Forest Fire.* Digital image. *Getty Images*. Getty Images, n.d. Web. Spring 2015. <http://www.gettyimages.com/detail/photo/plastic-water-tank-melted-in-the-extreme-heat-of-a-royalty-free-image/92633554>.

Epic Games. *Unreal Engine 4 Documentation*. 2014. Documentation. 11 Feb 2015. <https://docs.unrealengine.com/latest/INT/Engine/Content/FBX/MorphTargets/index.html>.

Far Cry 3. Computer software. *Far Cry 3*. Ubisoft, 29 Nov. 2012. Web. Spring 2014. <http://far-cry.ubi.com/fc-portal/en-gb/home/>.

Far Cry 3 Screenshots for Windows - MobyGames. Digital image. *Moby Games*. Moby Games, 30 Dec. 2012. Web. 04 May 2014. <http://www.mobygames.com/game/windows/far-cry-3/screenshots/gameShotId,598316/>.

Free Photoshop Patterns and Textures of Wood and Metal. Digital image. *SitePoint*. N.p., 20 July 2012. Web. 07 Apr. 2015. <http://www.sitepoint.com/30-free-wood-and-metal-photoshop-patterns-and-textures/>.

French Goodyear Workers Burn Tyres in Riot against Bosses | The Times. Digital image. *The Times*. N.p., 8 Mar. 2013. Web. 3 Feb. 2015. <http://www.thetimes.co.uk/tto/news/world/europe/article3708226.ece>.

Gorbett, Gregory E., and James L. Pharr. *Fire Dynamics*. Upper Saddle River, NJ: Pearson, 2011. Print.

Graf, S.H. "Ignition Temperatures of Various Papers, Woods and Fabrics." March 1949, Bulletin No. ed.: 7.

Heat Distortion from Jet Engine. Digital image. *Foundation 3D*. N.p., June 2006. Web. 19 Jan. 2015. <http://www.foundation3d.com/forums/showthread.php?t=3262>.

Hooper, Fred. *VFX Fire Pack*. Rockville: Epic Games, 22 Apr. 2015. Unreal 4 Engine Particle Systems Package.

Jerusalem - Police Arrest Suspects In Mosque Arson Attack. Digital image. N.p., 3 Oct. 2011. Web. 12 Mar. 2015. <http://www.vosizneias.com/92422/2011/10/03/jerusalem-police-arrest-suspects-in-mosque-arson-attack/>.

Laylin, Tafline. Tyre on Fire. Digital image. *Green Prophet*. N.p., 23 May 2011. Web. 06 Nov. 2014. <http://www.greenprophet.com/2011/05/recycling-tires-mosquitoes-rats/>.

Levesque, JeanFrancois. "Far Cry: How the Fire Burns and Spreads." N.p., 6 Dec. 2012. Web. Spring 2014. <http://jflevesque.com/2012/12/06/far-cry-how-the-fire-burns-and-spreads/>.

"Location Based Opacity in UE4 - Part1." *YouTube*. YouTube, 3 June 2014. Web. 15 Apr. 2015. <https://www.youtube.com/watch?v=XYPPd5oLoPM>.

"Marble: Characteristics, Uses and Problems." *U.S. General Services Administration*. U.S. General Services Administration, 24 Feb. 2012. Web. 05 Mar. 2015. <http://www.gsa.gov/portal/content/111858>.

Modise, Refilwe. “A Young Man Drags a Burning Tyres during a Service Delivery Protest in the Boiketlong Informal Settlement.” Digital image. *The Citizen*. N.p., 15 Oct. 2014. Web. 06 Mar. 2015. <http://citizen.co.za/258211/sebokeng-residents-protest-service-delivery/>.

Objects on Fire in the video game Alone in the Dark. Digital image. *Giant Bomb Fire Propagation Galleries*. N.p., 26 July 2008. Web. 5 May 2015. <http://www.giantbomb.com/fire-propagation/3015-342/images/>.

"Old Burnt Door Texture." *Texture Library*. Texture Library, n.d. Web. 07 Mar. 2015. <http://texturelib.com/texture/?path=%2FTextures%2Fdoors%2Fwood+doors%2Fdoors_wood_doors_0088>.

Plantation on fire. Digital image. *Far Cry 3 Walkthrough [GUIDE]*. CheatMasters Blog, 11 Dec. 2012. Web. Spring 2014. <http://www.cheatmasters.com/blog/2012/12/11/far-cry-3-walkthrough-guide/>.

Rama. "Rama's Vertex Snap Editor Plugin." *Epic Wiki*. Epic Games, 2014. Web. Fall 2014. <https://wiki.unrealengine.com/Rama's_Vertex_Snap_Editor_Plugin>.

Rizzo, Francesca. *Burnt Plastic*. Digital image. *Red Bubble*. N.p., n.d. Web. 27 Feb. 2015. <http://www.redbubble.com/people/frantab/works/2263772-burnt-plastic>.

Rusch, M.D. "How Different Fabrics and Materials Burn Cotton, Leather Ect." *YouTube*. YouTube, 1 June 2013. Web. Spring 2014. <https://www.youtube.com/watch?v=Vio4kleGxhM>.

Ryse: Son of Rome. Computer software. *Ryse®: Son of Rome*. Crytek, 22 Nov. 2013. Web. Spring 2014. <http://www.crytek.com/games/ryse/overview>.

Ryse Son of Rome Gameplay Walkthrough Part 1 - The Beginning (XBOX ONE). *YouTube*. YouTube, 21 Nov. 2013. Web. 12 May 2014. <https://www.youtube.com/watch?v=xmV4CdNdx0k>.

Smith, Andrew. *Burnt Cardboard*. Digital image. *Deviant Art*. N.p., n.d. Web. 15 Feb. 2015. <http://stock7000.deviantart.com/art/Burnt-Cardboard-123806645>.

Stroup, David W., and Daniel Madrzykowski. "Section 2: Flammability Hazard of Materials." *Fire Protection Handbook, Volume 1*. N.p.: National Fire Protection Associates, n.d. 2-31--46. *Fire Protection Handbook Volume 1*. Web. Spring 2014. <http://fire.nist.gov/bfrlpubs/fire08/PDF/f08006.pdf>.

"Textile Fibers Burning Test." *YouTube*. YouTube, 16 Oct. 2010. Web. 07 Apr. 2015. <https://www.youtube.com/watch?v=kb4tCcnA6jo>.

The Tyres Emitting Hazardous Smoke at New Baneswhor. Digital image. *Moth Shut Up*. N.p., 22 May 2008. Web. 14 Apr. 2015. <https://mothshutup.wordpress.com/2008/05/22/never-ending-atrocities/>.

Uncharted 3: Drake's Deception - Chateau Pt 2: Burning Up Gameplay Movie (PS3). Digital image. *YouTube*. YouTube, 6 Oct. 2011. Web. Fall 2014. <https://www.youtube.com/watch?v=Hy6InG5WUrE>.

Uncharted 3: Drake's Deception. Computer software. *Naughty Dog: Uncharted Drake's Deception*. Sony Computer Entertainment, 1 Nov. 2011. Web. Spring 2014. <http://www.naughtydog.com/games/uncharted3_drakes_deception/>.

Unreal Engine 4. Computer software. *What Is Unreal Engine 4*. Vers. 4.8. Epic Games, 2014. Web. Fall 2014. <https://www.unrealengine.com/what-is-unreal-engine-4>.

Urbex: Abandoned, Burned, Semi-Demolished Emge Foods Meat Processing Plant. Digital image. *Love These Pics*. N.p., 16 June 2012. Web. 13 Feb. 2015. <http://www.lovethesepics.com/2012/06/urbex-abandoned-burned-semi-demolished-emge-foods-meat-processing-plant-72-pics/>.

VideoHive Fire And Sparks. Digital image. *Hyperlino*. Video Hive, 15 May 2014. Web. 20 Jan. 2015. <http://www.hyperlino.com/stock-footage/84264-videohive-fire-and-sparks-7718052.html>.

Wooden Branches on Fire. Digital image. *Wood Perfected*. N.p., 19 May 2011. Web. 09 Apr. 2015. <https://woodperfected.wordpress.com/tag/wood-panel-industries-federation/>.

Zavos, Alison. Spectral emissions from different chemicals burning: (left to right) Methane, Calcium Sulfate, Calcium phosphate, Sodium chloride, Potassiam phosphate, Sodium borate. Digital image. *Feature Shoot*. N.p., 29 Nov. 2010. Web. 08 Mar. 2015. <http://www.featureshoot.com/2010/11/qa-ryan-matthew-smith-seattle/>.