A MATERIALS SCIENCE DRIVEN PATTERN GENERATION SOLUTION TO FRACTURING COMPUTER GENERATED GLASS FOR FILMS AND GAMES

A Thesis

by

DAVID CHARLES MONROE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Philip Galanter
Committee Members,	Ergun Akleman
	John Keyser
Head of Department,	Tim McLaughlin

August 2014

Major Subject: Visualization

ABSTRACT

Believably and realistically fracturing computer generated glass for visual effects has been previously solved through various methods such as algorithmic approaches, utilizing texture maps, or finite element analysis. These solutions can achieve some believable results but often at the cost of one or more of the following: simulation time, preparation time, art directability, consistency with materials science research, or the requirement of creating or utilizing fixed assets or maps. In this thesis I present a novel method that draws from the appropriate literature and focuses on quickly generating accurate fracture patterns. The method takes inputs such as the artist's animation of an impact and desired object properties, and outputs fracture patterns used for breaking objects apart based on input values, materials science literature, and fracture mechanics. After determining all of the fracture pattern variables such as the number of radial and concentric cracks, the artist is able to override the computed parameters to retain control and art directability. Implementation of this method was performed using MAXScript, the built-in scripting language for Autodesk 3ds Max. The result is a computationally fast and mechanically accurate tool while retaining art directability to fulfill film storyboards or game design.

NOMENCLATURE

v	Velocity
V	Volume
M	Mass
F	Force
h	Thickness
c	Speed of Sound in the Material
E	Young’s Modulus
B	Bulk Modulus
ρ	Density
ν	Poisson’s Ratio
γ	Surface Tension
σ	Stress
ε	Strain
2D	Two-Dimensional
3D	Three-Dimensional
CG	Computer Graphics
GUI	Graphical User Interface
VFX	Visual Effects
SFX	Special Effects
FEA	Finite Element Analysis
FEM	Finite Element Method

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

I. INTRODUCTION

This thesis targets the problem of believably shattering and simulating virtual objects defined as glass due to collision impacts. In many films and games, there is often a need to simulate glass fracturing for scenarios such as a character's body, projectile, or other moving objects causing the glass to shatter. Some films and games depict battle scenes in large cities where flat window glass is ubiquitous from street level windows to entire paned walls of skyscrapers, and rely on visual effects to allow destruction of the environment. These objects and environments are typically created by polygonal modeling using 3D CG software packages, and then are rendered with other layers or composited into the corresponding live action scene. In these packages, an artist can easily use a rectangular prism with a glass material to create a virtual window pane. This model's only definitions are the eight corner vertices and six polygonal faces. In the real world, a pane of glass is comprised of an immense number of atoms, giving it continuous surface and interior definition. When glass in the real world is subjected to a strong force, the bonds between its atoms are separated, cracks are initiated and propagated, and the glass shatters. Given the practical limits in computing power to represent, simulate, or otherwise recreate this behavior, there is no low-level way to fracture a virtual polygonal object. This thesis provides a system to simulate breaking glass by taking artist inputs, finding collision points, generating realistic fracture patterns, and using these patterns to break the polygonal objects into pieces for simulation or animation.

II. BACKGROUND AND RELATED WORK

This chapter discusses the related work that applies to this thesis. Section II.1 discusses the physical properties of glass, while section II.2 discusses the fracture mechanics and fractography of glass. The physical properties and fracture mechanics sections are later be referenced for explanation of this thesis's methodology. Lastly, section II.3 discusses related visual effects approaches to the problem of recreating the phenomenon of glass fracture.

II.1. Materials Science and Physical Properties of Glass

This section explores the background research for the materials science and physical properties of glass. Materials scientists examine the properties of matter and the relationship between a material's microscopic atomic structure and the material's macroscopic properties. This thesis utilizes this knowledge to ensure accuracy and believability when recreating this phenomenon in the CG virtual world. As this section discusses the physical properties of glass, references are made to how these properties are be utilized in the methodology chapter.

II.1.1. Chemical Composition and Molecular Structure

Materials are separated into categories based on their atomic structure such as metals, polymers, composites, and ceramics and glasses. In this research the focus is on ceramic and glass. Generally, glass is defined as a hard brittle substance that is transparent or translucent and is made by fusing sand with soda, lime, and other ingredients. Some scientists regard some organic substances as glass because they can be supercooled to an extent that they become rigid solids without visible crystallization. For this thesis, only inorganic glass as defined by the American Society for Testing and Materials (ASTM) are discussed. Their definition of glass is “An inorganic product of fusion which has cooled to a rigid condition without crystallization” [37].

The most common type of glass is soda-lime-silica glass, and it is used in most windowpanes and glass containers. The flat soda-lime-silica glass sheets produced and used for windows is composed of about 72% silicon dioxide (SiO_2, “silica”), 14% sodium oxide (Na_2O, “soda”), 10% calcium oxide (CaO, “lime”), and 4% other minor ingredients. Other types of silicate glasses include fused silica, sodium borosilicate, and lead-oxide. Different creation methods, percentages of chemicals used, and strengthening or tempering after creation all affect how the glass fractures when subjected to various forces.

The atomic structure of soda-lime-silica glass a matrix of non-crystalline silica with sodium and calcium ions spread throughout. Glass is an amorphous solid, meaning it lacks long-range order in which the molecules are arranged in a regular and periodic manner. Figure 1 shows, from left to right, a representation of an amorphous solid, crystalline solid, and the structure of glass.

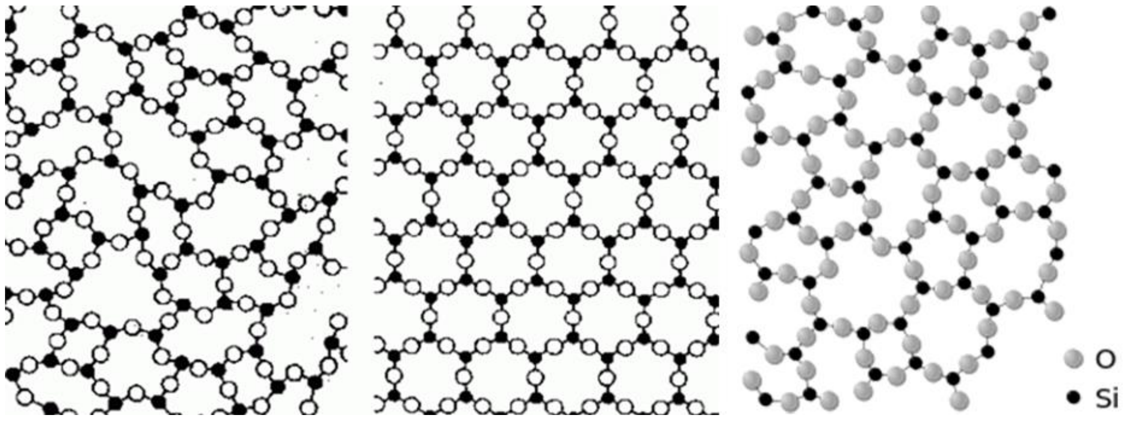


Figure 1: Amorphous, Crystalline, and Glass Structure [44]

This thesis focuses on flat soda-lime-silica glass fractures as opposed to other types of glass, as the breaking of atomic bonds propagates differently depending upon the atomic structure of a material as discussed further in II.2.

II.1.2. Mass, Density, and Volume

The density (ρ) of soda-lime-silica glass is about 2.5 grams per cubic centimeter (g/cm^3). Other various glasses range from 2.4 to 8.0 g/cm^3. The volume of a sheet of glass that is approximately a rectangular prism can be estimated by multiplying together the glass's length, width, and height. Many 3D CG software packages, such as Autodesk's 3ds Max, can use a real world unit scale and calculate the volume of a polygonal mesh by tetrahedralizing a copy of the mesh and adding up all of the tetrahedral volumes. Mass of the virtual glass can then be derived by multiplying the artist-defined density of the glass with the calculated volume of the polygonal model.

II.1.3. Brittleness, Ductility, and Stiffness

Stress is the measure of applied force while strain is the measure of deformation the object undergoes due to that force. If an object absorbs little energy and deforms a small amount prior to fracture, it is classified as brittle. Glass, ceramic, and cast iron are examples of materials classified as brittle. If an object absorbs energy and deforms a large amount prior to fracture, it is classified as ductile. Gold, copper, and clay are examples of materials classified as ductile. Figure 2 shows a graph of stress versus strain that illustrates how much energy is absorbed before fracture occurs in the object.

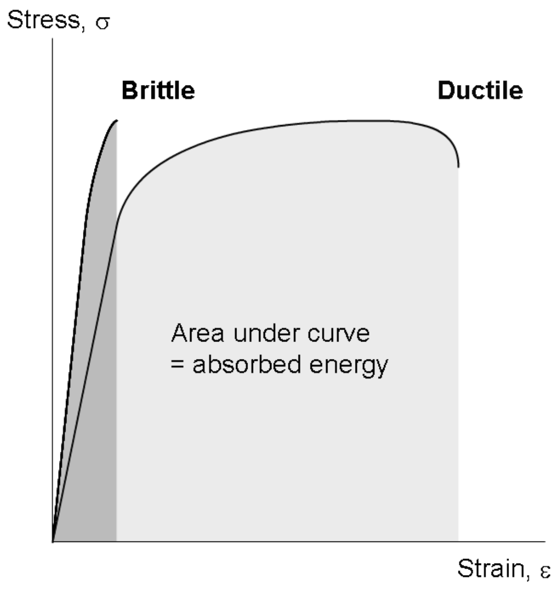


Figure 2: Stress vs. Strain for Brittle and Ductile Materials [43]

Other brittle materials include some plastics like polymethylmethacrylate (PMMA), laminated, toughened glasses, safety glasses, other ceramics, most non-metals, and some metals when subjected to low temperatures. Brittleness, ductility, malleability, plasticity, stiffness, and temperature are all properties that affect how a material reacts to stress. Figures 3 and 4 show how temperature can greatly change glass's stiffness or strength. Heat strengthening, or tempering, can increase the strength of glass by a factor of five for most glass objects [7]. This thesis assumes that the glasses being simulated for fracture are not under extreme temperatures.

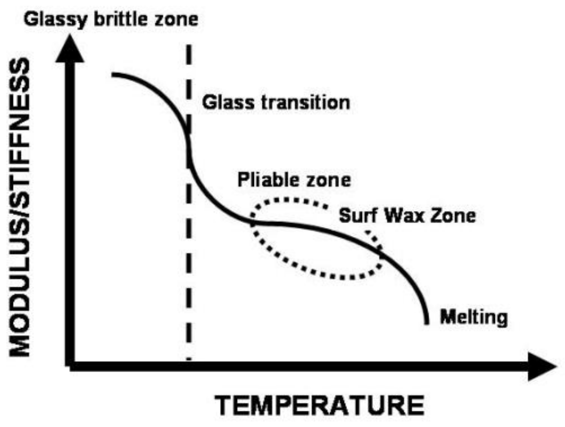


Figure 3: Stiffness vs. Temperature for Glass Brittleness [40]

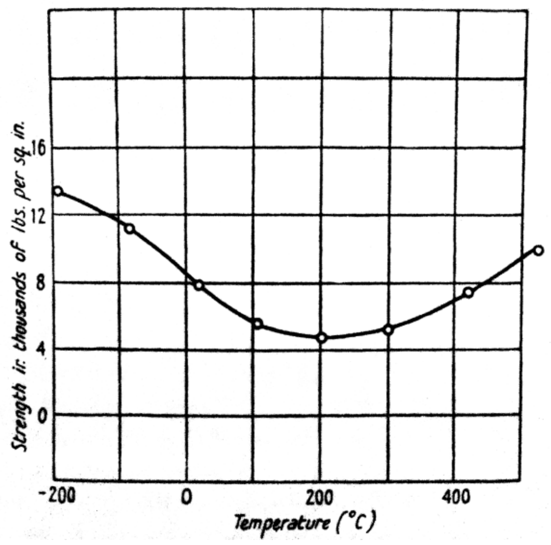


Figure 4: Strength vs. Temperature for Glass Brittleness [36]

Young's modulus (*E),* also called tensile or elastic modulus, is a measure of the stiffness of a material. It is the ratio of the strain along an axis compared to the stress along that axis. A material such as steel or diamond has a high Young's modulus. A material such as rubber or foam with a low Young’s modulus is very deformable. Figure 5 displays common ranges of Young’s modulus and density.

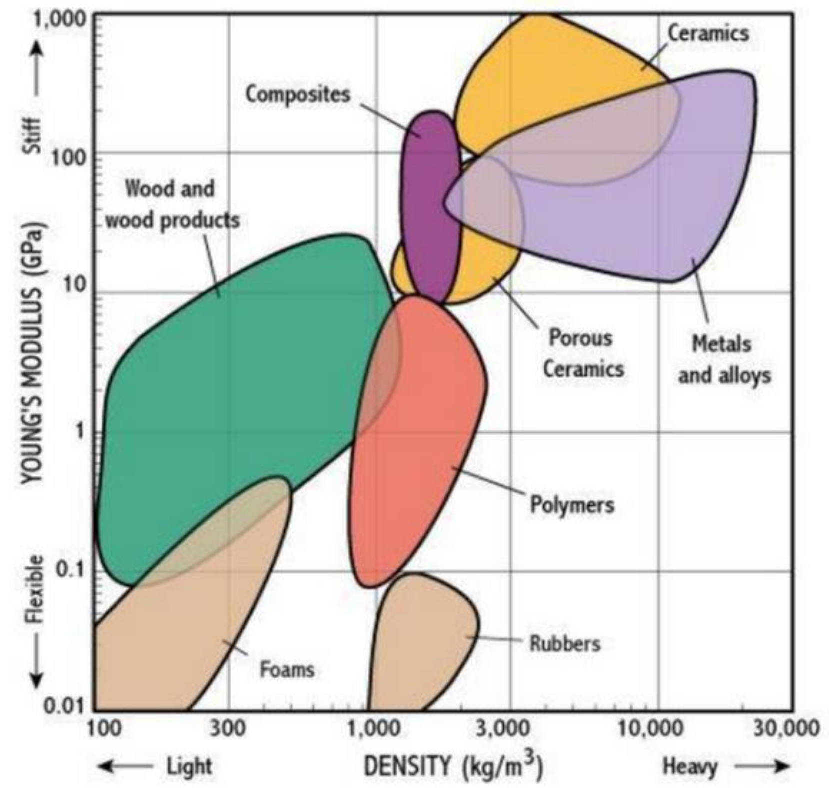
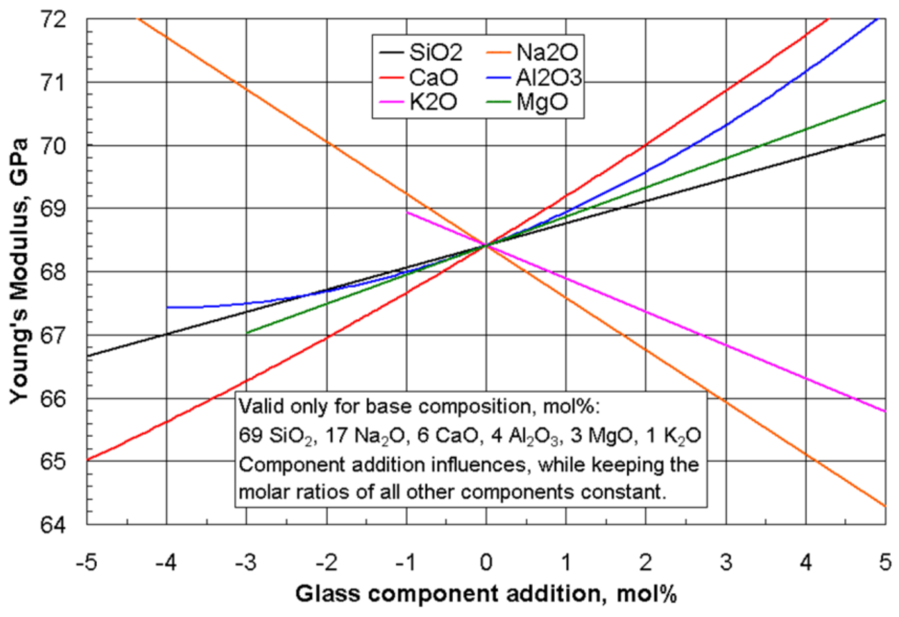


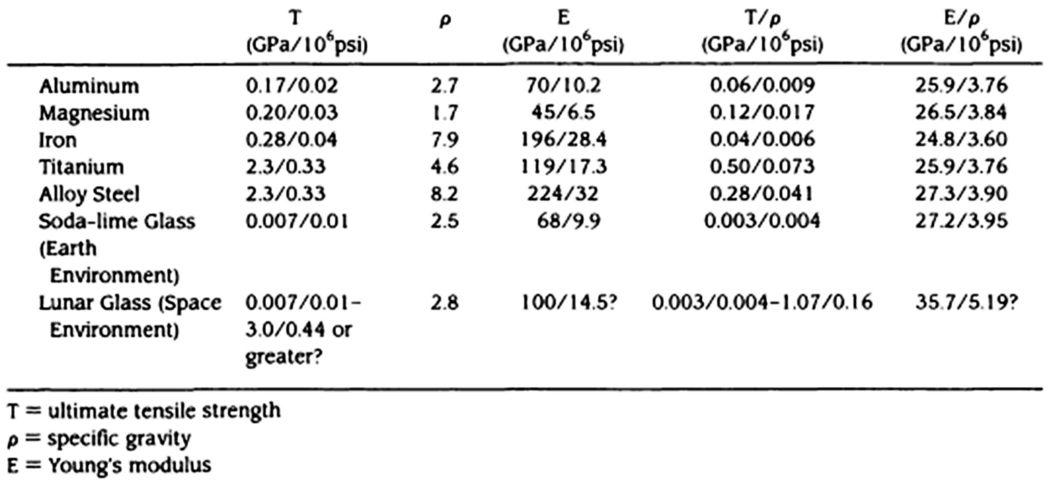
Figure 5: Young’s Modulus vs. Density for Various Materials [9]

Young’s modulus is measured in Pascals, a measure of pressure, and is calculated by dividing the stress by the strain. As stated earlier, the chemical composition of glass influences the glass’s stiffness. Typical values of density and Young’s modulus for some common materials are shown in Figures 6 and 7.



Valid only for base composition, mol%:
69 SiO_2, 17 Na_2O, 6 CaO, 4 Al_2O_3, 3 MgO, 1 K_2O
Component addition influences, while keeping the molar ratios of all other components constant.

Figure 6: Young's Modulus vs. Glass Chemical Composition [13]



	T (GPa/10^6psi)	ρ	E (GPa/10^6psi)	T/ρ (GPa/10^6psi)	E/ρ (GPa/10^6psi)
Aluminum	0.17/0.02	2.7	70/10.2	0.06/0.009	25.9/3.76
Magnesium	0.20/0.03	1.7	45/6.5	0.12/0.017	26.5/3.84
Iron	0.28/0.04	7.9	196/28.4	0.04/0.006	24.8/3.60
Titanium	2.3/0.33	4.6	119/17.3	0.50/0.073	25.9/3.76
Alloy Steel	2.3/0.33	8.2	224/32	0.28/0.041	27.3/3.90
Soda-lime Glass (Earth Environment)	0.007/0.01	2.5	68/9.9	0.003/0.004	27.2/3.95
Lunar Glass (Space Environment)	0.007/0.01–3.0/0.44 or greater?	2.8	100/14.5?	0.003/0.004–1.07/0.16	35.7/5.19?

T = ultimate tensile strength
ρ = specific gravity
E = Young's modulus

Figure 7: Tensile Strength, Density, and Young's Modulus of Various Materials [4]

When a material is compressed, it may expand perpendicularly to the direction of force. This phenomenon is called the Poisson effect, named after Siméon Poisson, where

Poisson's ratio is the negative ratio of transverse to axial strain. A material that expands a great deal laterally when compressed, like rubber, has a ratio of around 0.5, while a material that does not, like cork, has a ratio of around 0.0. Glass has a ratio of around 0.18 to 0.3, again, depending on the specific chemical composition of the glass, as shown in Figure 8.

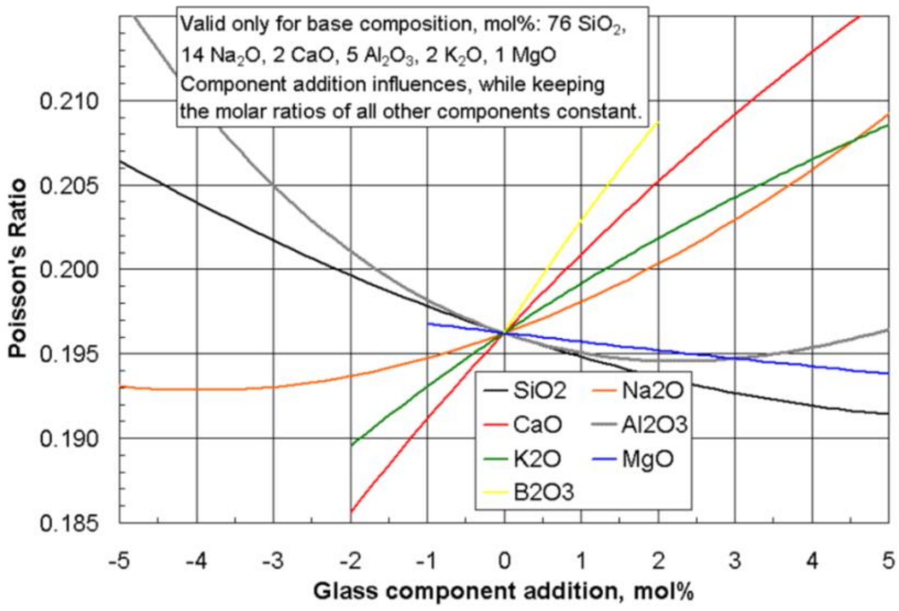


Figure 8: Poisson's Ratio vs. Glass Chemical Composition [14]

In short, Young's modulus is the ratio of stress to strain and is dependent upon many factors about the specific material, such as chemical composition. This property affects fracture patterns and is discussed further in the formulas of section II.2.4.

II.1.4. Material Strength and Toughness

Material strength and toughness is defined as the ability to absorb energy and deform plastically. This plastic deformation is a result of the material resisting the stress loaded onto it. Transverse loading applies a compressional force along the longitudinal axis of the object, while axial loading applies a tensional force perpendicular to that axis, and torsional loading applies a pair of equal and oppositely directed shearing forces. Yield strength measures the lowest amount of force to produce permanent deformation. Compressive strength measures withstanding a compressive force, while tensile strength measures withstanding a stretching force. Fatigue strength measures the strength of a material to repeated force applications or cyclic loading. Impact strength measures the ability to withstand suddenly applied forces and is the focus of this thesis. This strength is often measured by the Izod impact strength test or Charpy impact test. Young's modulus, volume, distribution of force, density, material strength, chemical composition, thickness, and other material properties all play a role in the strength or toughness of a material. Fracture strength of a particular specimen depends considerably on the size, shape, orientation, and distribution of imperfections in the material [11]. For this thesis, flaws are assumed to be randomly distributed throughout the material. This does not allow exact replication of a specific fracture, but an approach that statistically simulates the phenomenon fracture.

II.2. Fracture Mechanics and Fractography of Glass

Fracture mechanics is the study of the propagation of cracks in materials and is used to calculate the force on a crack. Fractography is the study of fracture for understanding the causes of failure and for predicting failure. Alan Arnold Griffith, with later modifications from George Rankine Irwin, developed the basis for modern failure mechanics with his prediction of fracture stress [15]. Much of materials science with respect to fractography and failure models is focused on the determination of whether or not a given material fractures when placed under a specific force. Visual effects artists are often not interested in whether an object fractures or not, as fulfilling storyboards or a desired outcome for the film or game is the top priority. The artists are interested in replicating the results of the fracture as opposed to the calculations leading up to determining if a visible fracture occurs. The system developed in this thesis focuses on the results of a specific fracture rather than whether or not an object fractures when impacted.

Errol B. Shand’s *Glass Engineering Handbook* published in 1958 [34] forms a strong basis for fracture analysis that consolidates and adds new ideas and discoveries to Griffith’s publication of flaw theory of brittle fracture in 1920. By examining the number of radial cracks produced from impact fracture, Shand estimates breaking stress as shown in Figure 9.

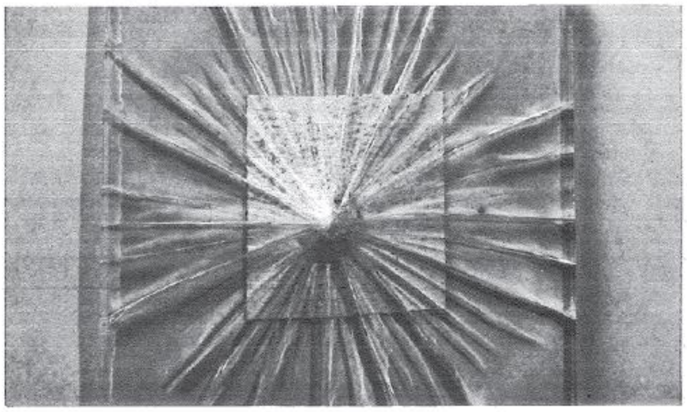


Figure 6
Fracture pattern, glass plate broken by impact.

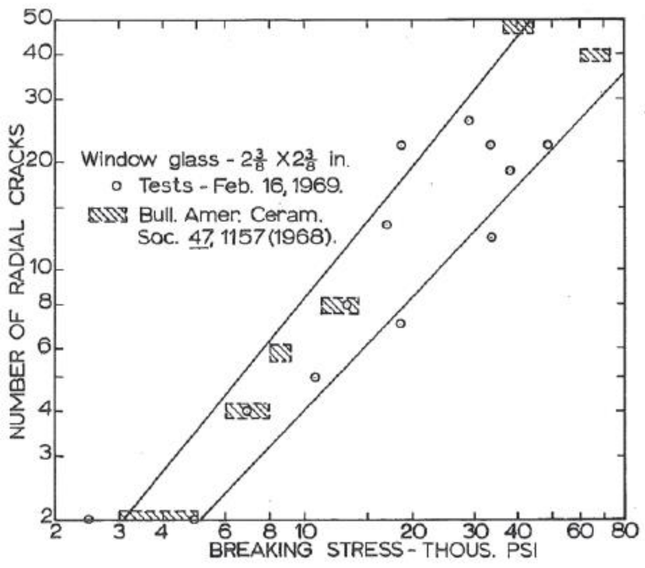


Figure 7
Breaking stress versus number of radial cracks around origin. Window glass broken by impact.

Figure 9: Radial Cracks vs. Breaking Stress [34]

II.2.1. Crack Initiation and Propagation

When a material does fail due to an applied force, atomic bonds are broken and a crack is initiated. In brittle materials like glass, crack propagation is very fast. Common glass cracks often propagate at over 1,000 meters per second [3] as shown in Figure 10. Due to this high speed and visual effects often using a rendering speed of 60 frames per second or less, including crack propagation is typically unnecessary. The system developed in this thesis cracks brittle materials like glass over the course of one frame.

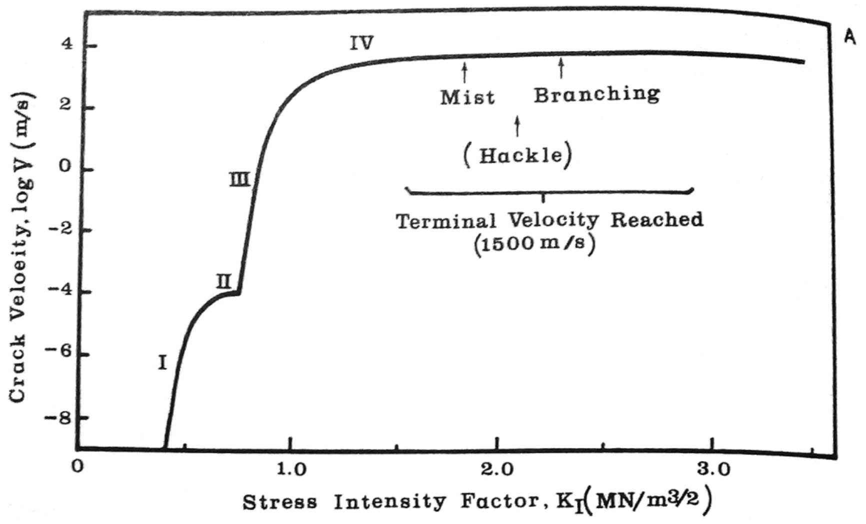


Fig. 3. (A) Schematic diagram of the crack velocity as a function of the stress intensity factor for soda-lime-silica glass,

Figure 10: Crack Velocity vs. Stress Intensity Factor [35]

Glass is not a crystalline solid but rather is an amorphous substance that contains microscopic imperfections of groups of atoms with different crystal orientations. The location where these groups with different orientations meet is a grain boundary. In glass, cracks propagate along these weaker grain boundaries called intergranular fracture. Figure 11 shows a schematic of the intergranular crack fracture that occurs in glass fracture.

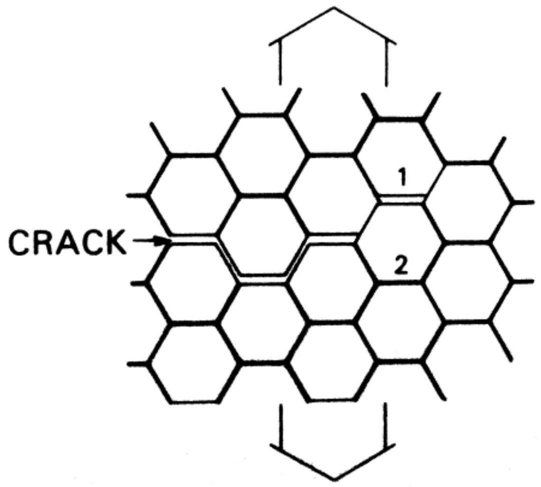


Figure 11: Schematic of Intergranular Crack Fracture [32]

Crack energy dissipates as distance increases from the point of impact, as energy is lost by breaking atomic bonds along grain boundaries. If there is a lack of energy, a crack may terminate before reaching the edge of the glass specimen. Figures 12 and 13 show a top and side view, respectively, of this dissipation.

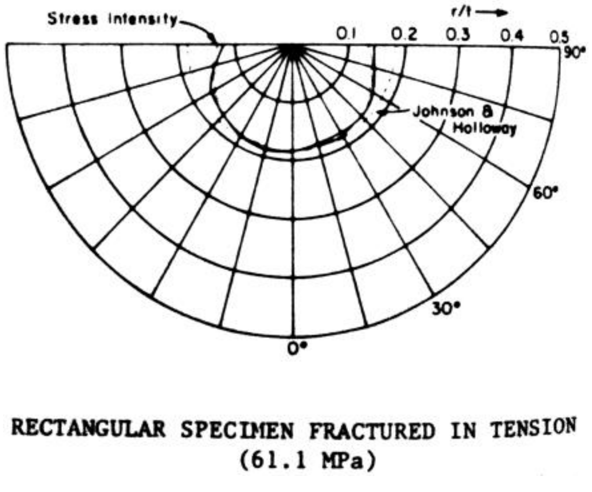


Figure 12: Top View of Energy Dissipation [17]

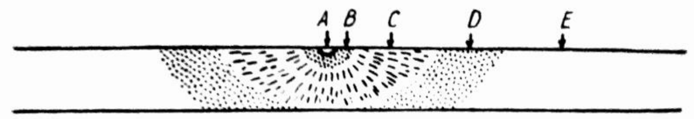


FIG. 33. Explosion centre and distribution of hackle-texture.

Figure 13: Side View of Energy Dissipation [28]

II.2.2. Mirror, Mist, and Hackle Regions

Another phenomenon of impact fracture is the formation of mirror, mist, and hackle regions directly surrounding the impact location. The mirror region is very smooth and highly reflective. The mist region is slightly rougher and less reflective. The hackle region is very rough with large irregularly oriented facets. These regions are only a few millimeters wide, and often would not be seen in a visual effects rendering for a film or game. These regions are not be modeled by this thesis solution and could be more easily represented by a texture map as opposed to geometrical alterations or polygonal objects. These regions are shown in Figures 14 and 15.

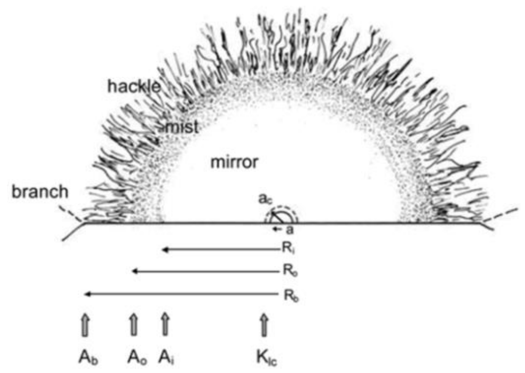


Figure 14: Diagram of Mirror, Mist, and Hackle Regions [1]

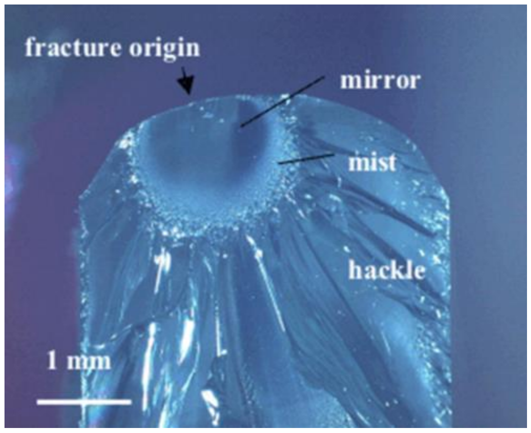


Figure 15: Photograph of Mirror, Mist, and Hackle Regions [42]

II.2.3. Radial and Concentric Crack Behavior

The cracks formed from impact fracture that radiate outward from the impact location are called radial fractures. If the glass is held in a frame, lateral or concentric fractures may form around the point of impact, as shown in Figure 16.

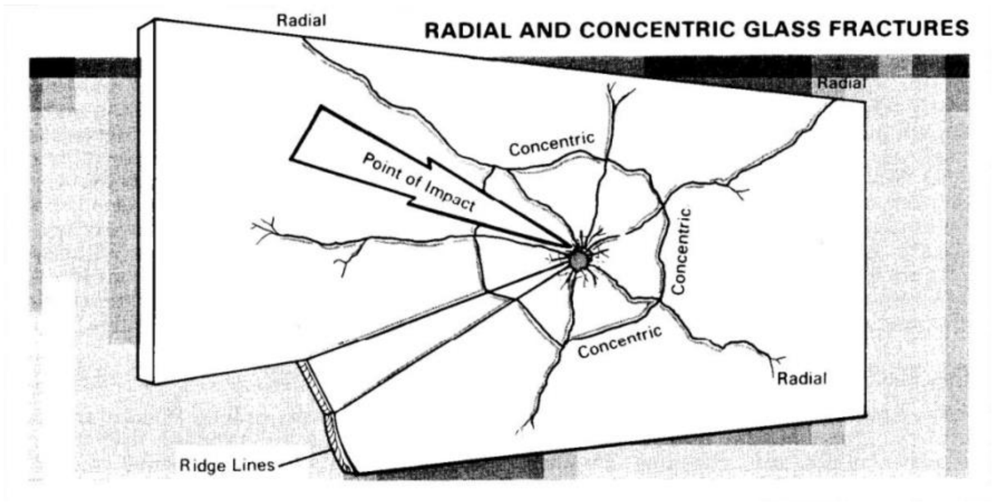


Figure 16: Front View of Radial and Concentric Glass Fracture [2]

Concentric or circumferential cracks are formed due to the resistance from the glass being held in a frame against the force pushing the glass out of it. In Figure 17, notice the directions and angles of the two kinds of fractures. Accurately modeling these crack types and crack features such as jaggedness and crack branching is a step towards to achieving believability. Note that small flakes are not be modeled in the presented solution. The artist may choose to add a particle system afterward to simulate this phenomenon as opposed to this system creating thousands of modeled flakes each with their own collision meshes.

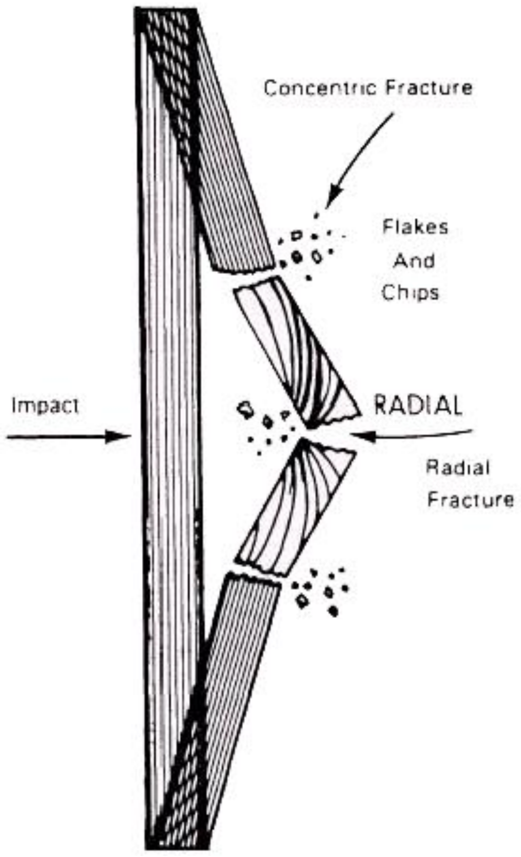


Figure 17: Side View of Radial and Concentric Glass Fracture [24]

II.2.4. Fracture Predictions and Patterns

Low-velocity impact, high-velocity impact, and thermal fractures in glass can be differentiated by their unique fracture pattern features. Crack branching occurs when an applied stress is sufficiently high [35]. In low velocity projectile impact, radial cracks form and propagate outward from the point of impact, and concentric cracks may form if the glass is held in a frame [33]. High velocity projectile impacts often produce a cone or crater where the exit side is larger than the entry side [33], as shown in Figure 18. Thermal fracture looks quite different from projectile impact fracture, as the cracks are curved with smooth edges and have no indication of a point of origin [33].

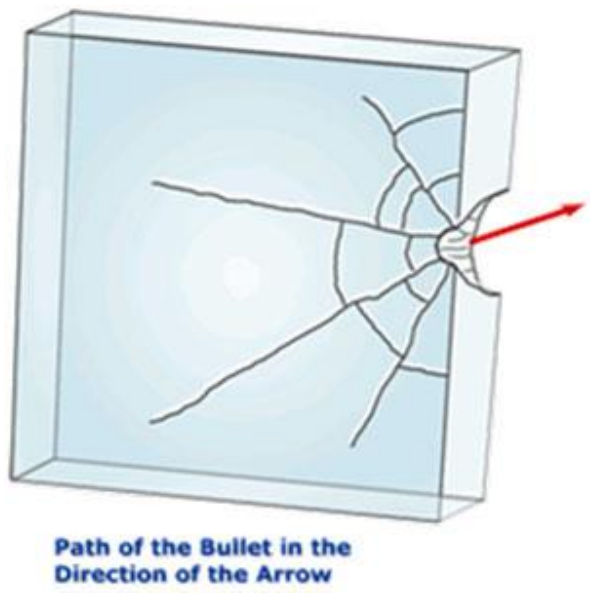


Figure 18: High Velocity Impact Fracture [10]

It has also been hypothesized that fracture in brittle materials follow a self-similar process of crack propagation and could be modeled through fractal geometry [22]. The term “fractal” was coined by Mandelbrot in 1975, derived from the Latin “fractus” meaning broken, and describes a self-similar process in which a feature at one magnification is related to another at another magnification by a scalar quantity [21]. A definitive formula utilizing fractal mathematics for accurate glass fracture has yet to be discovered or developed.

The number of pieces generated from impact fracture has been measured in various studies. Locke and Unikowski [19] created a testing environment shown in Figure 19 that records how many pieces fly back towards the direction of the impact to a glass pane held in a frame. Using this breaking rig in multiple tests, calculations can be made by counting up the number of the pieces in each tray and recording their size. Their results show that larger pieces fall more directly backwards while smaller pieces are scattered laterally more often. This thesis is focused primarily on fracture patterns, as shown by Locke and Unikowski [20] in Figure 20.

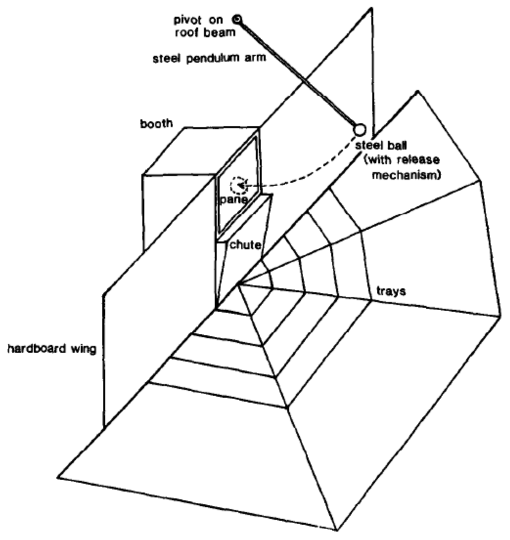


Fig. 1. The breaking rig.

Figure 19: Breaking Rig for Calculating Glass Distance Travelled Backwards [19]

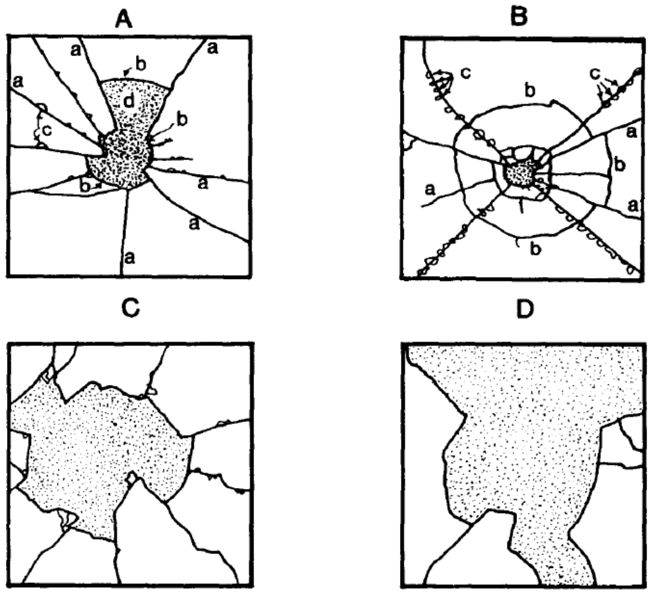


Fig. 2. Examples of broken panes showing the following features: A, 6 mm plain glass (1.0 × 1.0 m) radial cracks; B, 6 mm wired glass (1.0 × 1.0 m) circumferential cracks; C, 4 mm patterned glass (0.25 × 0.25 m) regions of surface flaking and D, 4 mm plain glass (1.0 × 1.0 m) hole resulting from large pieces dropping vertically down. Note: the scale of C is different from the other three diagrams.

Figure 20: Examples of Broken Panes with Different Properties [20]

J. L. Ladner, D. L. Ahearn and R.C. Bradt's [18] research shows the effect of glass panel lamination on fracture patterns, as shown in the diagrams in Figure 21. Annealed, tempered, and laminated glass panels all have identifiable impact centers, but exhibit different features [18]. Annealed glass exhibits classical radial crack patterns, impact resistant glass exhibits some star-like radial crack patterns with circumferential cracks, and tempered glass exhibits a dicing pattern that occurs due to the high level of stored elastic strain energy [18]. Figure 22 shows typical fracture patterns in laminated glass in car windshields from an impact velocity of 10 meters per second [35].

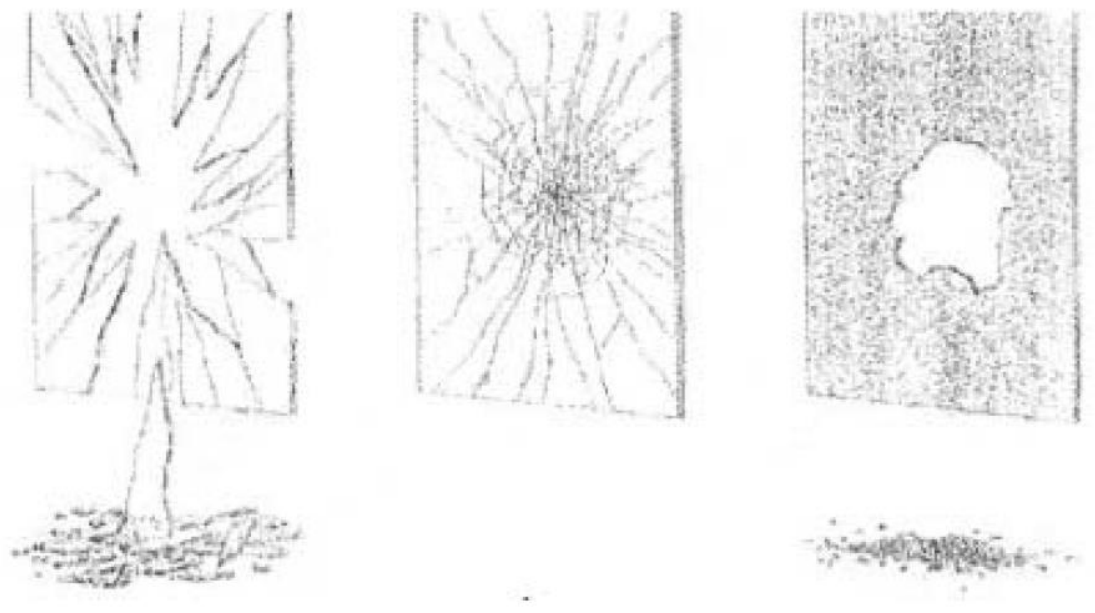


Figure 21: Impact Fracture Patterns on Annealed Glass (left), Laminated Impact Resistant Glass (center), and Tempered Glass (right) [18]

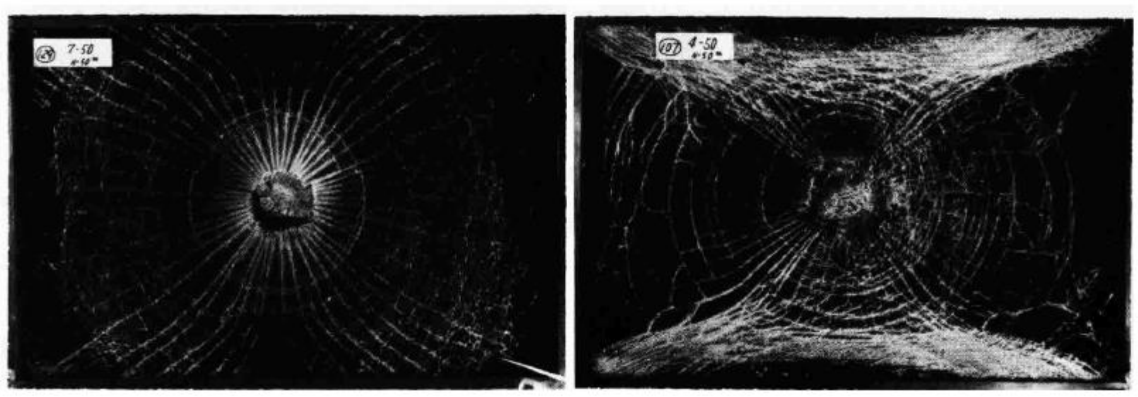


Figure 22: Typical Fracture Pattern of Laminated Glass Broken by a Headform Impact (Falling Height: 5.0m, Impact Velocity: 10m/s) [35]

Norihiko Shinkai's research, "*The Fracture and Fractography of Flat Glass*", also observes the fundamental features, marks, and characteristics of cracks in glass [35]. The fracture patterns resulting from uniform pressure as opposed to impact force are distinctly different, as shown in Figure 23. The focus of this thesis is on impact fracture.

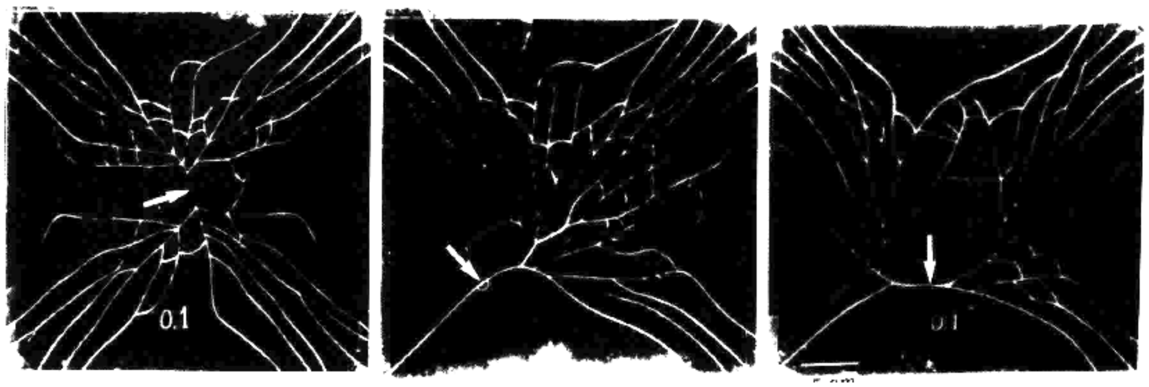


Figure 23: Crack Patterns Produced on Flat Glass from Uniform Loading [35]
(Arrows indicate the initiating points of the fractures)

In impact fracture, radial cracks are propagated, and tangential cracks may form when the glass is thin or when the glass is subjected to high speed impacts [35] as shown in Figures 24 and 25.

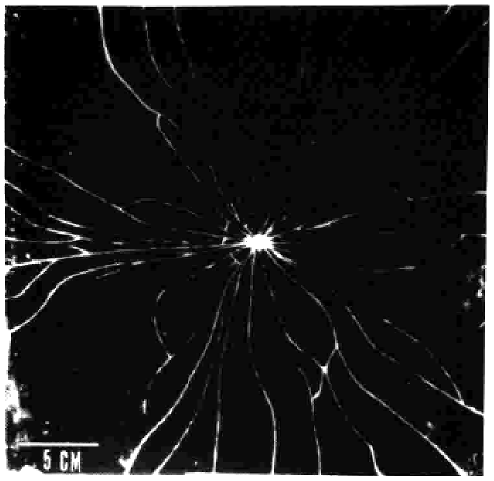


Figure 24: Crack Pattern Produced on a Glass Plate from a Ball Falling
(Thickness of Glass: 2mm, Ball Weight: 173g, Impact Velocity: 3.4m/s) [35]

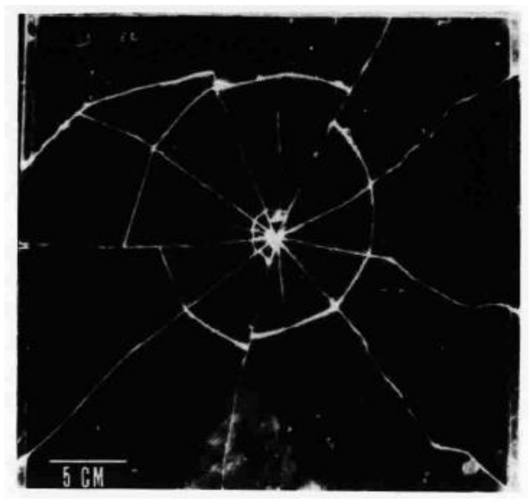


Figure 25: Crack Pattern Produced on a Glass Plate from a Ball Falling
(Thickness of Glass: 3mm, Ball Weight: 66.7g, Impact Velocity: 6.0m/s) [35]

Nicolas Vandenberghe, Romain Vermorel, and Emmanuel Villermaux's research, "*Star-Shaped Crack Pattern of Broken Windows*", explores a global scaling law for the number of radial cracks generated in brittle plates from impacts. Their model, based on Griffith's theory of fracture and fracture energy, predicts that the number of radial cracks can be estimated based on impact speed and plate thickness, as shown in Figure 26.

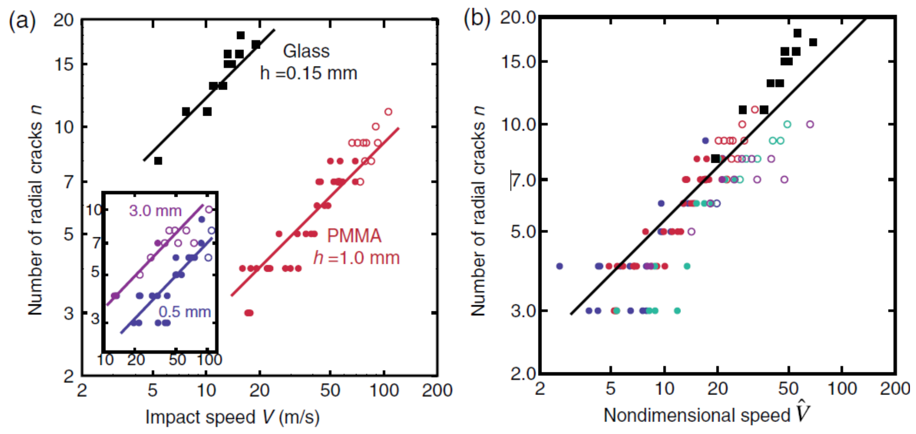


Figure 26: Number of Radial Cracks Increasing with Impact Speed (A), Rescaled Using Non-Dimensional Speed (B) [39]

The formula for non-dimensional speed (V^) versus radial cracks is calculated as V^=$(E*h/\gamma)^{2/3}$(v/*c*), where V^ is the non-dimensional speed, *E* is Young's Modulus, *h* is the thickness of the plate, γ is the fracture energy of the material, v is the velocity of the impact, and *c* is the speed of sound in the material, calculated as $c = (E/\rho)^{1/2}$ with $E = Y/(1-v^2)$, where *E* is Young's Modulus, *ρ* is the density of the material, and *v* is the Poisson's Ratio of the material [39]. Their formula accurately models the brittle materials tested: flat PMMA and glass plates of various thickness. The continuous line in Figure 26 (B) is n=1.7$(V^\wedge)^{1/2}$, where n is the number of radial cracks.

II.3. Visual Effects Approaches to Fracturing Glass

Visual effects artists working on films and games have used a wide variety of approaches to solving the problem of object fracture on-screen. Live action films can use practical special effects (SFX) such as sugar glass which is safe for an actor to break through or fall on. However, sugar glass doesn't break into large and sharp shards like many fully annealed glass plates produce [7], which decreases the believability of the stunt. Many films, such as "*The Avengers*", or "*Mission: Impossible - Ghost Protocol*" as shown in Figure 27, use various 3D computer software packages to create virtual glass that is either composited into the live action footage or rendered into the CG film. For example, DreamWorks's "*Megamind*", is a complete 3D CG film with no live action elements and relies on visual effects (VFX) solutions for object fracture.

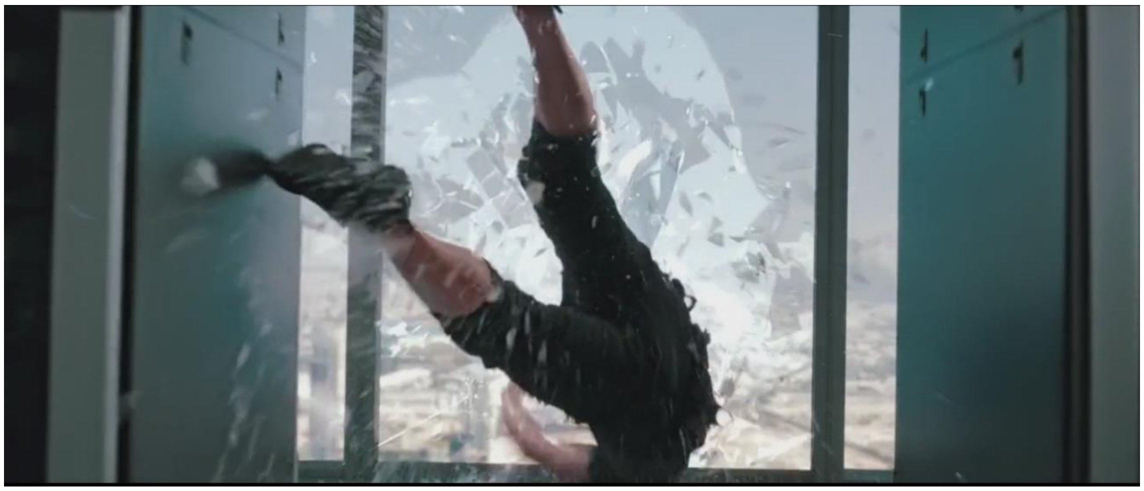


Figure 27: Virtual Glass Fracture in "*Mission: Impossible – Ghost Protocol"* [26]

Objects created in 3D software packages using simple primitives, like a rectangular prism, have no inherent interior detail or structure. A basic method of dividing an object for fracture would be to manually create more geometry by creating additional span lines, vertices, and subdividing polygonal faces. This method of breaking up an object is non-automated, time consuming, and not based upon any real world fracture parameters. The next sections discuss previous and current approaches that automate this process.

II.3.1. Finite Element Solutions

Finite element analysis (FEA), or finite element method (FEM), is a computational technique to find approximate solutions to structural analysis by calculating stresses. This method generates an interior system of points called nodes that make a grid called a mesh. The mesh is given properties such as mass, volume, temperature, stress, strain, and an input force with velocity and acceleration. The mesh grid then reacts to the incoming force through nodes reacting to each other and calculating node connections pushing or breaking due to stress or strain. While this method can be very accurate to simulate real world physics of gravity and collisions, it is very computationally expensive. One of the earliest pioneers of finite element analysis used outside of mechanical engineering and for computer graphics is James F. O'Brien. His Ph.D. thesis "*Graphical Modeling and Animation of Brittle Fracture*" from Georgia Institute of Technology in 2000, also published in Siggraph 1999, demonstrates this approach to breaking objects based on linear elastic fracture mechanics through finite element analysis, shown in Figure 28.

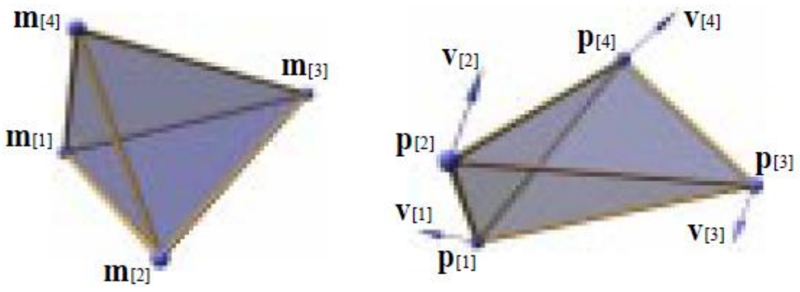


Figure 28: A Tetrahedral's Nodes, Each Containing Material Coordinates (left), Positions, and Velocities Values (right) [25]

A current piece of visual effects software that uses this approach is Digital Molecular Matter (DMM) developed by Pixelux Entertainment where O'Brien served as a consultant. DMM has been used in over 20 feature films, including *"X-Men: First Class"*, *"Avatar"*, *"Source Code"*, *"Mission Impossible 4"* (as shown in Figure 27 on page 26), and was used in the game engine of *"Star Wars: The Force Unleashed"* [27].

FEA can achieve believable results, but is significantly more computationally intensive than an algorithmic or texture based approach. Also, the FEA solution is very hard for an artist to tweak for art directability. If the artist desires an extra concentric crack or radial crack, the artist only has access to real world parameters such as stiffness or mass. There is no way to override the final produced fracture patterns. Achieving a radial and concentric pattern out of finite element analysis requires a very large, accurate, and complicated system.

II.3.2. Algorithmic Solutions

Another way of automating the process of dividing 3d polygonal meshes is to use a space division algorithm. This automation can save the artist large amounts of time, especially if there are hundreds of objects that need to be fractured. A common algorithm used for dividing 2D or 3D spaces is the Voronoi algorithm. The artist adds any number of placed or randomized seed points inside the geometry of the object and then uses the Voronoi algorithm which divides the space such that each seed has a corresponding region consisting of all points closer to that seed than to any other seed. However, I feel this solution tends create a stone or concrete fracture type of look, as the regions mimic intergranular fracture as found in concrete due to its aggregate interior. Figure 29 shows a comparison between transgranular fracture (A), where cracks pass through grains, intergranular fracture (B), where cracks propagate along grain boundaries, and randomized seed Voronoi (right).

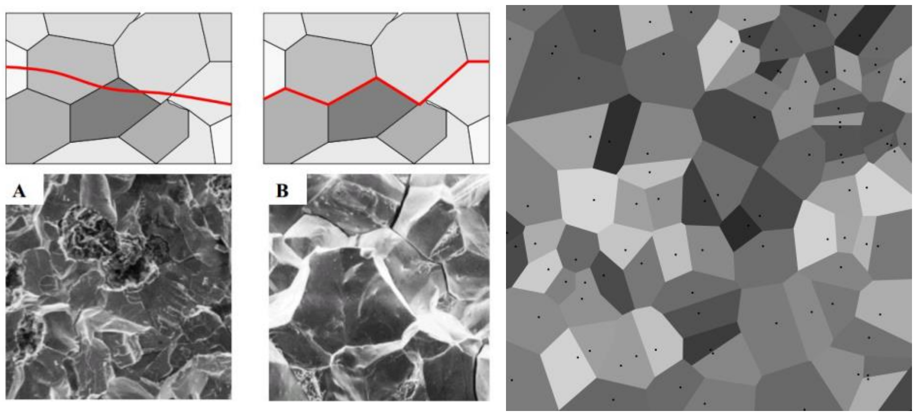


Figure 29: Transgranular Diagram and Photograph (left), Intergranular Fracture Diagram and Photograph (center), and Randomized Voronoi (right) [8]

While randomized Voronoi is a somewhat accurate solution to modeling how concrete might fracture because it accounts for the various sizes of aggregate and interior air bubbles, this solution cannot model the concentric and radial crack patterns of glass. Many current visual effects solutions still use the Voronoi algorithm for glass, but manipulate the seed points to attempt to mimic these patterns, as shown in Figure 30. Notice the inaccuracy of creating disjointed and angular concentric crack patterns due to the simultaneous creation of radial and concentric cracks. Radial cracks propagate first through the material and do not branch irregularly at the locations that the concentric cracks that later occur. Radial cracks branch independently of the location of the concentric cracks. This is a feature that is impossible to generate using Voronoi seed placement for division.

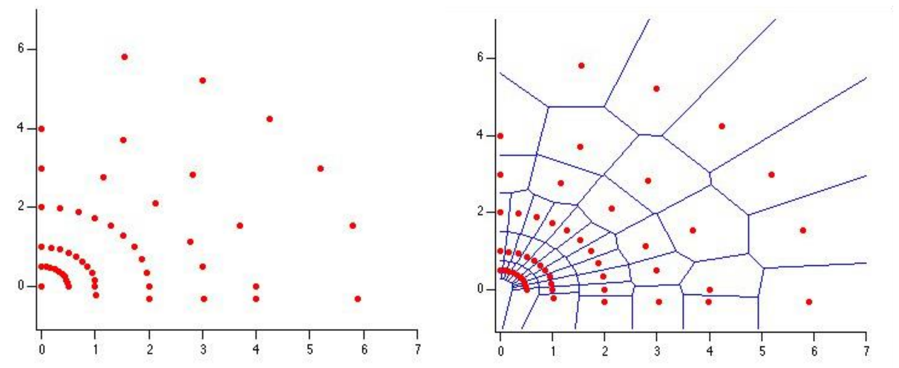


Figure 30: Voronoi Seed Placement and Dividing Regions [41]

Examples of current software that use this approach are RayFire which is a plugin for 3ds Max, and FractureFX which is a plugin for Maya. RayFire was used in games such as *"Batman Arkham City"* and "*Diablo 3"*, and films such as "*The Avengers"*, “*Transformers 3”*, and “*Star Trek Into Darkness”* [31]. FractureFX was used in the films and TV shows *“Snow White and the Huntsman”*, “*Wrath of the Titans”*, “*Heroes”*, “*Dredd 3D”*, and more [12].

Both software packages support manually adding more geometry, a texture-to-fragment solution, and Voronoi algorithmic division with many variations for the seed placements. The seed placement can be random or semi-random to achieve specific looks such as wood splinters, brick patterns, or radial distribution for glass. Radial distribution of seed points does not match real fracture mechanics of glass. Figures 31-34 show RayFire’s Voronoi distribution methods for glass.

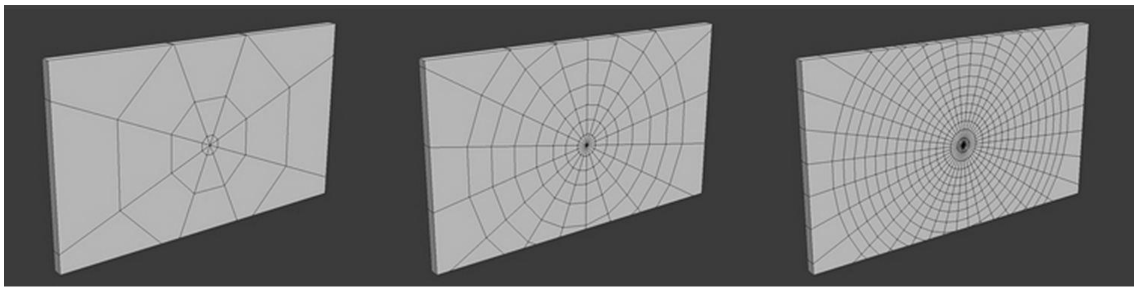


Figure 31: RayFire Radial Voronoi – Number of Radial/Concentric Cracks [31]

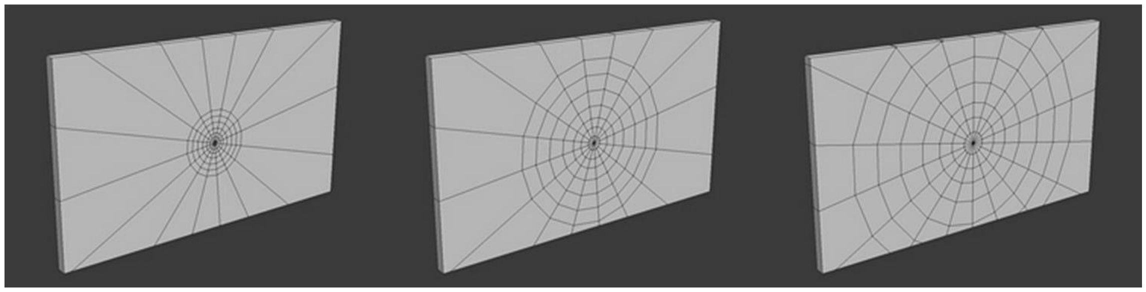


Figure 32: RayFire Radial Voronoi – Size of Concentric Cracks [31]

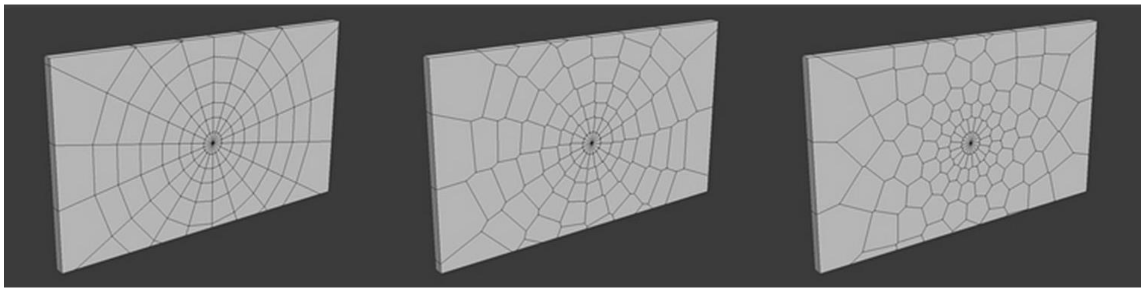


Figure 33: RayFire Radial Voronoi – Radial Bias Parameter [31]

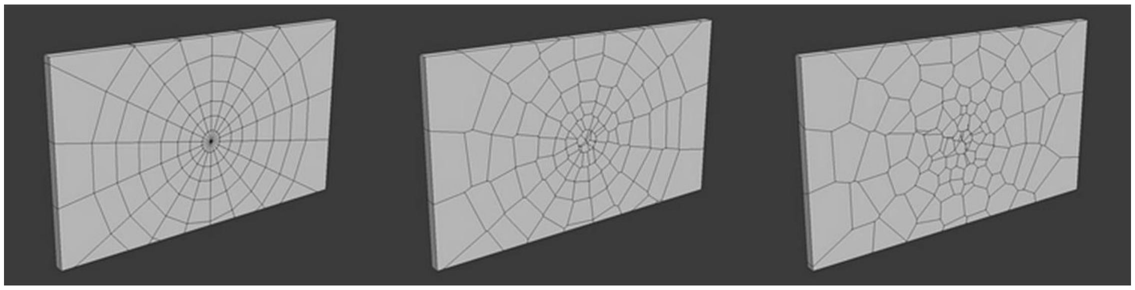


Figure 34: RayFire Radial Voronoi – Divergence Parameter [31]

In Figure 31, the radial cracks have exactly the same degree angle between each other. Real world glass fracture patterns are not so perfect. The concentric cracks also have the same distance between each perfect ring. Figure 32 shows variation in the number of concentric cracks, which can be controlled by the artist. Figures 33 and 34 show two parameters called “radial bias” and “divergence” that the artist can set. At a value of zero, the parameters have no effect, as shown in the left most examples, and at value of 100 the parameters greatly effects the seed distribution, as shown in the right most examples. These parameters do add variance by twisting or randomizing the seed points which does get rid of the perfect circles and exact same degrees between radials, but unrealistically bends the radial cracks at the intersections of the concentric cracks.

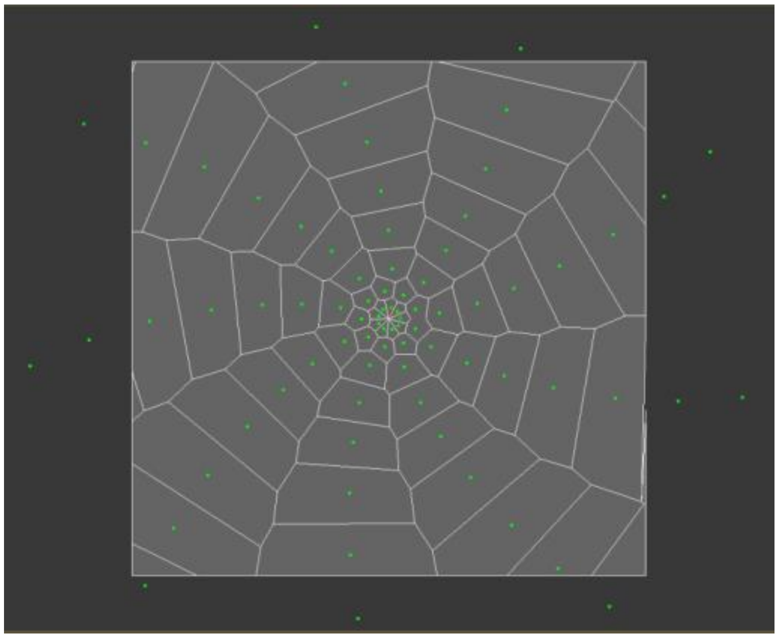
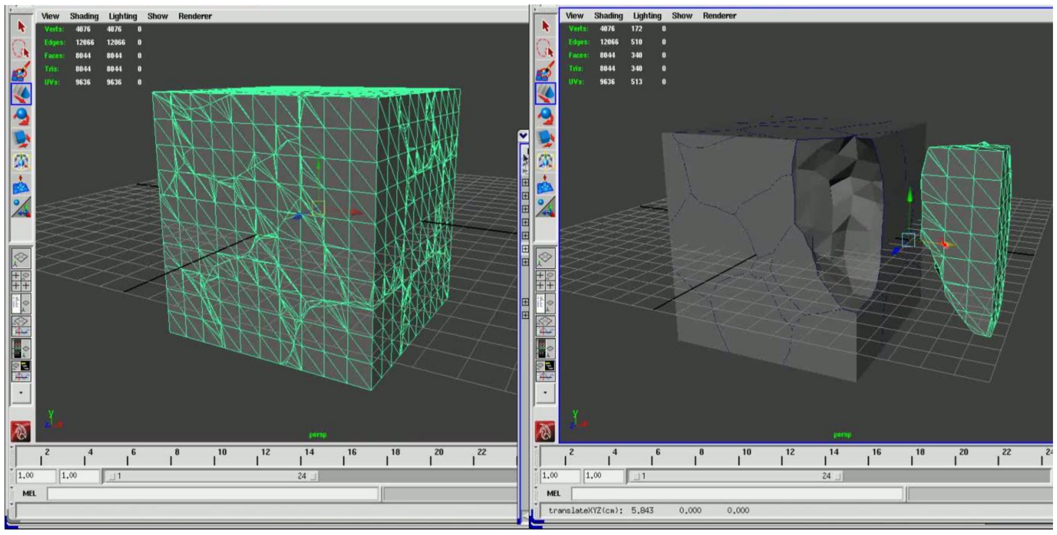


Figure 35: 3ds Max Screenshot Showing Seed Points Using RayFire

Figure 35 shows a screenshot from 3ds Max with the RayFire modifier applied to rectangular prism and has the seed points visible. The parameter used in this screenshot is "radial bias" at a value of 50. None of these parameters allow for radial crack branching, editable jaggedness, or imperfect randomization without disjointing the radial cracks at each concentric ring. This system is quick and computationally fast, but ultimately Voronoi is not a good representation of glass fracture. Other software that uses the Voronoi approach is PullDownIt, a plugin for 3ds Max and Maya made by Thinkinetic [38] and Cell Fracture, a plugin for Blender [6]. A Voronoi based solution was also used in the Walt Disney film *"Bolt"* with research by Hellrung, et al [16], shown in Figure 36.



```
Verts:   4076   4076   0
Edges:  12066  12066   0
Faces:   8044   8044   0
Tris:    8044   8044   0
UVs:     9636   9636   0
```

```
Verts:   4076    172   0
Edges:  12066    510   0
Faces:   8044    340   0
Tris:    8044    340   0
UVs:     9636    513   0
```

```
translateXYZ(cm):  5.843    0.000    0.000
```

Figure 36: Independent Fragments Generated Through Voronoi Algorithm [16]

Another use of the Voronoi algorithm was by DreamWorks artist Raghavachary in his paper "*Fracture Generation on Polygonal Meshes using Voronoi Polygons*", shown in Figure 37, and "*3000+ Variations of the Voronoi Diagram*", showing alterations on the Voronoi algorithm, shown in Figure 38. RayFire and FractureFX also support a texture-to-fragment solution which is discussed in the next section.

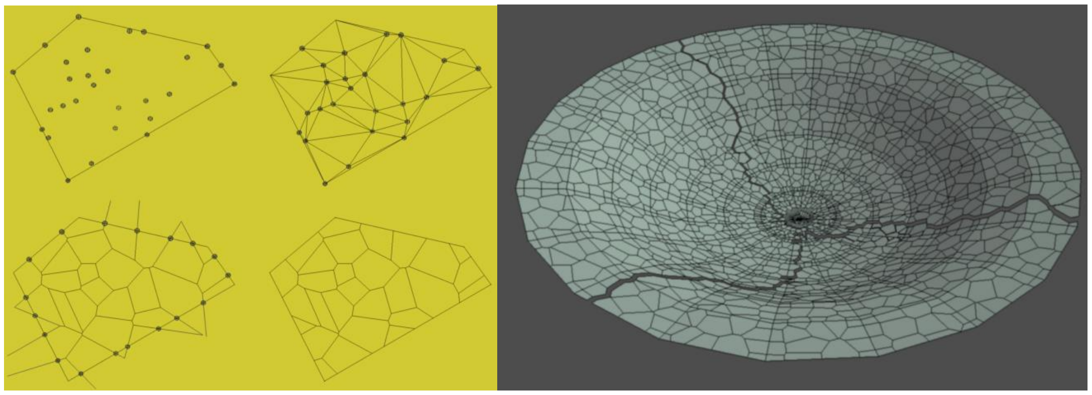


Figure 37: Voronoi Seed Points (left) and Generated Ceramic Cracks (right) [29]

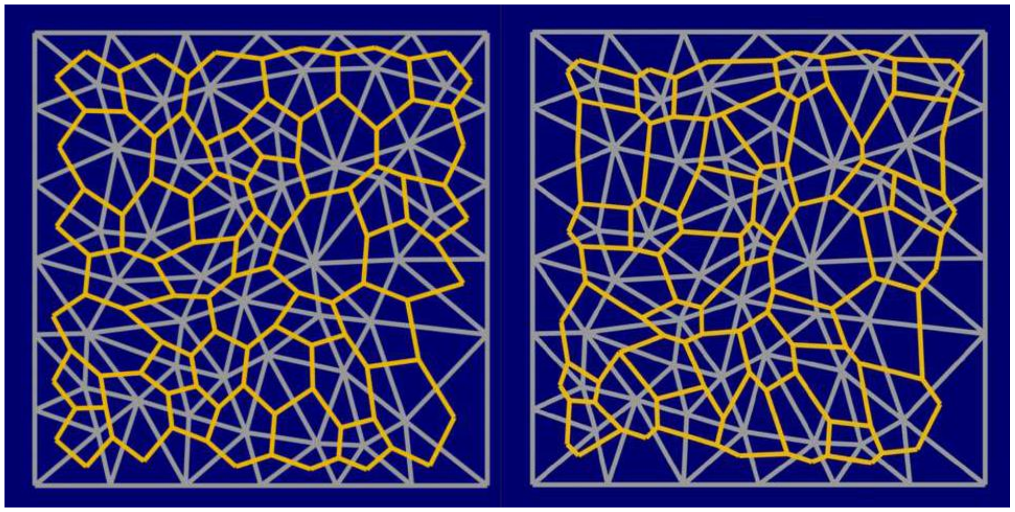


Figure 38: Delaunay Triangulations with Altered Voronoi Region Calculations [30]

The Voronoi algorithm can also be used in a volumetric approach, as seen in Müller's research, "Real Time Dynamic Fracture with Volumetric Approximate Convex Decompositions" [23]. Their solution is very fast and can be computed in real time and can use the impact location to influence seed positioning to vary the size of pieces generated, but also does not avoid the problems of radial versus concentric cracks, crack branching, or art directability.

Ultimately, the Voronoi algorithm method of dividing space is fundamentally different from that where radial and concentric cracks divide space. While the solution is computationally fast, it is inaccurate due to the combined creation and forced integration between the two different crack propagation phenomena.

II.3.3. Texture Based Solutions

Another common approach is using a texture map edited by the artist and applied to each specific object, where black lines drawn on the texture map are the dividing lines for new geometry. Photographs of real glass fracture can be converted to black and white and used as the dividing lines, as shown in Figure 39.

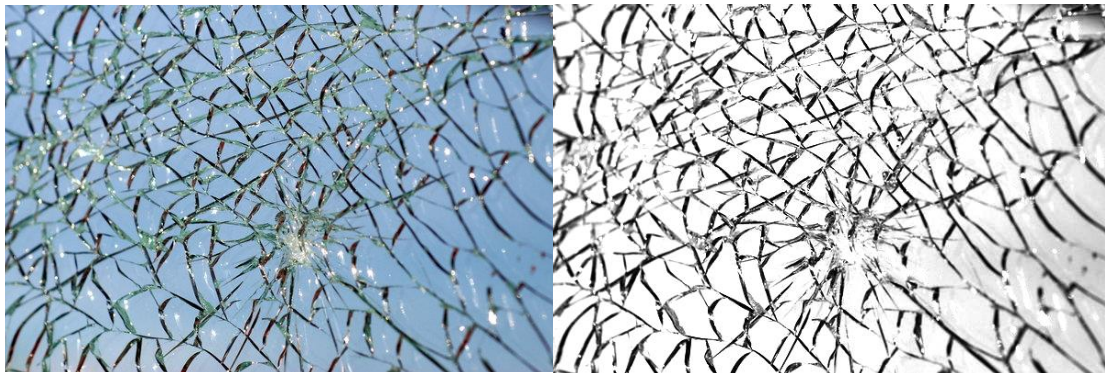


Figure 39: Photograph (left) Used for Divided Regions (right)

This solution can be accurate, as an artist can use a real glass fracture image as the base for the texture map. The main drawback to this solution is that the artist needs a unique texture map for each unique fracture. Examples of this approach in software can be found in BlastCode, made by Blast Code Inc., which is a plugin for Maya that supports texture-to-fragments and procedural texture generation system. BlastCode was used in many films and games such as *"Megamind"*, *"Kung Fu Panda"*, *"Transformers"*, *"Portal 2"*, *"Left 4 Dead"*, and *"Half Life 2"* [5].

III. METHODOLOGY AND IMPLEMENTATION

The methodology of this thesis is a pattern generation solution that avoids long simulation times and the requirement of fixed assets or maps, while being consistent to materials science research and maintaining art directability. The fracture patterns are in the form of spline shapes that are based upon the virtual collision, object properties, and materials science literature. All of the patterns generated are completely unique through variation and random number generation for crack angles and branching, avoiding the problems of two different objects using the same map and yielding the same result. The individual parameters about each fracture, such as the number of radial cracks, are determined upon real world impact analysis and can easily be changed should the artist desire. This spline pattern method also separates the creation of radial and concentric cracks, which is necessary for realistic solutions, broader scenarios, and allows the user to tweak them individually. After generating a pattern, the tool can quickly use it to break apart the 3D polygonal object through a scripted interface button that extrudes the patterns, finds intersections, generates new vertices, and creates each fragment for every object to be fractured. In addition to setting object properties, finding collision points, calculating overrideable pattern properties, and generating object pieces from the patterns, the solution can bake the simulation into regular keyframe animation and quickly undo any step in the process for flexibility. This tool is scripted in MAXScript, the built-in scripting language for 3ds Max, and utilizes the integrated physics engine MassFX for object simulated animation after impact.

III.1. Development Environment

This work was developed on a personal computer running Windows 7 Ultimate 64-bit. The computer consists of an Intel i7-860 processor running at 2.93 GHz, an NVidia GTX 460 graphics card with 1.5 GB of dedicated memory, and 16 GB of RAM. The software used is the student trial version of Autodesk 3ds Max 2014 (64-bit).

III.1.1. 3ds Max and MAXScript

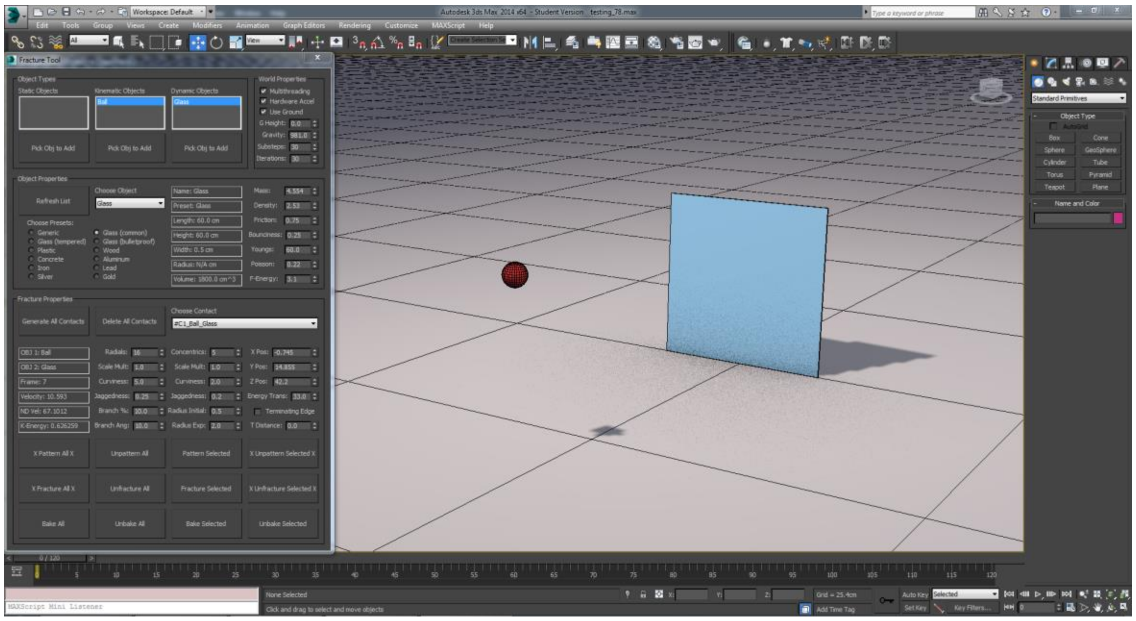
3ds Max is software developed and produced by Autodesk and has comprehensive 3D modeling, animation, rendering, and compositing solutions for games, film, and motion graphics artists. MAXScript is a built-in scripting language used to automate repetitive task, combine existing functionality in new ways, and develop new tools and user interfaces, and more. This thesis uses MAXScript to define a custom user interface with new tools for automating a solution to fracturing flat glass realistically when impacted by an object.

III.1.2. MassFX Physics Simulation

MassFX is a unified physics simulation framework, introduced by 3ds Max 2012 and is based on the PhysX SDK made by NVidia. The MassFX system is almost completely integrated and accessible to MAXScript for automation, and is very intertwined with the code presented in this thesis. If necessary, it would be possible to take out all of the MassFX elements out of the script and to use a different physics engine for simulation if desired.

III.2. MAXScript Implementation

This section presents an overview of the scripts process in III.2.1 followed by a chronological explanation of the properties, parameters, and details of the solution. To run the script from 3ds Max, a user only needs to click on “MAXScript” on the main tool bar, select “Run Script”, and navigate to the script file. Figure 40 shows a screenshot of the graphic user interface (GUI) that appears upon running the script.



Static Objects:
Kinematic Objects: Ball
Dynamic Objects: Glass

World Properties: [✓] Multithreading [✓] Hardware Accel [✓] Use Ground; G Height: 0.0; Gravity: 981.0; Substeps: 30; Iterations: 30

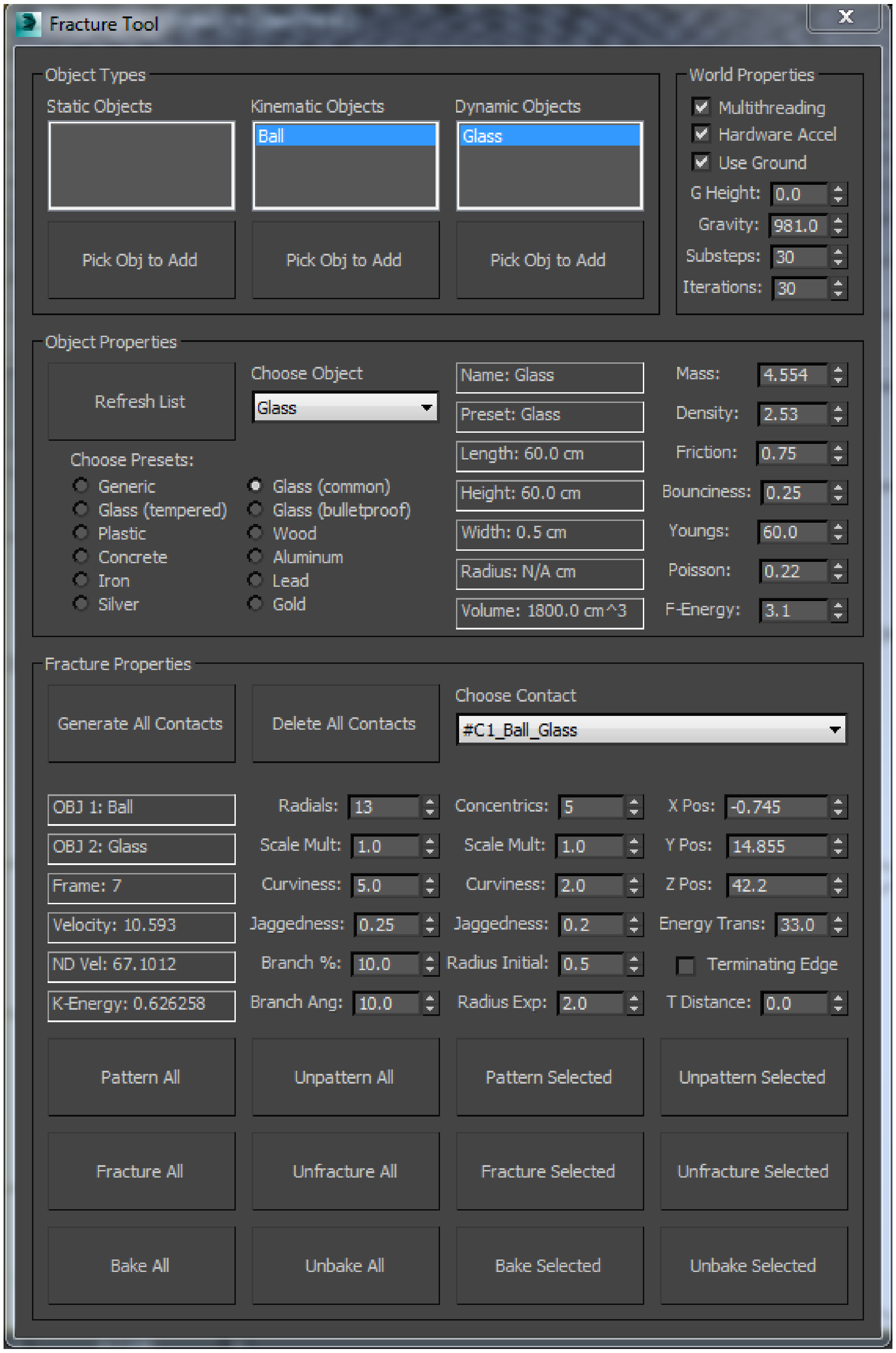
Object Properties
Choose Object: Glass
Choose Presets: Generic, Glass (tempered), Plastic, Concrete, Iron, Silver, [✓ Glass (common)], Glass (bulletproof), Wood, Aluminum, Lead, Gold
Name: Glass
Preset: Glass
Length: 60.0 cm
Height: 60.0 cm
Width: 0.5 cm
Radius: N/A cm
Volume: 1800.0 cm^3
Mass: 4.554; Density: 2.53; Friction: 0.75; Bounciness: 0.25; Youngs: 60.0; Poisson: 0.22; F-Energy: 3.1

Fracture Properties
Choose Contact: #C1_Ball_Glass
OBJ 1: Ball
OBJ 2: Glass
Frame: 7
Velocity: 10.593
ND Vel: 67.1012
K-Energy: 0.626259
Radials: 16; Scale Mult: 1.0; Curviness: 5.0; Jaggedness: 0.25; Branch %: 10.0; Branch Ang: 10.0
Concentrics: 5; Scale Mult: 1.0; Curviness: 2.0; Jaggedness: 0.2; Radius Initial: 0.5; Radius Exp: 2.0
X Pos: -0.745; Y Pos: 14.855; Z Pos: 42.2; Energy Trans: 33.0; [ ] Terminating Edge; T Distance: 0.0

Figure 40: 3ds Max Screenshot, Script GUI Open on Left

III.2.1. Script Overview

An artist needs to first create a simple scene in 3ds Max, with at least one object moving through another with a simple keyframe animation. Afterwards, the artist then runs the script, and begins supplying information about the objects they have created. The artist must specify what types of motion and what kind of materials the objects are, and then may proceed to finding the collision between these two objects with the button "Generate All Contacts". Default fracture parameters are set based upon the given material properties and initial animation. These parameters may be tweaked if an artist desires. The artist then can generate a fracture pattern that uses those parameters and divide the object into pieces based on that pattern. Lastly, the artist can bake the animation, which saves the physical simulation into keyframe animation. At any step in this process, the artist may use the corresponding buttons to undo any and all steps, such as unbaking the animation, returning the original object whole, deleting created fracture patterns, and removing generated collision locations. Figure 41 displays the script GUI.



Static Objects:

Kinematic Objects: Ball

Dynamic Objects: Glass

World Properties:
[✓] Multithreading
[✓] Hardware Accel
[✓] Use Ground
G Height: 0.0
Gravity: 981.0
Substeps: 30
Iterations: 30

Choose Object: Glass

Choose Presets:
- Generic
- Glass (tempered)
- Plastic
- Concrete
- Iron
- Silver
- [✓ Glass (common)]
- Glass (bulletproof)
- Wood
- Aluminum
- Lead
- Gold

Name: Glass
Preset: Glass
Length: 60.0 cm
Height: 60.0 cm
Width: 0.5 cm
Radius: N/A cm
Volume: 1800.0 cm^3

Mass: 4.554
Density: 2.53
Friction: 0.75
Bounciness: 0.25
Youngs: 60.0
Poisson: 0.22
F-Energy: 3.1

Choose Contact: #C1_Ball_Glass

OBJ 1: Ball
OBJ 2: Glass
Frame: 7
Velocity: 10.593
ND Vel: 67.1012
K-Energy: 0.626258

Radials: 13
Scale Mult: 1.0
Curviness: 5.0
Jaggedness: 0.25
Branch %: 10.0
Branch Ang: 10.0

Concentrics: 5
Scale Mult: 1.0
Curviness: 2.0
Jaggedness: 0.2
Radius Initial: 0.5
Radius Exp: 2.0

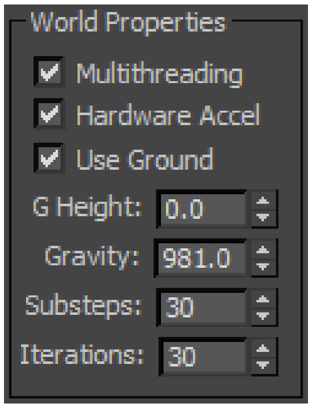
X Pos: -0.745
Y Pos: 14.855
Z Pos: 42.2
Energy Trans: 33.0
[ ] Terminating Edge
T Distance: 0.0

Figure 41: 3ds Max Screenshot, Script GUI

III.2.2. Scene Setup

The scene does not require any editing prior to running and opening the script. It is advised to change the system unit setting to centimeters or meters to avoid user confusion between unit systems. The script is built around displaying and utilizing information in the metric system, but converts inputs if necessary. The user should open the MassFX Toolbar to ensure that the initial startup values are saved and synchronized. The artist needs objects in the scene that collide with one another to use all of the features of the script.

III.2.3. World Properties


World Properties

✓ Multithreading
✓ Hardware Accel
✓ Use Ground
G Height: 0.0
Gravity: 981.0
Substeps: 30
Iterations: 30

Figure 42: Screenshot of World Properties Group

The world properties group of the script is located in the upper right hand corner of the GUI, and is shown in Figure 42.

The first option is a checkbox labeled “Multithreading”, which indicates whether or not the MassFX system executes multiple threads faster on a CPU that supports it. This is different than the 3ds Max rendering Multithreading checkbox, found in the rendering tab of the preference settings window.

The second option is a checkbox labeled “Hardware Accel”, which refers to the ability of your system to use hardware acceleration of some MassFX computations for improved performance. This requires your computer to be equipped with an NVidia GPU.

The third option is a checkbox labeled “Use Ground”, which refers to MassFX’s use ground collisions object, where when turned on, MassFX uses an invisible planar, static rigid body at the specified ground height level.

The forth option is a spinner labeled “G Height”, which refers to the ground height for the previous option. If “Use Ground” is unchecked, the ground height specified here does not interact with the physics simulation.

The fifth option is a spinner labeled “Gravity”, which refers to the global gravity of the entire scene in cm/s^2, applied on the Z axis. The default value is positive 981.0, which accelerates objects downward at approximately Earth’s actual rate.

The sixth option is a spinner labeled “Substeps”, which is synchronized with the corresponding in the MassFX toolbar. This parameter changes the number of simulation steps performed in-between each frame. A default value of 30 is given, in which collisions are correctly found at a highly accurate degree.

The seventh option is a spinner labeled “Iterations”, which is synchronized with the corresponding value in the MassFX toolbar. This parameter changes the number of times the constraint solver enforces collisions and constraints, such as for checking objects’ physical meshes for collisions.

III.2.4. Object Types

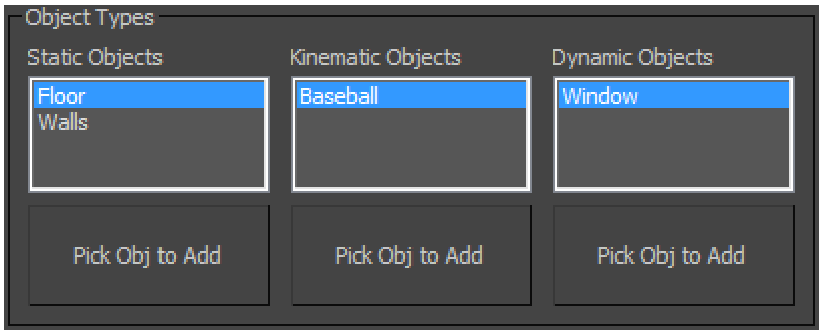


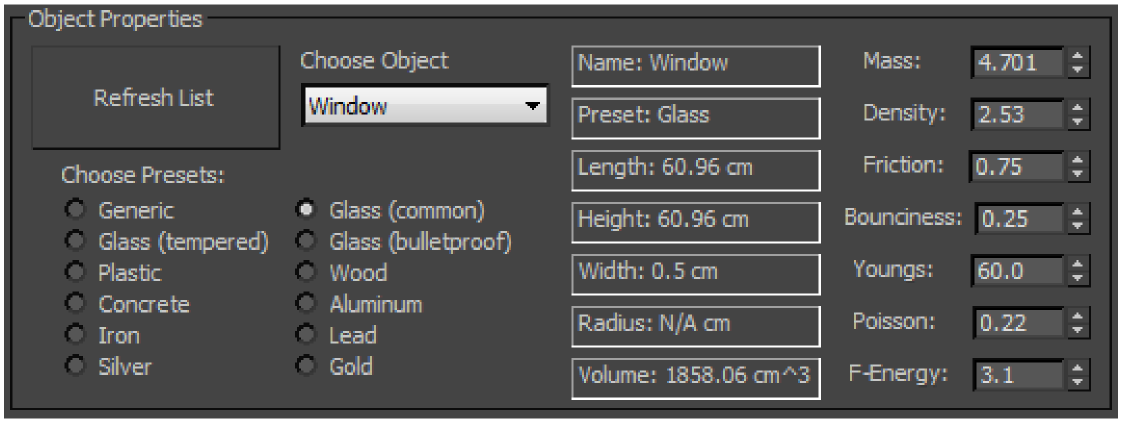
Figure 43: Screenshot of the Object Types Group

The object types group is located in the upper left hand corner of the GUI, and is shown in Figure 43. This is the area where the artist can specify which objects and how those objects need to be included in the simulation. There are three groups of objects: static, kinematic, and dynamic. Below these groups are pick buttons the artist can use to choose objects from the scene to add them to the list above, respectively. The static list is designed for simulating objects that are not be affected by gravity or move when impacted by other objects. Instead they act as a collision surface for other objects in the simulation. This list is best for walls, floors, and other non-moving surfaces a moving object may bounce off of. The kinematic object list is designed for objects that have animation key frames set by the artist, but also need to be added to the physics simulated after the final key to be affected by gravity and collisions. This list is best for the objects that impact each other, such as a baseball or rock. The dynamics list is designed for objects that initially are at rest, not affected by gravity, but then become active after a collision occurs. This list is best for the object that is to be fractured, such as glass or concrete.

When the artist uses the pick button to add an object to the list above, that chosen object is given a MassFX rigid body modifier with the list names rigid body type (static, kinematic, or dynamic). Other parameters are additionally given to the object, such as allowing kinematic objects to become dynamic objects after their final key frame and dynamic objects defaulting to a sleep state.

To remove an object from a list, double click on its name in the list. This also deletes the MassFX modifier applied to the object. If a list becomes too large from many objects being added, a scroll bar appears. If the user attempts to add an object to a different list while it currently exists in another, the previous modifier is deleted and then the correct modifier is added.

III.2.5. Object Properties


Object Properties

Refresh List

Choose Object: Window

Choose Presets:

○ Generic	[✓ Glass (common)]
○ Glass (tempered)	○ Glass (bulletproof)
○ Plastic	○ Wood
○ Concrete	○ Aluminum
○ Iron	○ Lead
○ Silver	○ Gold

Name: Window		Mass:	4.701
Preset: Glass		Density:	2.53
Length: 60.96 cm		Friction:	0.75
Height: 60.96 cm		Bounciness:	0.25
Width: 0.5 cm		Youngs:	60.0
Radius: N/A cm		Poisson:	0.22
Volume: 1858.06 cm^3		F-Energy:	3.1

Figure 44: Screenshot of the Object Properties Group

After adding objects to the lists in the “Object Types” group, the user can proceed into the Object Properties group, as shown in Figure 44, and click the button labeled “Refresh List”. This populates the “Choose Object” drop down list with all of the object names from all three of the list boxes in the “Object Types” group. The user may then select one from the drop down list and begin editing its properties for the simulation and fracture calculations.

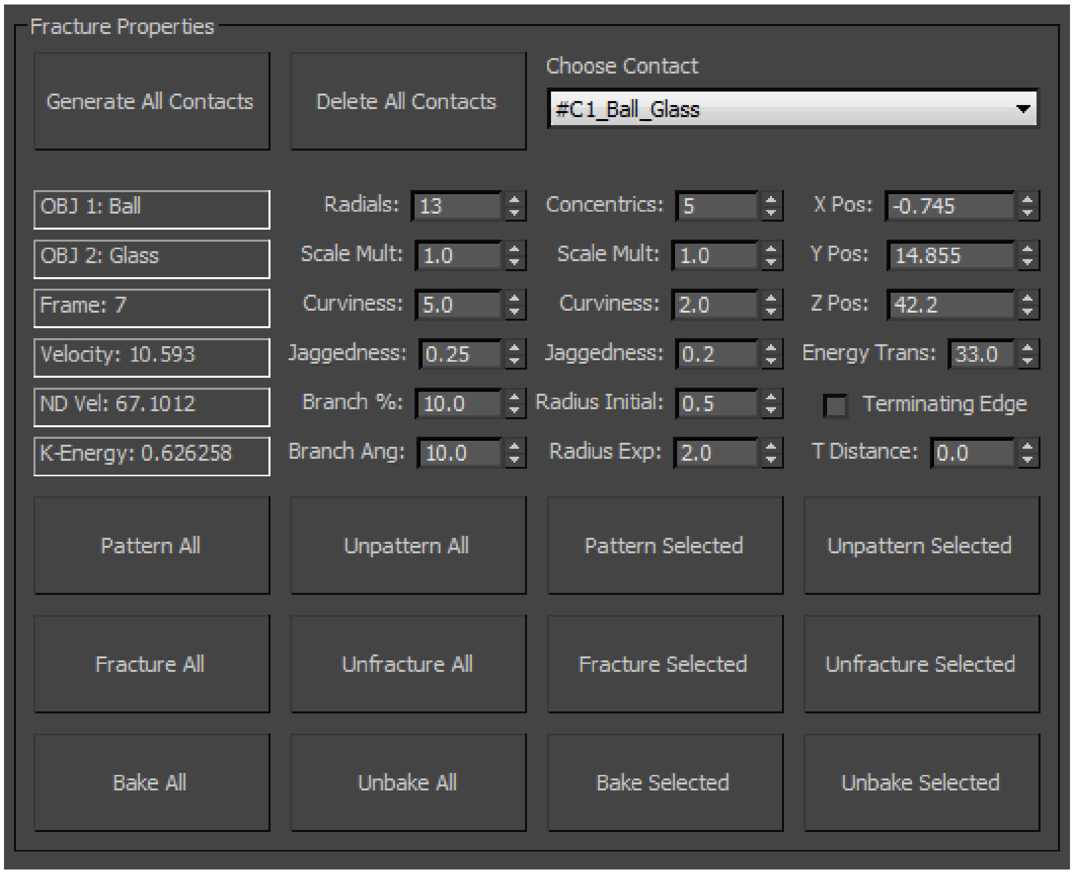
The first column to the right of the drop down list contains non-editable properties defined by the creation of the object, such as the name and volume. This column is useful for ensuring that the user is editing the correct object and that the system has the correct dimensions of the modeled object.

The second column contains editable properties that affect the simulation and fracture results. The mass and density spinners are synchronized with the MassFX modifier on the object. Altering the value of mass or density automatically changes the other value. The unit for the mass spinner is kg, while the density spinner is g/cm^3. The friction and bounciness spinners are also synchronized with the MassFX modifier on the object. Friction changes the MassFX modifier values for static and dynamic friction simultaneously, while bounciness changes the corresponding MassFX value for that object. The user may still change the friction values independently through the modify panel. The last three spinners labeled “Youngs”, “Poisson” and “F-Energy” correspond to the Young’s modulus, Poisson’s ratio, and fracture energy of the object. These values are not found in the MassFX system, and later are used in the calculation of the fracture pattern for glass.

Often, an artist may not know the approximate density or Young’s modulus of a material. With this in mind, the bottom left corner of this group contains a radio button list of presets. When selected on an object on the drop down list, choosing a preset fills in the far right column of this section with applicable data from the appropriate literature. If the user wishes to change the preset values for a given preset choice, they may open the script and edit the set preset values.

Once the artist has chosen each object that needs properties changed from the generic default preset, they may continue on to the “Fracture Properties” group. If the artist desires to add an object to the simulation and needs to edit its properties for fracture simulation, they currently must add it into the applicable “Object Type” list, and then hit the “Refresh List” button again. Using this button to refresh the list wipes all saved data from all objects and replaces it with the default generic preset values.

III.2.6. Fracture Properties



Choose Contact: #C1_Ball_Glass

OBJ 1: Ball
OBJ 2: Glass
Frame: 7
Velocity: 10.593
ND Vel: 67.1012
K-Energy: 0.626258

Radials: 13
Scale Mult: 1.0
Curviness: 5.0
Jaggedness: 0.25
Branch %: 10.0
Branch Ang: 10.0

Concentrics: 5
Scale Mult: 1.0
Curviness: 2.0
Jaggedness: 0.2
Radius Initial: 0.5
Radius Exp: 2.0

X Pos: -0.745
Y Pos: 14.855
Z Pos: 42.2
Energy Trans: 33.0
☐ Terminating Edge
T Distance: 0.0

Figure 45: Screenshot of the Fracture Properties Group

The “Fracture Properties” group, as shown in Figure 45, contains all necessary features to find collision points, generate fracture patterns from those points, and break the selected object into multiple objects based on the pattern. Each following section in this chapter covers either a set of buttons or parameters, as opposed to each section covering an entire group as the previous sections.

III.2.7. Collision Locations and Fracture Parameters

The button labeled “Generate Contacts” is used to find the collision points of any objects in the scene that have been added to the drop down list in the Object Properties group. Upon hitting the button, the script turns on MassFX’s use contact report system and steps through a simulation of the scene frame by frame from the first frame in the current animation range until the last frame in the range. On each frame MassFX checks to see if a contact is present. If there is a contact, the system records the two object’s nodes, names, contact position, and calculate the velocity of each object by finding the difference in distance over time by looking at the position of the object in the previous two frames, as calculated by two formulas below.

$$change\ in\ distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

$$velocity = \frac{(change\ in\ distance * current\ frame\ rate)}{100.0}$$

After calculating the velocity of the current collision, a locator is created at the collision point for debugging and artist override. The script then uses the two object's parameters as set in the Object Properties group to calculate the default parameters for the fracture parameters, such as number of radials cracks and crack features such as amount of crack branching. The following formulas are based upon the research by Nicolas Vandenberghe, Romain Vermorel, and Emmanuel Villermaux [39].

$$V\hat{} = nondimensional\ velocity\ = \left(\frac{E * h}{\gamma}\right)^{2/3} * \left(\frac{v}{c}\right)$$

$$c = \ speed\ of\ sound\ in\ the\ material = ({}^{B}/_{\rho})^{1/2}$$

$$B = bulk\ modulus\ of\ elasticity = \ E/(1 - \nu^2)$$

$$number\ of\ radial\ cracks = 1.7 * \left((V\hat{})^{1/2}\right)$$

$$kinetic\ energy = {}^{1}/_{2} * mass * velocity$$

Other calculations for determining starting variables for fracture patterns such as jaggedness include determining the kinetic energy in the collision and utilizing the object's set stiffness. If the artist wants to undo the generated contacts, they may click on the "Delete Contacts" button. This deletes all contact point locaters in the scene and clears the contact drop down list and all labels and spinners.

III.2.8. Editing Fracture Parameters

Now that the script has calculated all of the collision points in the scene, the drop down list labeled “Choose Contact” is now populated with the names of the points. When a point is chosen, all of the below labels and spinners are active and ready to change for that fracture.

In the first column, there are non-editable labels for information and debugging, containing: the names of the two objects involved in the collision, the frame number the collision occurred on, the velocity and non-dimensional velocity of the collision, and kinetic energy of the collision.

The second column contains editable spinners that control all aspects of the radial cracks to be generated. The first spinner, “Radials”, refers to the number of radial cracks generated. The second spinner, “Scale Mult”, refers to the scale multiplier of the radials. The fracture patterns generated are based upon splines with specific vertex locations. The scale multiplier refers to how far apart on average the points are from one another when the next point is generated. The third spinner, “Curviness”, refers to how much angle variation is added during point to point creation. The fourth spinner, “Jaggedness”, refers to how much jitter is added during point to point creation. If the curviness and jaggedness values are 0.0, the radial lines generated are perfectly straight. The fifth spinner, “Branch %”, refers to the chance of a branch occurring from point to point. If the branch percentage value is 0.0, the radial cracks never branch. The sixth spinner, “Branch Angle”, refers to the angle between the two new branches created.

The third column contains editable spinners that control all aspects of the concentric cracks to be generated. The first four spinners, “Concentrics”, “Scale Mult”, “Curviness”, and “Jaggedness” all have the same properties and effect as the radial parameters, except applied to the concentric cracks. The fifth spinner, “Radius Initial”, defined the radius of the smallest and closest concentric ring to the collision point. The sixth spinner, “Radius Exp”, refers to the exponential radius increase from each concentric radius to the next.

The fourth column contains editable spinners that control miscellaneous parameters for the fracture pattern. The first three spinners control the X, Y, and Z position of the impact point. The artist can manually override the position by editing these spinners or by moving the generated locater point in the scene. The fourth spinner, “Force Falloff”, simulates the momentum transferred outward from the collision point in the direction of the impact. A value of 0.0 applies zero force to pieces not directly hit by the object, while a high value applies force that is applied to every piece with diminishing returns for pieces farther away from the impact origin. The next option is a checkbox labeled “Terminating Edge”, that when enabled causes the radial and concentric cracks to terminate at the specified distance by the last spinner, “T Distance” for terminating distance. With the checkbox checked, a distance of 0.0 would cause no fracture pattern to be generated, while a value above 0.0 would cause the pattern to stop generating past that distance directly away from the impact location.

After editing these parameters, the artist can then switch to editing a different collision's parameters by using the drop down list. All of the parameters are saved and are reloaded upon switching back to the previous collision point.

III.2.9. Generating and Deleting Fracture Patterns

To generate a fracture pattern, the artist must first select the contact from the drop down list and then hit the "Generate Pattern" button. The script then records all of the relevant data and parameters and begins drawing the pattern. The pattern is initially draw at the scene origin and is later moved and rotated into its final place at the center of the object to be fractured.

Radial cracks are calculated one spline point at a time until a point is outside of the length or height boundary of the object. Points along the line are calculated based upon the earlier input point distance, jitter, and angle amounts. A branch occurs when randomly based upon the branch percentage value. When a branch does occur, a new spline is created and the branch percentage is lowered by squaring the value. Then the new spline is continued until it reaches the outside of the bounding box. This is scripted by a recursive function that passes the new lowered branch percentage to the next spline, and returns to create the right branch after reaching the end of the object or reaching the terminating distance for the left branch. Concentric cracks are generated by creating and editing n-gons with the appropriate radii, point numbers, and other features. These are separately created, moved, and rotated into place. After all splines have been generated, moved, and rotated into the appropriate location for the specific object and impact location, they are then welded and combined into one pattern node.

To undo the generated fracture patterns and delete all fracture patterns in the entire scene, the user can simply press the “Delete Patterns” button.

III.2.10. Utilizing Fracture Patterns to Fracture Objects

After at least one fracture pattern has been generated, the user may then click the “Fracture Selected” button to fracture the selected object using this pattern. The script saves the original object by hiding it. A copy of the object is recreated by creating a new object with the same parameters, creating additional geometry from the pattern using 3ds Max’s Pro Boolean modifier with the set imprint option, and finally breaking apart individual elements into objects using a modified open-source script from Hacksaw Tools. Each individual object created is given a unique name, but is given a MassFX modifier with the same parameters from the original object, such as rigid body type, density, sleep state, friction, and material. During this creation process, the mouse is set to the wait cursor, and then set back to the arrow cursor after completion, as this process may take a few seconds.

Once an object has been fractured into pieces, the artist may test out the simulation by running a play through simulation in MassFX. If the animation is complete, the artist can bake all simulated animation through the MassFX toolbar.

To undo a fracture, the user may simply click the “Undo Fracture” button. This deletes all newly created pieces, unhide the original object, and reset any effected variables.

III.2.11. Baking the Simulation

After the objects have been fractured and the simulation is ready to be converted and saved into keyframes, the artist can use the “Bake All” button. This is a predefined MassFX feature that steps through the simulation one frame at a time and places a keyframe for each object on each frame. Animation must be baked prior to rendering for the simulation to be rendered. If the artist needs to unbake the animation, they can us the “Unbake All” button. This deletes all keyframes from all baked objects and returns them to the previous step for further simulation.

IV. RESULTS AND DISCUSSION

IV.1. Fracture Patterns

The system as developed is very accurate for many types of fracture, though it sometimes requires a few artist overrides for specific cases. It is very flexible and can accommodate very diverse patterns such as those shown in earlier figures from Shinkai's research. Sometimes concentric patterns can propagate as single smooth round crack or can be quite complicated with irregular branching and jaggedness, as shown in Figure 46. In other cases, concentric cracks are regularly spaced apart, while in other cases they clump together at some terminal distance away from the impact location, as shown in Figure 47.

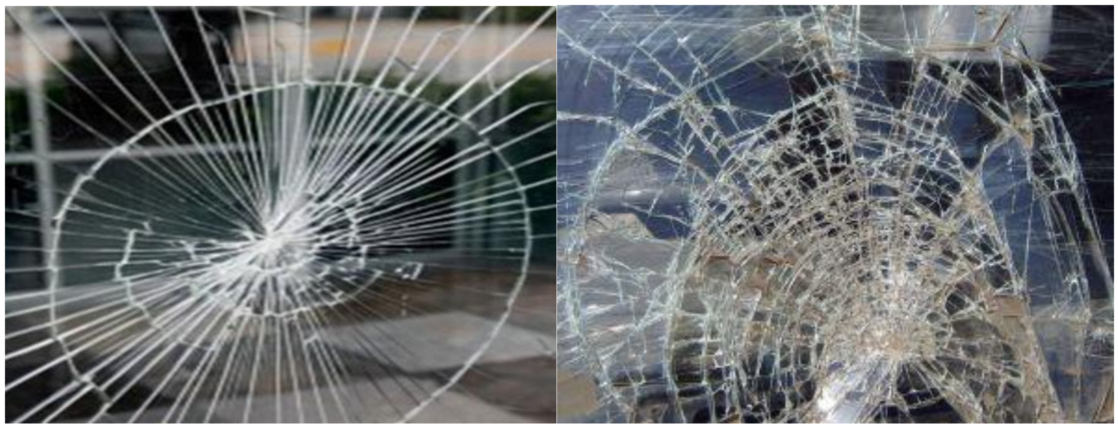


Figure 46: Photographs of Real Glass Fractures Depicting Complexity

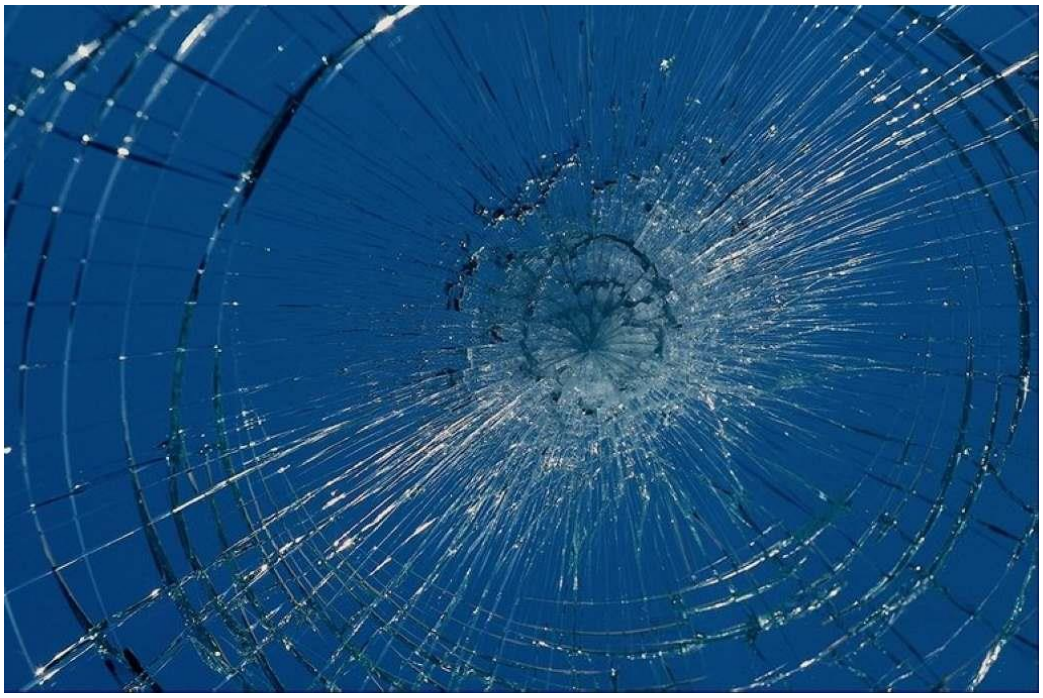


Figure 47: Photograph of Glass Fracture Depicting Concentric Cracks

With specific parameters designed to allow for the creation of these types of patterns, the solution is powerful and versatile. Examples of some possible generated patterns are shown in Figure 48, and an example replication of a real fracture and virtual fracture is shown in Figures 49 and 50. If the user desires a different pattern, or feels that a specific pattern is not believable, despite the properties the user chose for the object, the user can easily generate new altered patterns before proceeding fracturing the virtual glass.

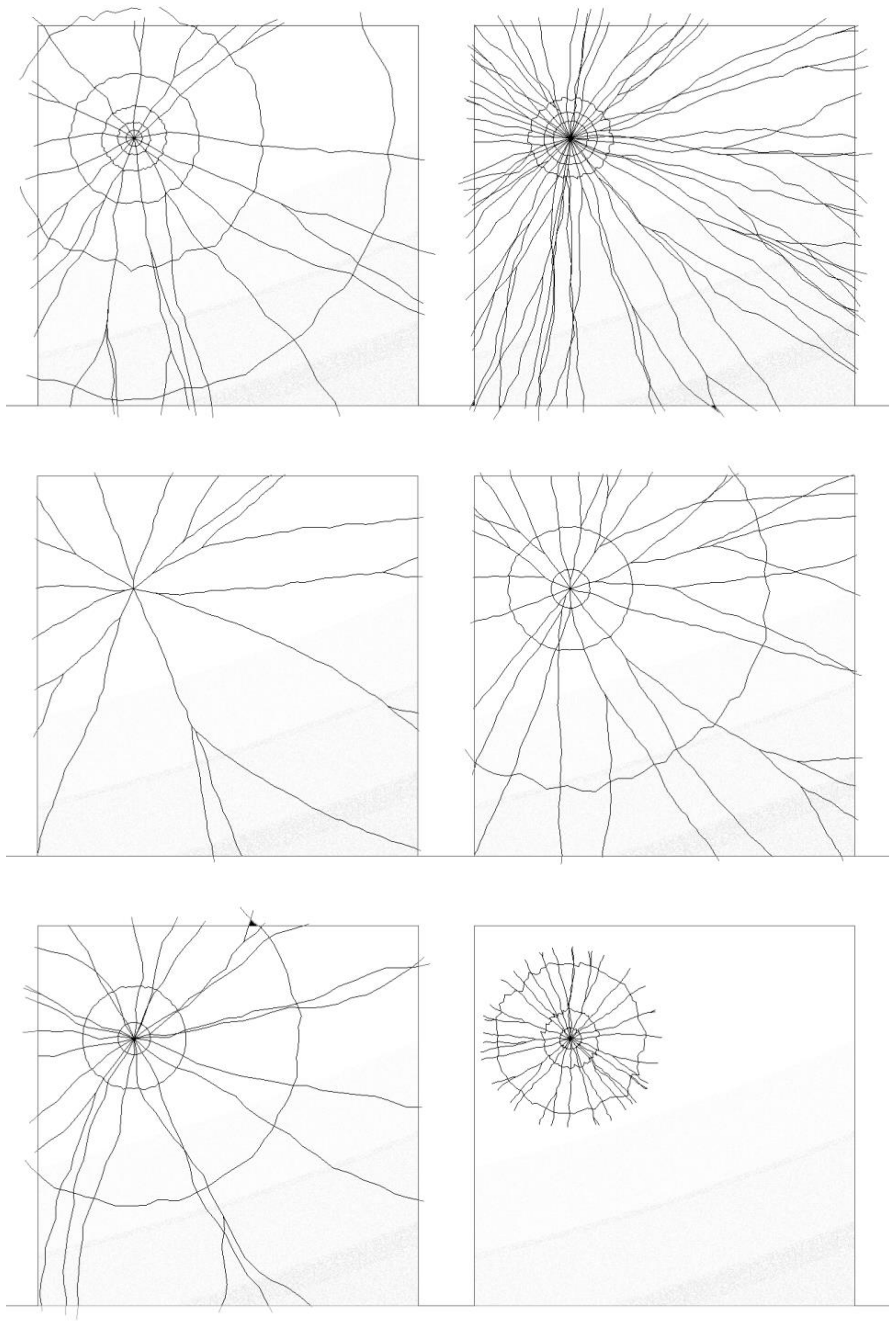


Figure 48: Screenshot of Example Possible Generated Fracture Patterns

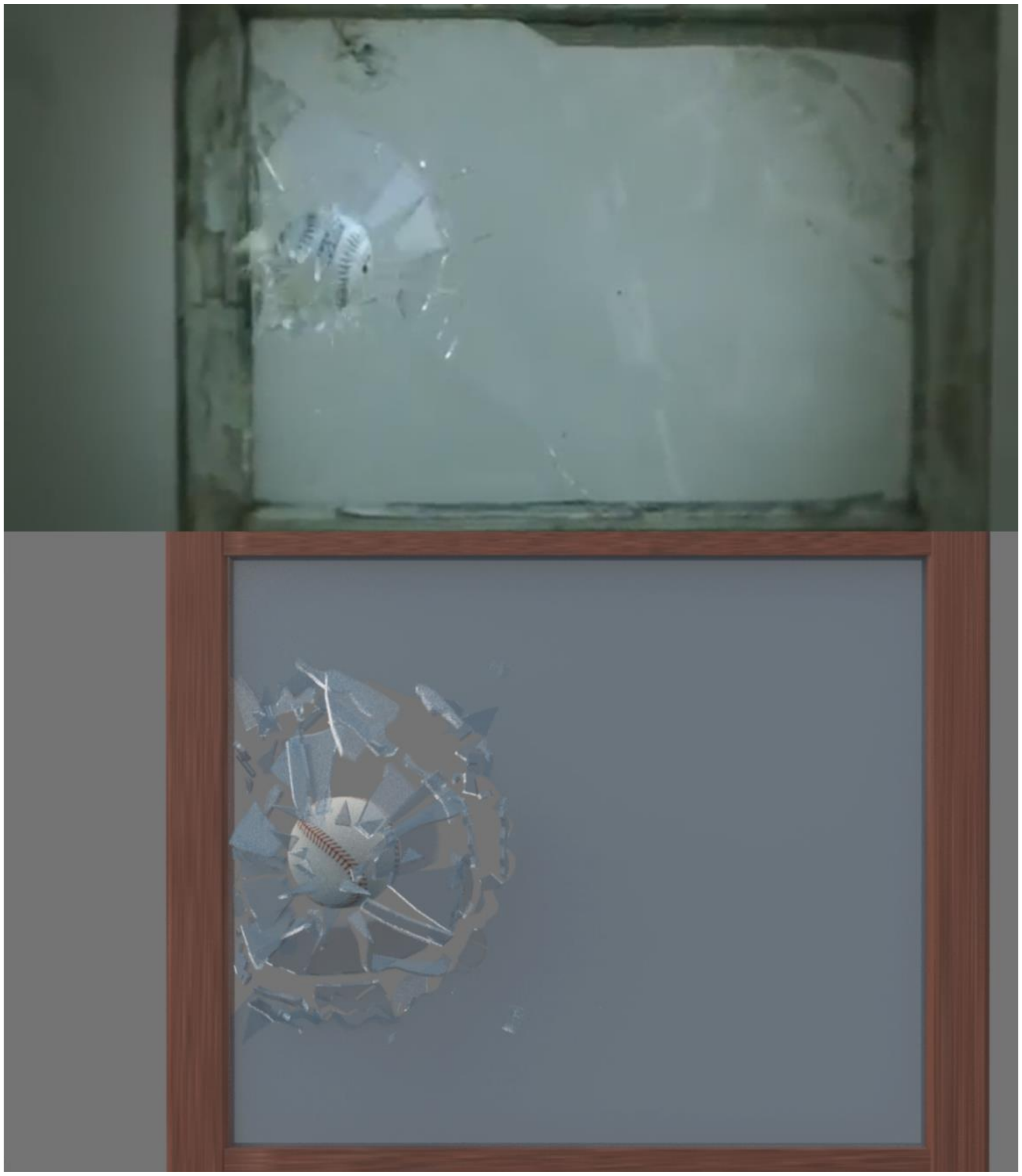


Figure 49: Real Baseball Fracture (top) Compared to Presented Solution (bottom)

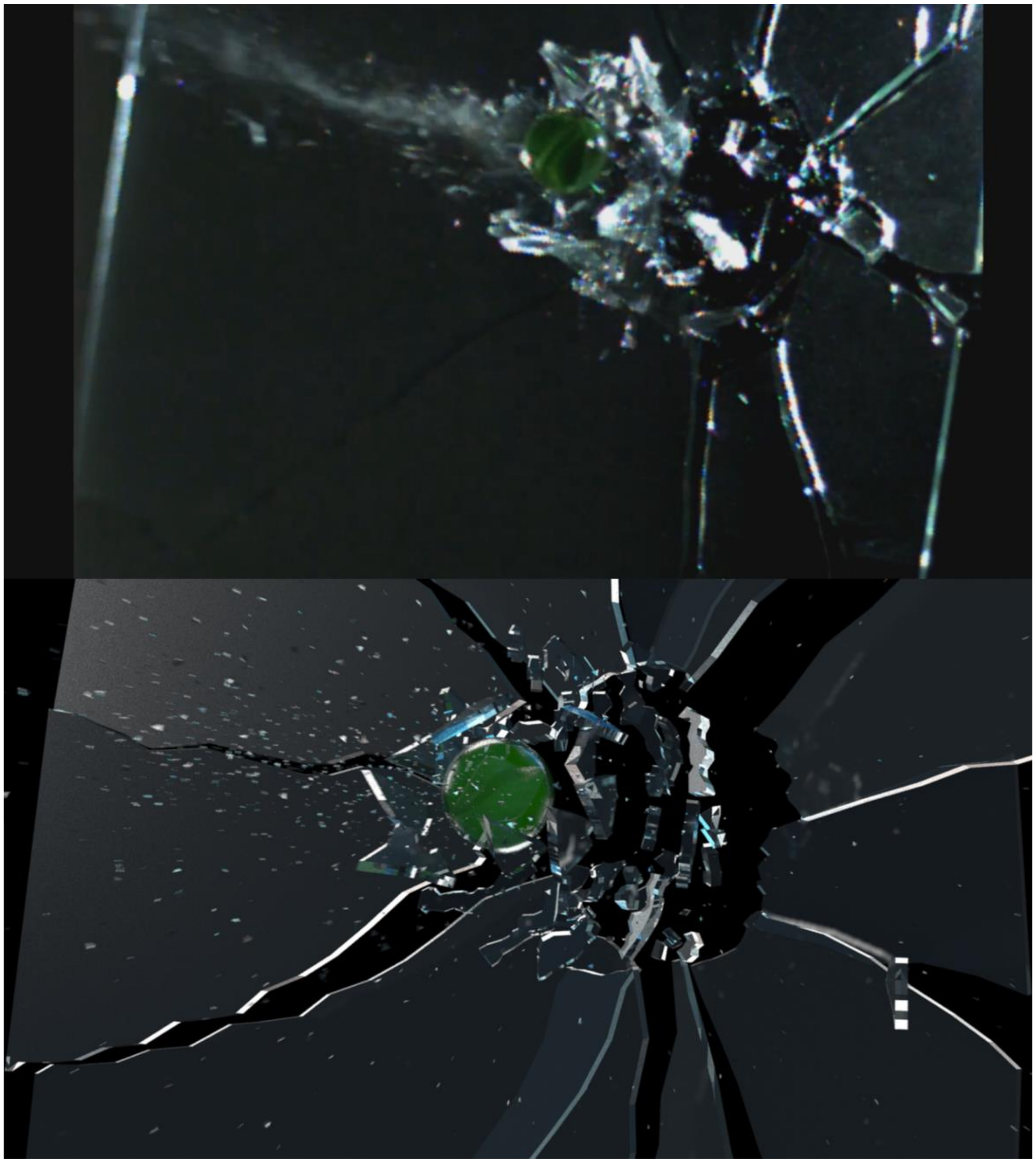


Figure 50: Real Marble Fracture (top) Compared to Presented Solution (bottom)

IV.2. Physical Simulation

The accuracy of the physical simulation of the glass fractures presented in this thesis and the accompanied video renders compared to real world glass fractures depends largely upon the physics engine used and the artist accurately supplying the engine with real world data and believable animation if the objects are not completely simulated. The presented solution uses the MassFX engine built in to 3ds Max, but could use any physics engine for simulating the object fragments after impact fracture.

IV.3. Art Directability

The presented solution offers direct control of the number of radial and concentric cracks that commonly occur in glass fracture, as well as control over features about them such as the amount that they curve. As mentioned earlier, many other solutions do not offer a direct control to the number of cracks, such as FEA which only allows object property manipulation, the Voronoi algorithm which only allows control over seed points which gives some control, or texture mapping which gives full control but requires unique drawn maps. Given this level of control, the presented solution offers a relatively high level of art directability for achieving specific fractures from an object collision.

IV.4. Computational Performance

As stated earlier, the system used to develop and utilize this script solution was a home computer with an Intel i7-860 processor, NVidia GTX 460 graphics card, and 16 GB of RAM. Computationally, the evaluation of the script is very fast. Detecting collisions requires checking each individual frame. In a simple 3ds Max scene, the system used was able to check over 120 frames in 3 seconds. Generating a complete pattern of vertices and splines takes less than 1 second. Patterns that have over 100 radial cracks and 100 concentric cracks may take a few additional seconds to generate. The time to fracture a specified object depends upon the pattern generated. A detailed pattern of 20 radial and 10 concentric cracks each with varying jaggedness and curviness parameters on a single object computes in less than 5 seconds, which generates approximately 200 unique objects each with their own collision meshes. A highly detailed pattern with 50 radial and 25 concentric cracks can take up to 1 minute, as this generates approximately 1500 unique objects. An incredibly detailed pattern with 100 radial and 50 concentric cracks can take up to 20 minutes as this generates over 6000 objects, but this is not advised for simulation as a single particle generator could fill the task. Undoing a pattern or fracture on any object takes less than 1 second to perform. With a generation time of often less than 1 minute per object for a highly detailed pattern and fracture, the system is very computationally fast.

V. SUMMARY AND CONCLUSIONS

The aim of this thesis was to create a method and tool to quickly simulate realistic and believable fracture patterns in glass and integrate them into an existing software package to save artist's time. With all of the controls and quickly modifiable parameters, this system achieves this goal. However, the script could have additional parameters and features added as well as customization for specific uses. Some examples of useful features that could be added would be support for blast waves or direct manipulation of collision points rather than only supporting and relying on object collision. If the artist desired an explosion or other force to break the glass, they would need to still use some form of object based collision in the current solution. The user interface could also be improved upon or customized to be more intuitive to new users. For performance, if the script was integrated as a preloaded plugin, the computational duration of fracturing objects may be reduced. As more information or research from materials science literature is created or utilized, patterns can be improved with additional support for more types of glass, types of cracks, types of impacts, and increasingly accurate materials science research on fragment shapes and sizes.

REFERENCES

[1] American Society for Testing and Materials, "ASTM C1678-09," Standard Practice for Fractographic Analysis of Fracture Mirror Sizes in Ceramics and Glasses, West Conshohocken, PA, ASTM International, 2009.

[2] U. S. Army, Field Manual No. 19-20, 1985. Available: http://library.enlisted.info/field-manuals/series-2/FM19_20/CH10.PDF [Feb 21, 2014].

[3] F. E. Barstow and H. E. Edgerton, "Glass-Fracture Velocity," Journal of the American Ceramic Society, vol. 22, pp. 302-307, 2006.

[4] J. D. Blacic, "Mechanical Properties of Lunar Materials Under Anhydrous, Hard Vacuum Conditions: Applications of Lunar Glass Structural Components," Lunar Bases and Space Activities of the 21st Century, vol. 1, pp. 487-495, 1985.

[5] Blast Code Inc., “A Few Feature Productions.” Available: http://www.blastcode.com [Feb 24, 2014].

[6] Blender Foundation, “Dynamics and Simulation.” Available: http://www.blender.org [Feb 24, 2014].

[7] R. C. Bradt, M. E. Barkey, S. E. Jones, and M. E. Stevenson, "Projectile Impact—A Major Cause for Fracture of Flat Glass," Practical Failure Analysis, vol. 3, pp. 5-11, 2003.

[8] W. D. Callister and D. G. Rethwisch, Materials Science and Engineering: An Introduction, 8 ed., Hoboken, NJ: Wiley & Sons, Inc., 2009.

[9] Cambridge University, “Young's Modulus – Density”, 2011. Available: www-materials.eng.cam.ac.uk/mpsite/interactive_charts/stiffness-density/NS6Chart.html [Mar 3, 2014].

[10] CrimeMuseum, “Glass Analysis.” Available: http://www.crimemuseum.org/crime-library/glass-analysis [Feb 27, 2014].

[11] F. Erdogan, "The Interaction between Inclusions and Cracks," Concepts, Flaws, and Fractography, New York. NY: Springer US, pp. 245-267, 1974.

[12] Fracture FX, “About Fracture FX.” Available: http://www.fracture-fx.com [Mar 3, 2014].

[13] GlassProperties, “Calculation of the Elastic Modulus of Glasses.” Available: http://www.glassproperties.com/young_modulus [Jan 7, 2014].

[14] GlassProperties, “Calculation of the Poisson’s Ratio of Glasses.” Available: http://glassproperties.com/poisson_ratio [Jan 7, 2014]

[15] A. A. Griffith, "The Phenomena of Rupture and Flow in Solids," The Phenomena of Rupture and Flow in Solids, Royal Society of London Philosophical Transactions Series A, vol. 221, pp. 163-198, 1921.

[16] J. Hellrung, A. Selle, A. Shek, E. Sifakis, and J. Teran, "Geometric Fracture Modeling in Bolt," ACM SIGGRAPH 2009: Talks, pp. 1, 2009.

[17] H. P. Kirchner and J. Conway, "Fracture Mechanics of Crack Branching in Ceramics," Fractography of Glasses and Ceramics, Westerville, OH, vol. 22, pp. 187-213, 1988.

[18] J. L. Ladner, D. L. Ahearn, and R. C. Bradt, "Laminate Design Effects on the Fracture Patterns of Impact Resistance Glass Panels," Fractography of Glasses and Ceramics VI: Ceramic Transactions, vol. 230, pp. 281-287, 2012.

[19] J. Locke and J. A. Unikowski, "Breaking of Flat Glass—Part 1: Size And Distribution of Particles from Plain Glass Windows," Forensic Science International, vol. 51, pp. 251-262, 1991.

[20] J. Locke and J. A. Unikowski, "Breaking of Flat Glass—Part 2: Effect of Pane Parameters on Particle Distribution," Forensic Science International, vol. 56, pp. 95-106, 1992.

[21] B. B. Mandelbrot, The Fractal Geometry of Nature, New York, NY: W. H. Freeman and Company, 1983.

[22] J. J. Mecholsky, T. J. Mackin, and D. E. Passoja, "Self-Similar Crack Propagation in Brittle Materials," Fractography of Glasses and Ceramics Westerville, Ohio, 1988, vol. 22, pp. 127-134, 1988.

[23] M. Müller, N. Chentanez, and T. Y. Kim, “Real Time Dynamic Fracture with Volumetric Approximate Convex Decompositions,” ACM Transactions on Graphics 32, 4, 2013.

[24] U. S. Navy, “Fractures Made with a Blunt Instrument or Object,” 1994. Available: http://www.tpub.com/maa/190.htm [Feb 21, 2014].

[25] J. F. O'Brien and J. K. Hodgins, "Graphical Modeling and Animation of Brittle Fracture," Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 137-146, 1999.

[26] Paramount Pictures, “Mission: Impossible – Ghost Protocol”, 2011.

[27] Pixelux Entertainment, “About Pixelux.” Available: http://www.pixelux.com [Mar 3, 2014].

[28] F. Preston, "A Study of the Rupture of Glass," J. Soc. Glass Technology, vol. 10, pp. 234-269, 1926.

[29] S. Raghavachary, "Fracture Generation on Polygonal Meshes Using Voronoi Polygons," ACM SIGGRAPH 2002 Conference Abstracts and Applications, pp. 187-187, 2002.

[30] S. Raghavachary and M. Matthews, "3000+ Variations of the Voronoi Diagram," ACM SIGGRAPH 2007, p. 136, 2007.

[31] RayFire Studios, “Glass Cracking.” Available: http://www.rayfirestudios.com [Mar 3, 2014].

[32] R. W. Rice, "Perspective on Fractography," Fractography of Glasses and Ceramics Westerville, OH, vol. 22, pp. 3-56, 1988.

[33] Scientific Working Group for Materials Analysis, "Glass Fractures," United States of America, 2005.

[34] E. Shand, "New Information on Glass Fracture," Proceedings of American Scientific Glass Blowers Society Symposium, pp. 117-128, 1969.

[35] N. Shinkai, "The Fracture and Fractography of Flat Glass," Fractography of Glass, New York. NY: Springer, pp. 253-297, 1994.

[36] A. Smekal, "The Nature of the Mechanical Strength of Glass," J. Soc. Glass Technology, vol. 20, pp. 432, 1936.

[37] J. E. Stanworth, Physical Properties of Glass, vol. 6, Clarendon Press, 1950.

[38] Thinkinetic, “PullDownIt.” Available: http://www.pulldownit.com [Mar 3, 2014].

[39] N. Vandenberghe, R. Vermorel, and E. Villermaux, "Star-Shaped Crack Pattern of Broken Windows," Physical Review Letters, vol. 110, pp. 174302, 2013.

[40] WaveEquation, “Surf Wax Hardness.” Available: http://www.waveequation.com/surf_wax_hardness.htm [Feb 21, 2014].

[41] WaveMetrics, “Image Interpolation,” 2004. Available: http://www.wavemetrics.com/products/igorpro/imageprocessing/imagetransforms/imageinterpolation.htm [Feb 21, 2014].

[42] R. Weissmann, “Relation of Edge Strength of Flat Glass to Fracture Mirror Size.” Available: http://www.glass-ceramics.uni-erlangen.de/Research/Glass/Project_g1.htm [Jan 7, 2014].

[43] Wikipedia Contributors, “Brittle v Ductile Stress-Strain Behavior,” 2010. Available: http://en.wikipedia.org/wiki/Brittleness [Feb 21, 2014].

[44] Wikipedia Contributors, “Silica,” 2004. Available: http://en.wikipedia.org/wiki/Glass [Feb 21, 2014].

Supplemental Sources Consulted

[45] T. Allen, J. Locke, and J. Scranage, "Breaking of Flat Glass—Part 4: Size and Distribution of Fragments from Vehicle Windscreens," Forensic Science International, vol. 93, pp. 209-218, 1998.

[46] L. Glondu, M. Marchal, G. Dumont, "Real-Time Simulation of Brittle Fracture Using Modal Analysis," IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 2, pp. 201-209, 2013.

[47] J. Heo, J. Seong, D. Kim, M. A. Otaduy, J. Hong, M. Tang, et al., "FASTCD: Fracturing-Aware Stable Collision Detection," in Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 149-158, 2010.

[48] H. N. Iben and J. F. O'Brien, "Generating Surface Crack Patterns," Graphical Models, vol. 71, pp. 198-208, 2009.

[49] H. Kirchner and R. Gruver, "Fracture Mirrors in Polycrystalline Ceramics and Glass," Concepts, Flaws, and Fractography, New York. NY: Springer, pp. 309-321, 1974.

[50] F. Lange, Origin and Use of Fracture Mechanics, New York. NY: Springer, 1974.

[51] J. Locke and J. K. Scranage, "Breaking of Flat Glass—Part 3: Surface Particles from Windows," Forensic Science International, vol. 57, pp. 73-80, 1992.

[52] J. Mecholsky, R. Rice, and S. Freiman, "Prediction of Fracture Energy and Flaw Size in Glasses from Measurements of Mirror Size," Journal of the American Ceramic Society, vol. 57, pp. 440-443, 1974.

[53] K. Museth and M. Clive, "CrackTastic: Fast 3D Fragmentation in The Mummy: Tomb of the Dragon Emperor," ACM SIGGRAPH 2008 Talks, pp. 61, 2008.

[54] J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins, "Graphical Modeling and Animation of Ductile Fracture," ACM Transactions on Graphics, vol. 21, pp. 291-294, 2002.

[55] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. J. Guibas, "Meshless Animation of Fracturing Solids," ACM Transactions on Graphics, pp. 957-964, 2005.

[56] G. D. Quinn, Fractography of Ceramics and Glasses: US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2007.

[57] S. C. Schvartzman and M. A. Otaduy, “Fracture Animation Based on High-Dimensional Voronoi Diagrams,” Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, pp. 15-22, 2014.

[58] J. Smith, A. Witkin, and D. Baraff, "Fast and Controllable Simulation of the Shattering of Brittle Objects," in Computer Graphics Forum, 2001, pp. 81-91.

[59] C. Zheng and D. L. James, "Rigid-Body Fracture Sound with Precomputed Soundbanks," ACM Transactions on Graphics, vol. 29, p. 69, 2010.