Diffuse-limited Aggregation created fire path and spread prediction

Sireethorn Satarabhandhu

I7911966

Msc Computer Animation and Visual Effects

Master Project

NCCA Bournemouth University

August 19th, 2011

CONTENTS

Page

1 INTRODUCTION

This project is about generating the prototype of fire path and predicting the fire path growth rate by concerning with surrounding fire area. The fire path will be created by the diffuse-limited aggregation (DLA) structure. The random walked algorithm of diffuse-limited aggregation (DLA) has been used for generating and calculating the fire spread. Furthermore, the factors of fire growth will be considered while the speed of fire is calculated. This program has been written in openGL and C++ language to simulate the prototype of fire path in 3D scene.

The main element for combustion would be considered to generate the prediction. Only flammable object (fuel object) can catch the fire. Therefore, the program should know whether the fire path going along with flammable or non-flammable objects. Moreover, the temperature, wind speed, and humidity also influence the spread of fire. Thus, program must calculate the fire path with these factors, which can be adjusted by users to see the different fire behavior.

Before the program has been implemented, further details of fire and diffuse-limited aggregation will be explained in the following section.

2 PREVIOUS WORK

2.1 Fire Definition

Fire is chemical reaction of the combination of three elements, which are fuel, oxygen, and heat. These three elements are needed for combustion and the continuing burning. Moreover, the products of combustion consist of heat, light, smoke, and fire gases (Hatton 2004). The Combustion chemical equation is as follows (Anon. 2003).

$$(C_6H_{12}O_5)_n + O2 + \text{Kindling Temperature} \rightarrow CO_2 + H_2O + \text{Heat}$$

2.1.1 Principles of fire

2.1.1.1 Fire Triangle

Fuel: Fuel is the flammable objects, such as papers, gas, woods, and etc. For fuel in forest, fuel is provided from nature, such as Peat Soil, Coal Seam, dead grass, dead leaves, dead branches, twigs, logs, fresh leaves, bushes, and big trees (Emily et al. 2000).

Oxygen: Oxygen is the main element of air, which also supports the combustion of fire. Thus, oxygen is everywhere and would spread all over the fire area. However, the amount of oxygen can be decreased from the burning stage and oxygen in particular position can be altered to different place from speed and direction of wind.

Heat: Heat is the energy that starts the burning process (Hatton 2004). There are two categories of heat sources, which are heat from nature and heat from human. The examples of natural heat source are lightning strikes (thunderbolt), friction of branches (or twigs) to combust and ignite the surrounding area, or explosion of volcano (Anon. 2011).

2.1.1.2 Chain reaction

Chain reaction: The additional component of fire. This is in the Tetrahedron. The fire triangle has to have enough to be able to sustain a chain reaction. The chemical chain reaction

can be defined to two types, which are inhibited and uninhibited chain. Inhibited chain reaction would stop or extinguish the fire within the particular fire flames. However, uninhibited chain reaction would continually sustain the fire burning within the particular fire flames (Hatton 2004).

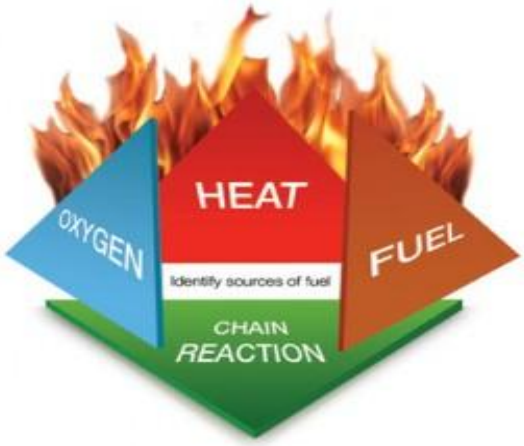


Figure 1: the principles of fire combustion consists of fire triangle and chain reaction (Little 2011)

2.2 Phases of fire

Fire would start ignition on one spot before it start spreading to the surrounding object (or fuel) by going along the heat transfer direction and fire intensity. The Firefighter Close Calls Company (2004) has explained the fire behavior, fire spread and the heat transfer in their article. Moreover, the company has illustrated the phases of fire as shown in figure 3 (Anon. 2004). After the three main elements of fire are not enough for combustion, fire would stop burning and left the burnt down object with smoke behind.

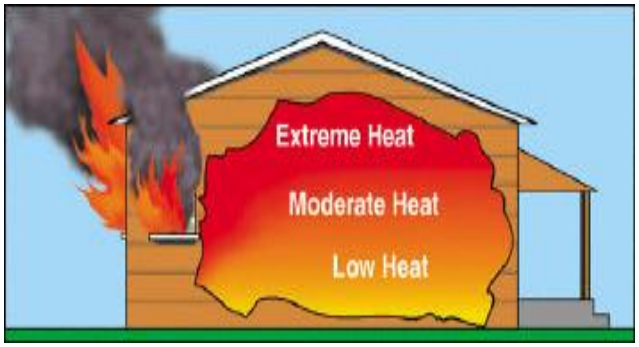


Figure 2: Layers of heat (Anon. 2004)

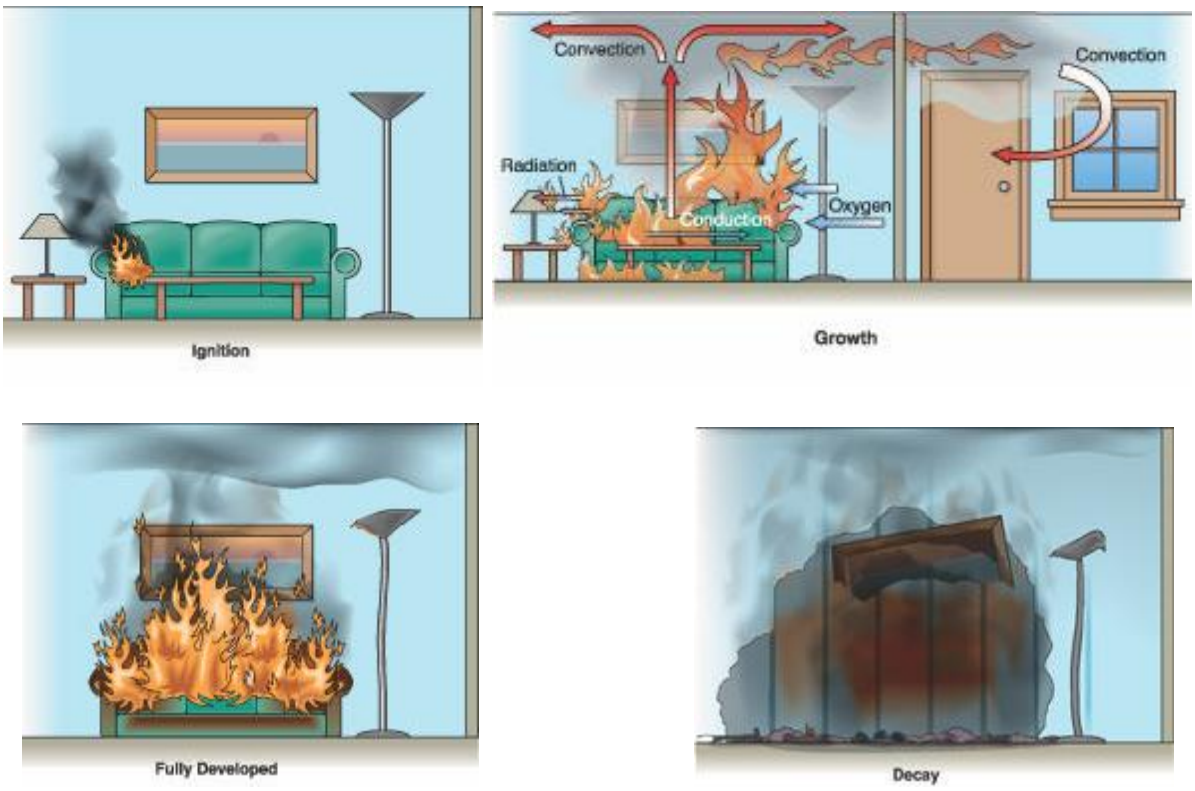


Figure 3: Phases of fire starts with ignition, growth, fully developed and decay stages. (Anon. 2004)

The area closed to the ground would have the lowest temperature, while the highest area has the highest temperature. Moreover, Firefighter Close Calls Company (2004) has illustrated heat layer as shown in figure 2.

2.3 Forest fire

2.3.1 Forest fire behavior

For the normal expansion, fire would travel in flat landscape, without any winds, and the fuel constantly scattered around the area. Fire will spread to every direction over the fuel area in the equally speed, which it makes the fire growth in the shape of circle that keeps getting bigger until they reach the area where is no fuel. Thus, the centre of circle is where fire starts.

However, forest usually has steep or slope landscape, which it makes the amount and the expansion of fuel unequally in different area. Moreover, when the fire starts, heat in those areas would be increased and float up to above the bonfire, in the higher atmosphere. Then, the colder air around the bonfire would flow to replace the heat's old area. This is known as the circle of winds in forest fire. Therefore, the forest fire would not spread in the shape of round circle. They would spread in the shape of oval instead because the fire burn to different direction and with the different fuel amount. Furthermore, the action of wind and the slope of landscape affect the fire growth. The strong winds would drag the flame to their direction, same as landscape, the flame would travel towards the uphill slope because the heated air draws the fire flames upward (Farmer 2009).

2.4 Diffuse-limited aggregation (DLA)

2.4.1 Definitions of DLA

Diffuse-limited aggregation is the way to perform organic forms or fractal growth, such as vein circuit, snowflake, and form of lightning.

Bourke (2004) has stated in his article that particles will form DLA structure from travelling randomly around ("Diffusion") before attaching themselves ("Aggregation") to the structure (Bourke 2004). He also explained the definition of diffusion-limited that particles are considered to be in low concentrations and the structure would grow from one particle at a time (Bourke 2004).

Figure 4: Form of lightning, which looks like DLA structure (Tubtub 2006)

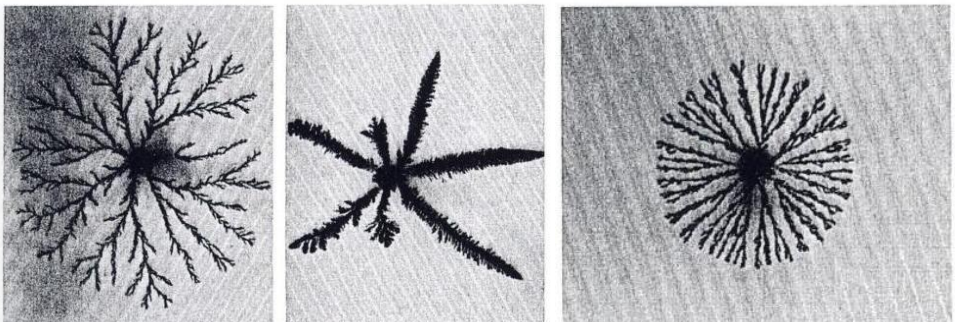


Figure 5: Examples of zinc deposit that look like DLA fractals (Pietronero 1989)

2.4.2 **DLA Process**

Reas and McWilliams (2011) have explained the simple rules of DLA process. Particles move through space with random direction. Furthermore, when the random walks of particles approach or collide with the origin or DLA structure, the particles will stick with the structure. DLA form would be built up from one particle at a time, which more particles would collide and cluster together (aggregation) over the time period.

The structure begins with an origin or a fixed seed. Afterwards, new particles would be created and start moving through space (Reas and McWillams 2011). At each step, direction of moving particles is random and the particle moves within a short distance. Every step of particle's movement, it will check with surrounding area that has collided with the fixed seed or not. If yes, the particle will stop and become another fixed seed on the growing form.

3 TECHNICAL BACKGROUND

3.1 Fire spread prediction

Wind speed, wind direction, slope landscape, fuel moisture and moisture of burning area are factors that influence the speed rate of fire spread. Moreover, the measurement of fire behavior, such as fire intensity, rate of growth, and fire length, are the best thing to use for predicting the burning path (Kennard 2008).

Fire needs three elements (fire triangle), which are fuel, heat, and oxygen, for combustion. Furthermore, the fire can spread, depends on ambient fuel, fire area landscape (slope), weather, and many variables around the burning area. The flux between flammable and un-flammable fuel can be used to predict the path by knowing which object can catch the fire and which object cannot. At the same time, heat transfer is involved in the combustion to react with the surrounding elements, such as fuel object, and air, and can measure the flame length and fire intensity of bonfire (Morais 2001). Morais (2001) has explained about the prediction that there are two strategies exist for predicting the path of fire, which are considering the fire elements itself and considering other factors around the fire. Before the fire spread will be predicted, the measurement of fire behavior, factors of affecting fire behavior, and fire ignitions type should be considered.

3.1.1 Fire measurement

Rate of growth

The rate is ratio of heat of un-ignited fuel and ignited fuel per amount of time, e.g. m/s (Morais 2001). Morais (2001) also showed the equation of rate of fire spread is shown as follow.

$$R = \frac{\sum_{m=1}^{u} q_m}{\sum_{n=1}^{v} Q_n}$$

The rate can be measured from heat, which is received by un-ignited fuel around fire, $\boldsymbol{q}$ (J/s-m^2), over the heat, which is required to ignite the fuel (or combustion) at the leading edge of fire, $\boldsymbol{Q}$ (J/m^3). The total heat $\boldsymbol{q}$ is equal to the total individual $\boldsymbol{u}$ energy flows received from heat

transfer and the total heat ***Q*** is equal to the total heat required to bring the individual ***v*** components of fuel bed from surrounding temperature to ignition temperature (Morais 2001).

Fire intensity

Fire intensity is the measurement of emitting heat from the burning spot or head fire (during combustion) per amount of time (Putnam and Karels 2004).

It is measured form heat released per meter of fire front (kW/m), which can be calculated from the multiplication of heat yield of fuel (J/g), amount of fuel per unit area (kg/m^2) and rate of growth (m/sec) (Wurm et al. 2011). Wurm, Instone and Parr (2011) have stated that the more fuel available for burning would make the faster rate of growth and greater fire intensity (Madrzykowski 2011).

Flame length

Flame length is the distance or height of flame from the base or ground to centre of fire flame.

3.1.2 Factors of fire growth

3.1.2.1 Fuel characteristics

Fuel size and shape: fuel is the main element for combustion, if the size of fuel is big, the combustion on this fuel would be slower than the combustion on smaller size of fuel. Fuel size and shape determines the ability of individual fuels to carry and sustain a fire. The different fuel size and shape can carry and sustain the different amount of fire. For example, fire on small and flat fuel, such as dead leaves, small branches, twigs, and dead glasses, would travel quicker than fire on large fuel, such as trees, grubs, stools, big branches, big twigs, and logs. The larger fuel would need higher temperature (or hotter heat) to ignite (Anon. 2004).

Fuel loading (Quantity of fuel): quantity of fuel that is available to be and can be measured as weight, mass per specific unit of fire area (hectare) or ton of dry fuel per acre (Putnam and Karels 2004). It would affect the intensity of fire, more quantity of fuel, stronger burning flame. Moreover, the stronger flame would emit heat (hotter) to surrounding area.

Fuel compactness: fuel would be accumulated in layers (NYCEMT86 2007) and compacted to be the thicker fuel at particular area. The thickness of fuel also affects the length and intensity of flame. The thicker fuel makes the fire flame longer and stronger. However, if the fuel is very compacted until oxygen cannot get through the compacted area, the fuel would be harder to catch fire. Thus, the fire growth would be slower where there is no oxygen at the area.

Arrangement of fuel for continually burning: The fuel is scattered all over the forest. If the position of each fuel is continually connected to each other or Direct Flame Contact, fire could travel very quickly and continuously. However, if the distance between each fuel is too far to reach from the burning spot, the fire growth would be slower, unless other factors, such as wind, or heat convection, became the factor of fire growth (NYCEMT86 2007). Furthermore, fire would travel vertically quicker than horizontally (Hatton 2004).

Humidity of fuel: the dry fuel can be set the fire easier than the wet fuel. Thus, humidity of each fuel is concerned for forest fire spread behavior. Jenkins (2008) has analyzed the moisture of fuel with the speed of fire and concludes the results as graph shown in figure 6.

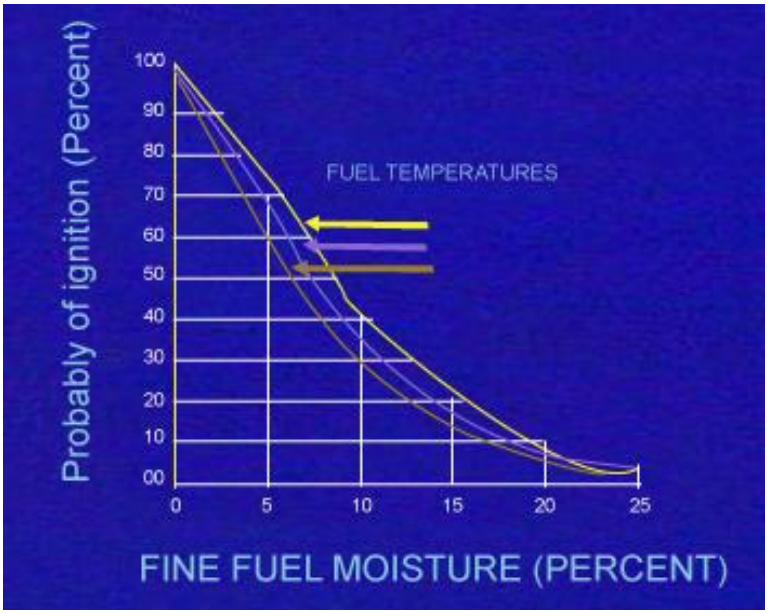


Figure 6: Graph explains how the fire growth effects from fuel moisture (Jenkins 2008)

3.1.2.2 Weather of burning area

Humidity of burning area: The humidity of the area directly changes the humidity of fuel. If the area has the high humidity, fuel in that area also has the high humidity. Thus, it would be harder to catch the fire or, if the fire is set, fire growth would be slower in this area.

- *Relative humidity's rules of thumb* (Bergemann 2011)

1. Every 20 °C decrease, relative humidity would increase in double amount. Hence, every 20 °C increase, relative humidity would decrease with a factor of 2 (half of the latest amount)
2. Relative humidity would be the highest in early morning (dawn) and would be the lowest in the afternoon.

Temperature: Temperature affects the humidity of fuel. The higher temperature causes the lower fuel humidity. The hotter temperature would cause the drier fuel. Thus, fuel would be easier to be burnt.

Wind: Wind has the major affected to forest fire. Wind increases the oxygen amount to forest fire area. Fuel can be dried very quickly. Therefore, the speed of fire growth also increases. Moreover, wind would blow out the ember to get spot fire in another area, which spot fire could start the fire in widespread area (Anon. 2011). Moreover, wind is the big factor to determine the direction of fire spread and its speed.

3.1.2.3 Landscape (topography) of burning area

Slope: Fire would travel quicker in the slope (steepness) landscape than the flat landscape. Moreover, the speed of fire growth along the slope landscape would be affected by wind. If the forest fire spread on the downhill slope, the wind blows against the direction of head fire, thus, the speed of fire growth would be decreased. Moreover, fire would travel faster in the uphill direction because whenever the fire starts, heat will travel upward in the air. Therefore, the fire flame would catch the hotter position of forest (Farmer 2009). Figure 7 and 8 illustrate the fire growth effected by slope area (Jenkins 2008).

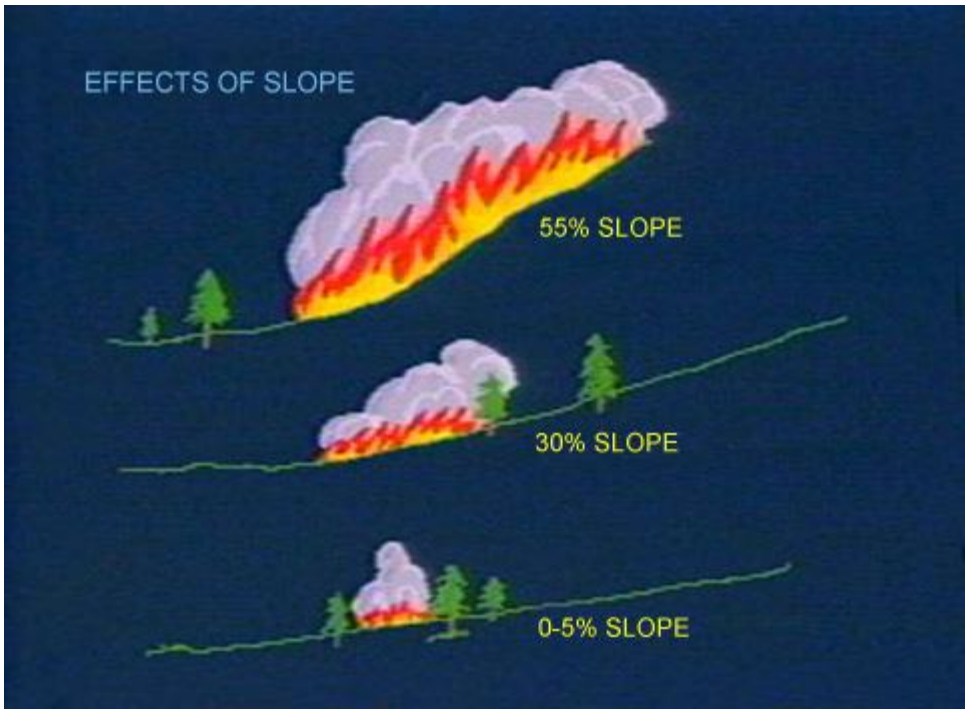


Figure 7: Different fire growth from different slope (Jenkins 2008)

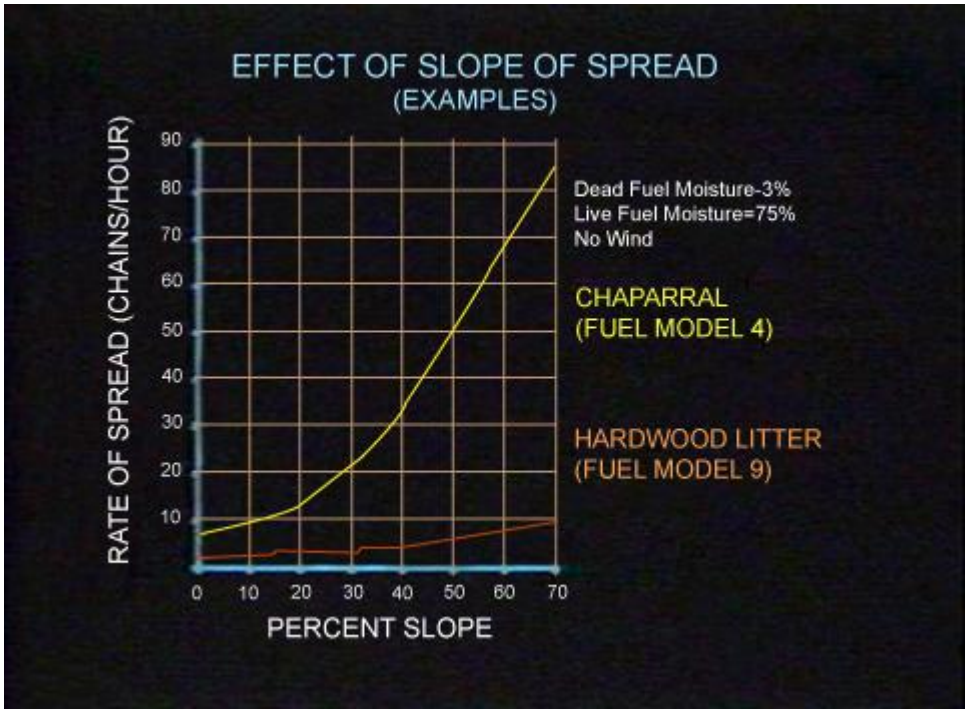


Figure 8: Graph shows the effect of slope with fire spread rate (Jenkins 2008)

Aspect of landscape: Landscape, which faces to sunshine, would receive the heat more than the other size. Thus, those areas would have drier fuel, which is easier to catch fire. Moreover, the slope area that faces to the North would hold more humidity, while the slope that faces to the South would be the drier fuel area (Hanson 2011).

Barriers: Barriers is the object that could stop the fire or decelerate the spread of fire (Anon. 2001).

3.2 Diffuse-limited aggregation (DLA)

Particle would be released from the surface of distance sphere, which has origin point at the centre (Sottile 2010). Afterwards, the particle will randomly walk to the origin or seed point within the step size, which the randomly walk (diffusing particles) can be performed by Brownian motion like (Bourke 2004).These released particles can be called as 'traveler'. The radius of sphere, where particles (or travelers) originate, can be adjusted, depends on the growing size of aggregates and the accomplishment of travelers reached the DLA structure (Sottile 2010).

Number of random walk iteration would be limited by the maximum step size or generations. When traveler gets approached to the origin point or other point on DLA structure, the traveler will be freeze and stick to that point to form the new branch or point of the aggregation. If the traveler has not reached or approached to any part of DLA growing form within the limited step size, this traveler will be killed and new particle (new traveler) will be released out from random position on sphere's surface. The whole steps of DLA forming process will be repeat until it gets the satisfied DLA size. Moreover, distance between origin seed and the new frozen particle can be adjusted to perform different images of DLA.

Lam (2010) has written the Java-applet performing the generation of DLA structure, which each step of growth has been showed slowly. Therefore, the picture can be captured while his program is running (Lam 2010). The following figures from Lam (2010) program show the steps of forming DLA structure by beginning with single origin at the centre and the first released travelers from surface of sphere until it took the random walked to reach the centre point or other part of the structure.

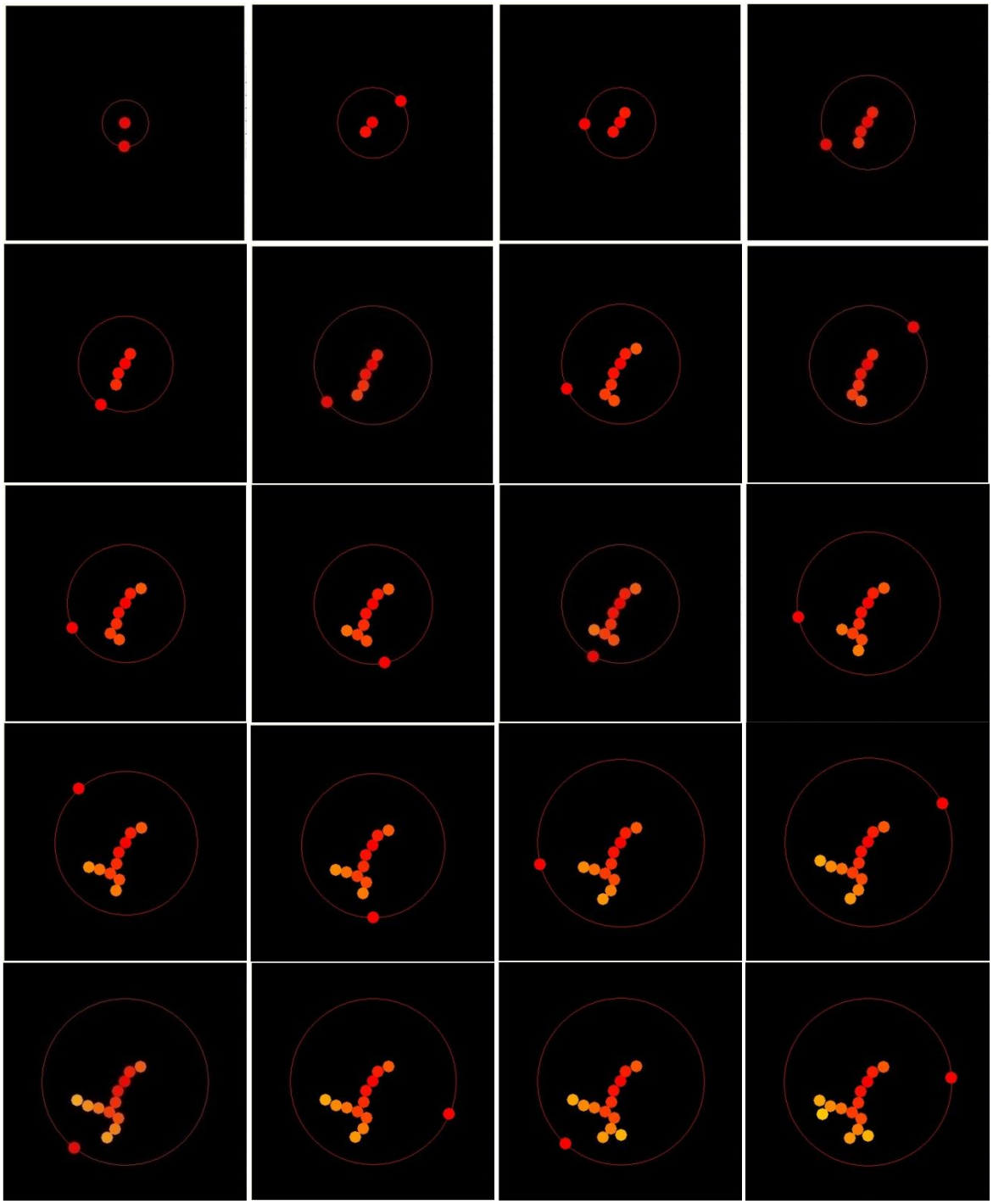


Figure 9: Steps of forming DLA structure, captured from Lam's program (Lam 2010)

Stock (2009) has explained the methods of forming DLA that each point on DLA structure, including origin point can be applied with the stickiness value. This value will give

difference probability to that position for travelers to stick with (Stock 2009). If the point has less-sticky value, the traveler that approaches to that area would not be freeze. However, it will bounce back or move away from that point with the certain distance, which this push-back distance will be assigned to each point on DLA structure itself to response to the approaching travelers (Stock 2009). Moreover, the traveler then takes more random walk to go to stick with other point of DLA structure that has satisfied stickiness value. The probability can identify the length between frozen traveler and origin. If the probability is high, it means that the new growth structure will be located closed to the origin point (or centre of the structure) (Xia and Unger 2008).

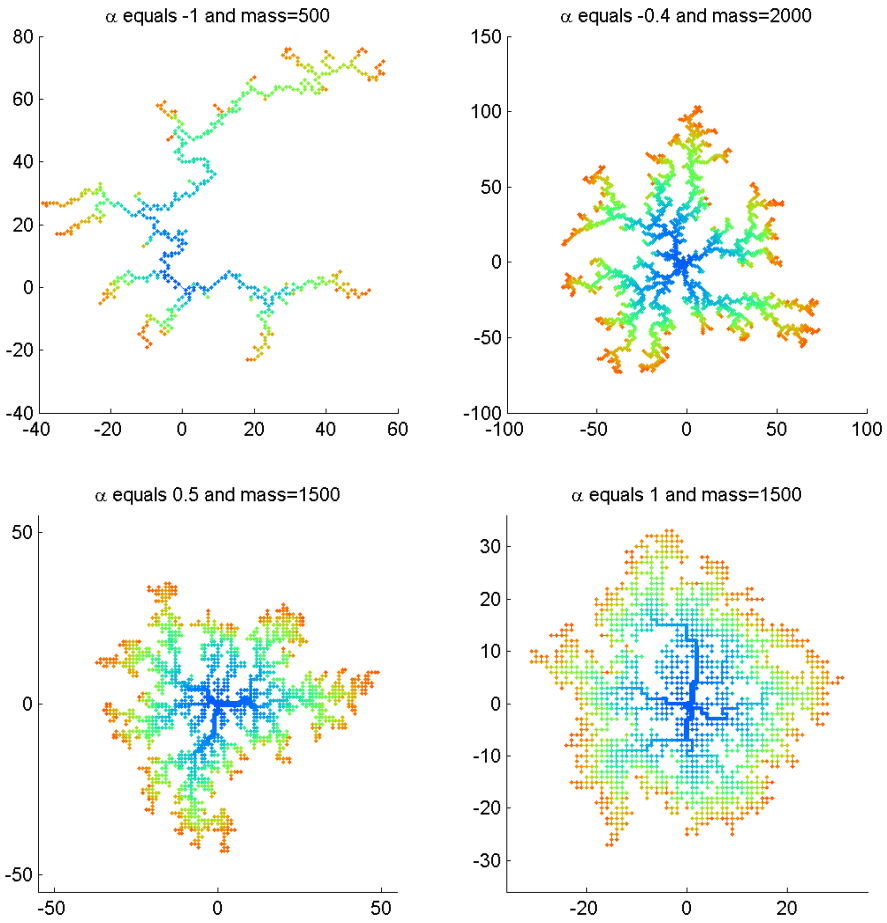


Figure 10: These four graph in this figure show the different DLA structure that has different probability (α) of stickiness (Xia and Unger 2008).

4 BASIC IDEA

Fire path can be predicted by concerning with its fire principal. The combustion would transfer the heat to surrounding area around bonfire. Those heats combining with oxygen in the air and fuel of object are the main part of fire spread. In this project, each object in the fire area should be assigned the fire variables to react with the fire spread prediction. Apart from fire principal, other variables, such as wind, moisture in the air, pattern of landscape and etc, should also be considered to effect with the fire.

When all the fire requirements are met, the fuel can catch the fire and the fire starts to travel to any direction. The direction of head fire would randomly go to any flammable object (or fuel) around the fire starter point. Therefore, DLA has been considered to take part of forming randomly fire direction for the prediction. The program should have the structure of DLA that has origin or centre point as the fire starter point.

4.1 Project Overview

4.1.1 Fire variables

According to the study of fire behavior, fire must have the following variables and methods for predicting the fire spread.

Fuel amount: This variable would tell how much the fuel is load in each object, which it can be used to calculate the fire intensity, rate of growth, and flame length.

Fuel moisture: This variable would identify the ability of each object to catch the fire.

Object type: This variable would identify each object that is flammable or not.

Object shape: This variable would tell the shape of fuel object.

4.1.2 Area of fire

Information about the fire area should be considered for predicting the fire path. Thus, the following variables and methods are important for program to know.

Moisture of the area: This variable would tell the moisture of the fire area, which it can use for predicting the possibility of fire over the area.

Area temperature: This variable would indicate the temperature (or heat) of the fire area.

Wind force: This variable would give the direction and intensity of wind in the fire area.

Slope of landscape: This variable would indicate how height of slope on the fire area, which it can indicate the appearance of landscape. The slope can be assigned to make the different fire spread from flat landscape.

Boundary or size of fire area: This variable would tell the edge or boundary of fire spread. The size can be given as width and height value of the landscape.

Fire starter position: This variable would tell where the fire starts.

4.1.3 Diffuse-limited aggregation variables

The program should consider the following variables and methods to form DLA structure on the fire area. This structure will be used for the pattern of fire spread before other factor is added.

DLA size: This variable would tell how big the structure would be. The size can be considered from the amount of points on the structure.

DLA direction: This variable would indicate where DLA should head to, which it has to be considered from the wind force over the fire area.

DLA starter point: This variable is where the fire starter point is. Thus, it can be at the centre or any side in the fire area.

Distance between DLA points: This variable would identify the maximum distance between DLA points or clusters. The size of DLA (how big) can be different by adjusting this distance.

Radius of sphere: This variable would tell the radius of sphere that is used for releasing the random walker to travel inside the structure's area. Thus, new DLA point or new cluster can be generated.

Maximum iterations: This variable would limit number of walking step or iterations of walker before it hits the point on the structure. If it takes too long to find the structure, the current traveler will be killed and new traveler will be released instead.

Location of released traveler: This variable would tell where the traveler is located after it has been released from surface on the sphere. Thus, the traveler can notice where to go before walking to random direction.

DLA Collision checking method: This method would stick the traveler to the DLA structure and convert the traveler to be the new DLA point. The collision checking will check if the travel has traveled approach (within the given distance) to any location of the structure. Moreover, it can also check if the new point has collided with other previous point or not.

4.1.4 Fire path prediction

To predict the fire spread, the following variables and methods should be considered.

Area size: This variable would tell the boundary of fire spread.

Fire burning time: This method would be calculated from the fire variables to tell how long the fuel can be burnt and how long to take it to be burnt down.

Fire direction: This variable can be found from calculating the fire direction, according to the wind direction, fire intensity, the pattern of DLA structure and other factor, such as slope of landscape.

Speed of fire: This variable is the same as rate of growth measurement that is explained in previous section, which amount of fuel and time will be concerned.

Object Collision checking method: This method would check if the DLA structure (or randomly fire flux) has collided with any object of the fire area or not. If yes, the program

should identify the type of object and also the burning ability of the object. Moreover, if the object is non-flammable, the program should avoid those object area and find the new path's direction and position.

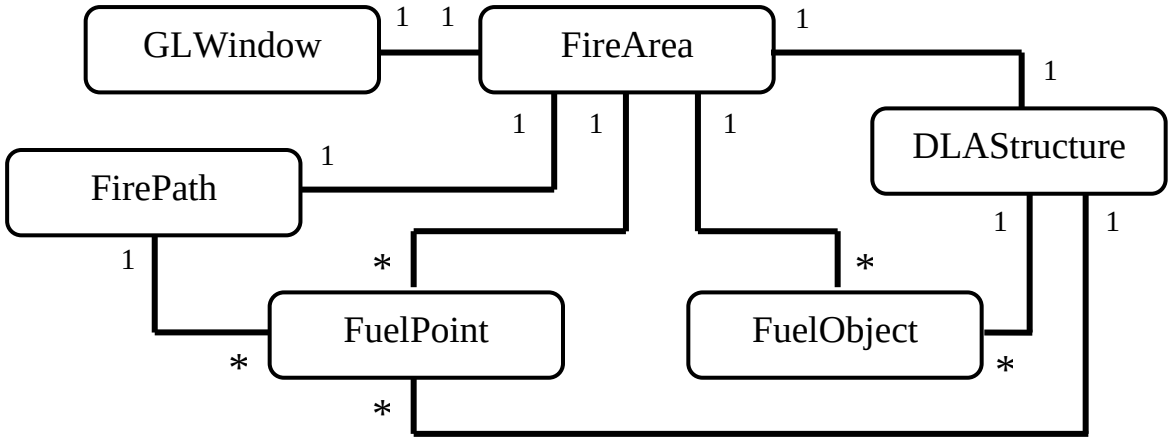
Boundary Collision checking method: This method would check if the fire path or DLA structure has formed outside the provided area or not. This method should control the available location for fire to burn, according to the given area size.

4.2 Class diagram

Program begins with considering the fire area and positioning the fuel object on the area. When the area (or scene) has been created, fire starter point can be indicated. The starter point is decided to be randomly chosen or fixed at the centre. Thus, the implementation of forming DLA can be tested. The design of flowing variables in each method is explained in the following class diagram.

4.2.1 Class diagram

Please see Appendix A. for full class diagram with classes and members details.



5 IMPLEMENTATION

Program is first implemented with the simple 3D scene, filled with the random points that represent as fuel. Afterwards, the fire variables, such as fuel amount, burning status, and moisture value, are assigned to those points for a later prediction part. After the fire area is created, program can be noticed the edge or boundary of the area. Thus, DLA structure now can be implemented within the area and specific size. Moreover, those DLA structure would be built for representing the fire path, which the function of checking collision, scene landscape, weather, fuel object and predicting fire behavior are concerned. Further information about steps to implement the program would be described later in this section.

5.1 DLA implementation

DLA has to be created from the large set of particles or points. Therefore, the size of DLA can be identified by limiting the number of points that is available to construct the DLA structure. The structure is created from one particle at a time. The particle can be called as 'traveler' after it is released from the sphere's surface and before it hits any part of recent DLA form. Moreover, the origin point of this structure can be located at the centre or be in any part of the scene, depends on its assigned. The steps of implementing DLA form are as follows.

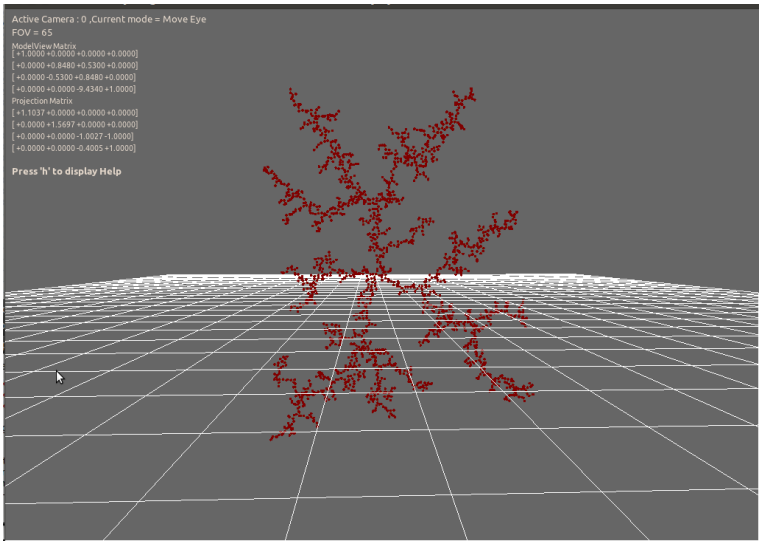


Figure 11: the first implemented of DLA structure

5.1.1 **Random function of travellers**

The sphere that is considered here is used for releasing the travelers to travel inside the area. This sphere would have the origin point of DLA structure at the centre. Moreover, the radius of the sphere would be used to calculate the random position of where traveler is born. The radius is adjusted each time the traveler is released. However, it is between the initial radius value and the maximum as three times bigger of initial radius. The size of radius would be random picked from simple random function (rand() in C++ language), which can specify the limited range numbers of random area. Random points on a sphere can be found by using the function below (Pseudo-code), which the program expects the random function to give the point that locates in XZ-plane. The following equations have been used to find the released position of traveller, which they have been modified from *Diffuse Limited Aggregation Rendered with Sunflow* project of Reavis (2010).

$XYRadius = sqrt(radius^2 - Z^2)$

XYThetaValue = rand() between PI*2 and zero

X = current position + [cos(XYThetaValue)*XYRadius]

Z = current position + [sin(XYThetaValue)*XYRadius]

Y = current position + [rand() between (-radius) and (radius)]

New released traveler's position is vector (this.X, this.Y, this.Z)

Furthermore, the equations have developed to control the direction of DLA form. The sphere can be adjusted by limiting the size of radius and setting the new centre point to create the structure. This feature would generate the DLA structure with the satisfied direction. The position and section of sphere can indicate the direction of DLA structure or the direction of where the fire is heading to from the effect by wind force. The figures below show how the limitation of sphere is work and how the DLA structure direction goes, the green points represents as the sphere section and the dark red point represents the created points of DLA form.

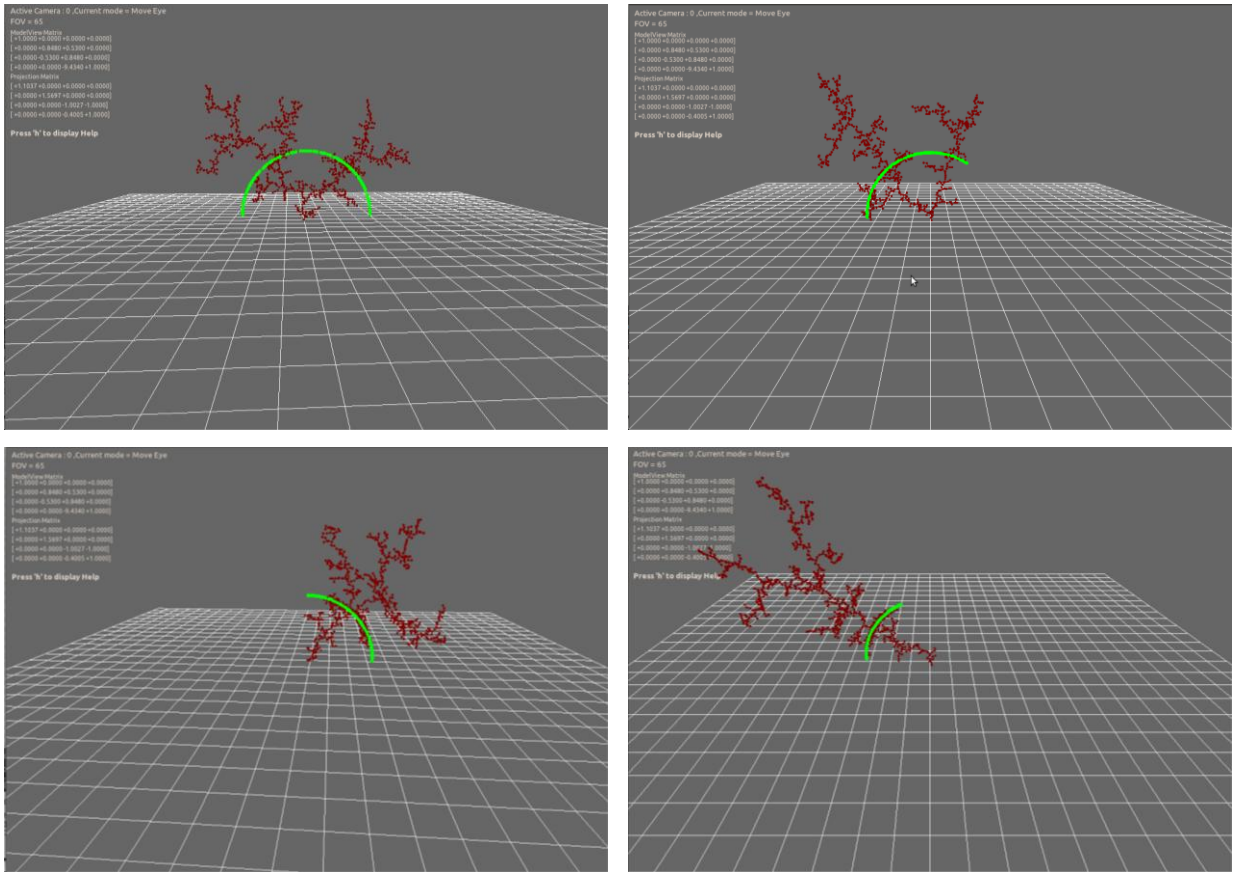


Figure 12: The different direction of DLA structure created by adjusting the sphere variables

Traveller would travel over the area by randomly moving up, down, left, or right at a time, until it reaches the satisfied distance away from any points of DLA structure. However, the moving iteration of traveller is being limited within the declared value of maximum iteration number.

5.1.2 **Boundary and DLA structure collision detection**

After the size of fire area has declared before the random walk of traveler is generated, program has to check the collision of the traveller's movement and its boundary. Thus, whenever the traveller moves, it needs to check if it collides with any side or edge of fire area. Figure 13 shows how boundary collision detection creates the DLA form.

Apart from checking collision with boundary, a traveler also needs to check if it collides with the previous points on DLA. Thus, those points would not overlap to each other and the

form would come out fine. However, the distance to check the approaching of traveler and the structure's point can be adjusted to generated different size of DLA structure. Every time the travelers approaches to any DLA point within the satisfied distance area, those travelers will be frost and become new point of DLA structure.

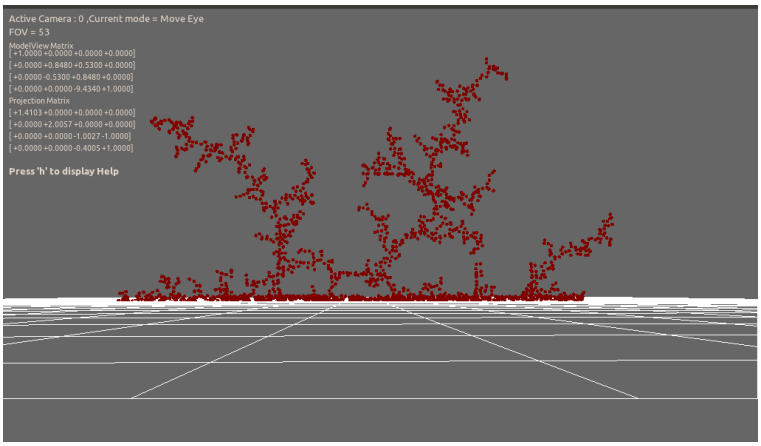


Figure 13: DLA structure with boundary detection

5.1.3 **DLA branch generation arrangement**

When the new point of DLA structure has been created, it will need to be assigned the appropriate number of generation to tell the chaining position with previous DLA structure. Program would know where the new point would be chained to. This generation arrangement would help program calculate the growth rate of fire path. Furthermore, the generation would help program calculate the spread order of each chain when the wind and slope factors have affected the fire path.

5.1.4 **Fuel collision detection**

Fuels have scattered around the fire area. Those fuels can be categorized as flammable object and nonflammable object. Thus, program can notice where the available place for fire to spread on. The DLA structure, which represents fire path, should not go through the nonflammable area. So, the program has to check if the new DLA point has collided with any location of nonflammable object.

In addition, if the DLA points have collided with the flammable objects within short distance, those objects would be set the fire and start burning until all the fuel has gone. The figures below show the DLA form with the non-flammable object collision detection.

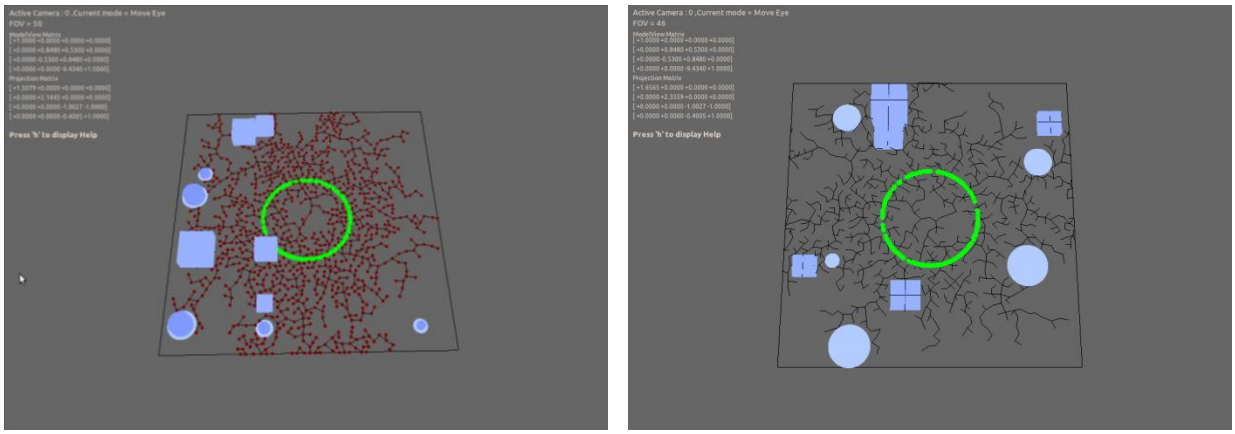


Figure 14: The DLA form with non-flammable object collision detection feature

5.1.5 **Other DLA form collision detection**

Program can draw fire path with different amount of origin up to five origins in one scene. When one origin has been starts the fire and starts to spread over the ground area, other fire path will respectively be set the fire afterwards with having the function to check collision with the area that has been already burning by other DLA form. Therefore, this collision checking function would help program avoid the overlapped burning area or re-burn no fuel ground point.

5.1.6 **Heat emitting**

The heat can be emitted from the burning fuel. Fire path travels over the ground that is flammable, which can be considered as the small fuel. Therefore, when the fire starts, surrounding area should receive the heat emitting and if they area received the large amount of heat, those surrounding area can be ignited. The program has a function to transfer the heat to surrounding area from the current burning point, which is in the burning point list. Thus, the spread will be faster, if there are lots of small fuels surrounding by.

5.2 Fire scene implementation

Fire scene has created with the feature of several factors that would affect the fire path prediction, such as slope area, wind, scene moisture and scene temperature variables. Therefore, user could see the different fire path behavior from the different value of those factors.

5.2.1 Adjusting fire seed amount and position

Program has the function to raise the amount of fire seed to generate more than one fire path in one scene. In addition, those fire seeds can be adjusted the position to the satisfied area.

5.2.2 Terrain

Landscape of fire area can effect to the fire spread. Thus, to create the efficient fire scene, the terrain is needed in this project. Program would randomly pick the centre of terrain point and the terrain will be generated by adjusting the height of the points. Size of circle will be considered to get the surrounding point within short distance, regards to radius of circle. Centre terrain point would be in the highest position while other points would be lower in order. Therefore, terrain will look like semi-circle and have the smooth surface. Moreover, the program allows user to adjust the number and height of terrain. Hence, user could see the different of fire spread over the flat area and slope area.

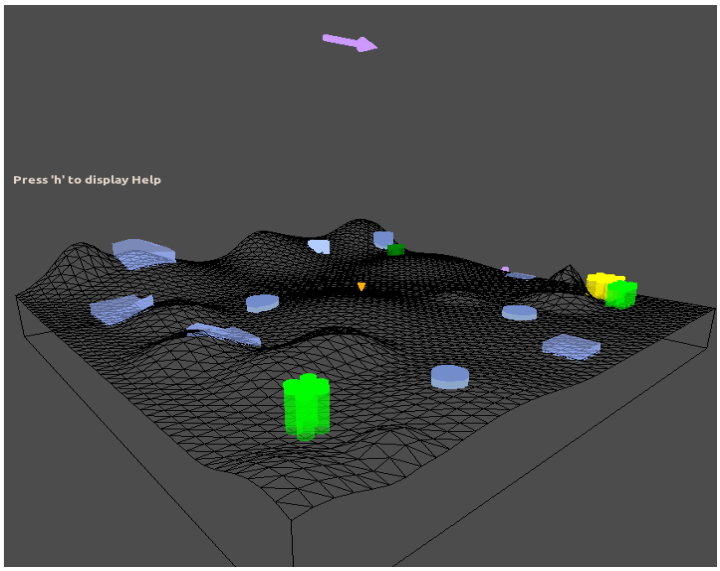


Figure 15: Terrain of the scene

5.2.3 **Wind function**

Wind can be enabling in the fire scene to show the different appearance of fire spread. The fire path direction and speed would be effected from wind. The fire path would spread along the wind direction and speed up the velocity of any fire points that has the same direction with the wind, otherwise, the other fire points that would go against the wind would have slower velocity or slower growth rate. Moreover, the speed of fire points, which has effected from wind, will be considered with speed of wind (or wind intensity) also. If the wind is strong, higher adjusted speed value would be applied, otherwise, the smaller adjusted speed value would be applied to each fire points.

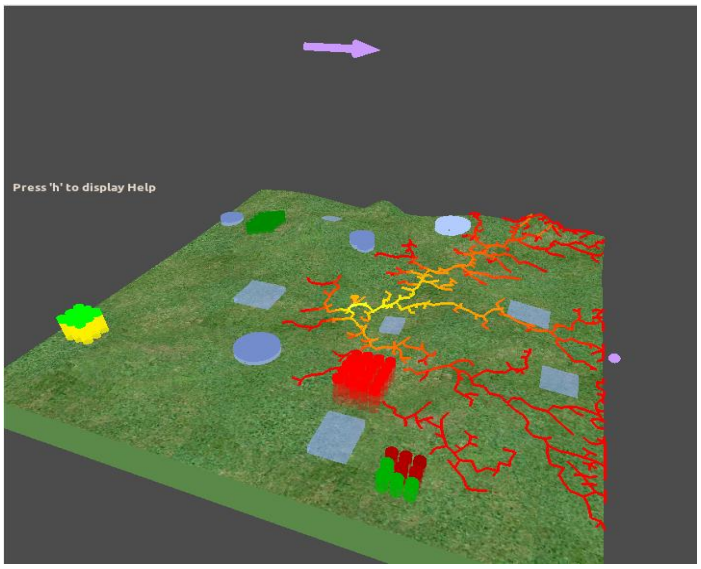


Figure 16: DLA structure effected from wind direction

5.2.4 **Moisture and temperature**

Other factors from fire scene that would affect the fire path are moisture and temperature of the fire scene. For moisture variables, the lower moisture would increase the rate of growth value, while, for temperature variables, the higher temperature would increase the rate of growth. However, both variables also has the effect together, following the '*Relative humidity's rules of thumb*' that has been explained in Berhemann articles (2011), which has stated that every 20 °C increase in the area, the relative humidity would increase in double amount, and vice versa.

Therefore, program will adjust the value of scene moisture by current half amount if the temperature has changed every 20 °C step.

5.2.5 **Flammable and non-flammable object location**

The flammable fuel and non-flammable fuel would be scattered all over the fire scene, respecting to assigned amount of each object type. The ground point would be assigned the status if they locate in one of these object area, thus, program would know which area can burn and which area cannot. Fuel amount of each flammable fuel would be randomly assigned, which it has an effect with the burning time of that flammable fuel, more fuel amount would need more time to burn down the object. Moreover, the fuel moisture and size can also be adjusted to extend or decrease the burning area.

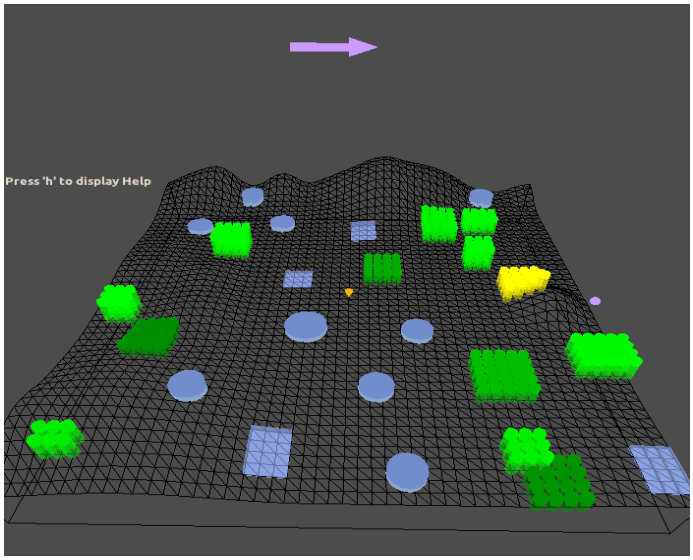


Figure 17: The flammable object (green) and non-flammable object (blue) scattered in the scene

For flammable object, the transferring heat to surrounding flammable object method has been assigned. Therefore, whenever the burning fire path approaches to the fuel object within the short distance, program will set fire at those areas and let that fuel object burn. Moreover, to burn the object, the moisture and height size of fuel object would be considered to create different burning time of particular object. Therefore, each burning object would be burnt down at different time.

If one of the flammable object starts to burn, moisture of the object would be decreased to zero and fuel amount of object would be decreased along the time. However, when the burning fuel has emitted heat and transfer to surrounding fuel object, those surrounding fuel objects will be also ignited if there is enough heat received from the burning fuel.

5.2.6 **Ground heat transferring**

After the ground point has received the heat and set the fire from burning fire path, the ground points itself would also transfer the heat and burn surrounding area if they have enough heat to ignite the surrounding field. However, the ground point would not ignite the surrounding ground area if those places have been already ignited once because it means that all the fuel should be already all burnt.

5.3 Fire path prediction implementation

Fire path would travel on the scene area with stable rate of growth. The assigned value of each factor that would affect the fire path would be calculated and the sequence of burning will be arranged. The burning fire path will be animating with respecting to these variable's calculations. Moreover, the heat emitting amount from the fire path would be shown as the colour of burning area.

5.3.1 **Rate of growth calculation**

Fire rate of growth would be calculated from considering all the assigned factors in the fire scene area. The amount of heat that requires to ignite the un-ignited fuel and the amount of heat that requires to ignite every fuel in the entire scene would be two values needed to calculate the fire growth rate. In addition, rate of growth would also affect the burning time of each fire point and ground point. If there is higher rate of growth, the fire path colour changing rate would be faster than the changing rate in lower growth rate scene.

5.3.2 **Fire path burning sequence**

This method would control the burning time sequence of every fire path that has been generated in the scene. The fire path burning sequence would consider the generation of each DLA fire point. Every fire point would be arranged in order that respects to the fire point

generation, rate of growth and other factors in the scene, such as slope value, wind value, and moisture value. Therefore, the animation of fire growth would be accurately implemented.

5.3.3 Fire path colour adjusting

Colour of fire would be set depending on the heat emitting value and the burning time, red-ish colour means the new fire born with the high heat emitting amount and colour would be grading to yellow, white, and black when the fuel amount has been decrease from burning state along the time.

5.3.4 Heat transferring to ground area

When the points of fire path have been burning, they would emit the heat and transfer those heats to the surrounding ground area. Thus, the ground point that is located closed to the burning fire path would also be burning.

5.4 Implementation bugs and problems solving

The fire spread subject to the DLA structure only but not subject to the entire scene area

According to the purpose of the project, this project aims to predict the fire path by using DLA structure algorithm. Therefore, there will be the left area that has not been ignited when every fire point is already burnt, which it is not right in real world. The burning area should emit the heat or fire surrounding fuel area also so there will be no place left-over without burning, except the non-flammable area. This problem has been solved by adding the heat emitting and heat transferring function to surrounding area within satisfied function.

Moreover, the place that received heat from burning point should also appear as the burning area. Thus, program has added the colour control function to control the texture mapping of the ground in the scene. The ground of the scene would have the colour of burning state. If the ground point receives large amount of heat, the ground point would have red-ish colour, and if the ground point receives small amount of the heat the ground point would have yellow-ish colour. Furthermore, if those ground points has been received the heat from burning area and has been ignited over the time period, the colour of those point would change to black colour to represent as the ash of fire object.

The effects of wind speed and fire path

The wind function would drag the DLA fire path along with its direction. When the wind is very strong (high speed value), the fire point that has the opposite direction to the wind direction would be less generated. However, after the wind speed has decreased, the new generated fire point has no change from the previous one. Thus, this problem has been fixed by adjusting the position of sphere, which releases the new traveller of DLA creation progress, to be closer to the origin point. Finally, the program can create different fire path from the different wind speed value.

DLA collision detected with other DLA form

When there are more than one fire seed have applied to generate the fire path, program must have check the distance between each DLA structure to not be collided to each other. Thus, the DLA form has to be created one at a time and put the DLA point list into the m_previousDLA list. This list will be used for the next DLA structure creation to check whether the traveller, which will become new point on the structure soon, is located in satisfied distance from previous DLA form. If the condition is not passed, this traveller would be removed and new random traveller would be considered. The function of checking collistion between DLA structure works fine but when the fire path is about to be burning, the burning fire point would look un-natural. Therefore, this problem has solved by adding another function of checking current traveller with other origin point and considers the possibility to stick the traveller to the DLA form, instead of only considering the collision with DLA point of previous form. Moreover, the form number of each DLA structure has applied to the m_DLApointList. Thus, program would know which form that the fire point is belong to.

Furthermore, the function to draw fire path would also check the collision with the other DLA point, subject to form number of each burning point. If the path is collided, those points would not be drawn. This function works fine if the fire seeds are far from each other in appropriate distance. However, if the fire seeds stay closed together, the collision would be error and the appearance of fire path would not look like the DLA form because the collision point has not been drawn but the other point that is chain from that point has been drawn. This problem would not be solved to satisfied result yet because there are still some remain point that draw

without the head line point. Nevertheless, the program has adjusted the distance of collision checking and has added the drawing and burning status to those burning point to try to solve this problem.

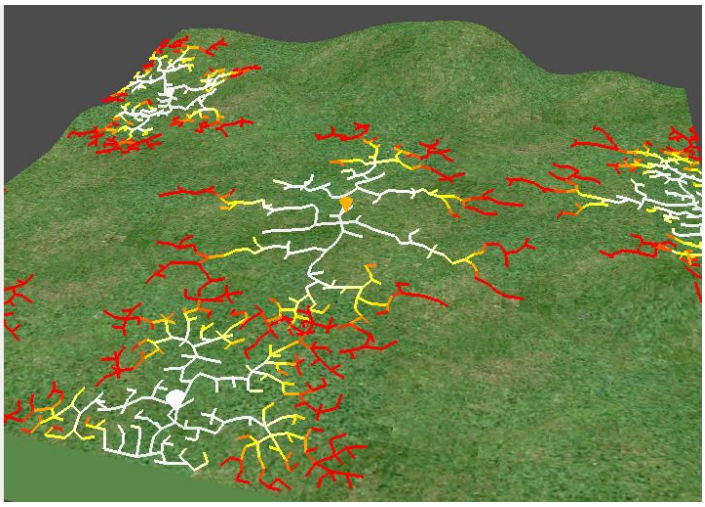


Figure 17: The bugs from collision checking function between different DLA form

DLA generation number conflict

Generation number of each DLA branch should be applied to the new DLA point, created in the same form number. Hence, the generation would be considered to calculate and re-arrange the fire path, subject to the scene factors, such as wind and slop function. However, the assigned generation number has conflict with those scene factors when the size of DLA structure is too big. The program then has to re-arrange the list by decreasing the generation number if the wind and slope function have been considered. Furthermore, the generation number would consider the distance from origin list also.

Ground point's heat transferring to surrounding ground area

After the ground received the heat from burning fire path point, the ground point should be ignited if the amount of heat is large enough. However, if the ground point is ignited, it should be able to transfer the heat to the other ground points, like fire path does. But the problem is it would conflict with other function that will check the burning status of ground point with fuel object area. Therefore, the ground point would still transfer the heat to surrounding points and set

the fire if the positions of two points are closed enough. The problem has been solved by the adjusting the amount of heat transfer to be half of the recent amount of current ground point. In addition, for the conflicted function, that function has been changed to consider only the heat amount, instead of burning status. Thus, the ground point can emit and transfer the heat without any conflict with other burning status checking function. Moreover, the program will also check if the surrounding point, which is received the heat from burning area, has not been burnt yet. Thus, the re-ignited issue with the non-fuel point would not occur.

Colour controlling from emitting heat area

The colour would be changed from red-ish to yellow-ish colour if the heat at the area has decreased more than half of initial heat amount. Moreover, the heat transferring function is also being called if those points still ignite. The collision has occurred here. For illustrations, when the point A has decreased and the colour has also changed to the yellow-ish colour, and afterwards, when the other fire point (point B) within the short distance just ignites, it will transfer the heat to point A. The heat amount of point A will be increased, which it means the colour of point A will be back to red-ish colour. This is the collision happens between the heat emitting and colour controlling function. Thus, the program has been assigned the new status to the point to check wheather if the fuel has already burning down to the state of yellow-ish colour. If yes, then that point should not be changed the colour back to red again.

Furthermore, the ground texture mapping that would appear the different colour of burning area does not look smooth because the texture mapping considered the assigning position with GL_QUADS drawing. Thus, the burning colour texture that appears on the screen has mapped on the small box area to draw burning surface. The following Figure 17 shows the colour texture mapping on quads of burning area.

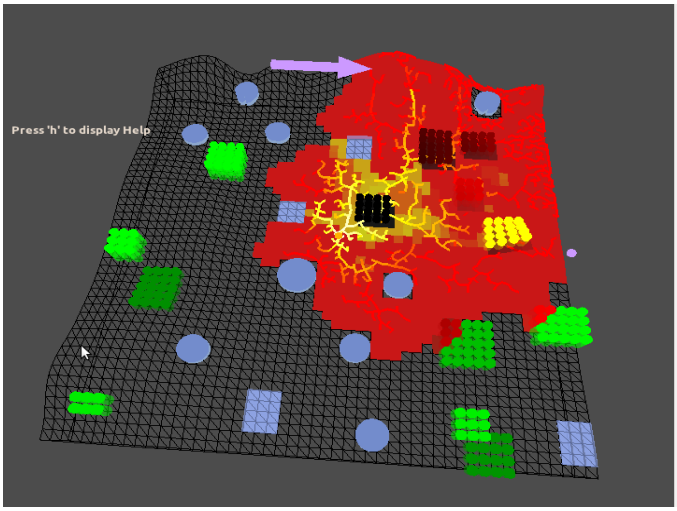


Figure 18: Colour of burning and quads texture mapping on the grid

In addition, for the issue of colour controlling, the heat emitting value colour has been checked and it is accurately regards to time period and the different heat amount in burning part. Thus, the issue should come from the code that has to works with shaders of those area because the output of heat emitting colour still looks un-smooth and the shaders considered to only the Quads surface, which looks similar to the big pixel box. Thus, the way to fix this issue is to decrease the Quads size then the program would subject the shaders to smaller pixel box, which it would be more delicate than it appears in Figure 18.

Fire path speed

When fire path starts the fire, it should go over the place along the generation of DLA path and scene factors that affects the fire. The speed of fire path would also be adjusted if the moisture and temperature of the scene has adjusted while the animation is played. However, the animation of fire path does not grow smoothly, but it grows like the step, respecting to the total burning time of points in current burning generation. This problem makes the fire spread looks strange. Therefore, the function to analyze the time step and the drawing fire path has been added to control the animation. This function would calculate the total number of burning time of points in each generation and consider it with the current time step and the current drawing generation to toggle the time.

Nevertheless, the problem of un-smooth fire spread animation still remains. The fire path has been drawn with un-stable speed. Fire points, located in long distance from origin point would growth slower than fire point, located in short distance from origin point. Therefore, to fix the problem, the program has re-arranged animation controllers by considering the current burning fire point with the distance from origin point. Afterwards, the speed at the drawing spot would be adjusted. Hence, the fire growth would be animated with the stable speed.

Animation controls

Although all the results regards to the fire growth behavior, the animation of fire path did not smooth because the program drew the fire line by concerning the generation of fire path at the time. The program did not draw each fire point over the time period, thus, it looked un-smooth. Actually, fire path should travel along the ground like the slow flow of water. However, the animation has been fixed by adjusting the order of drawing fire path by considering each fire point index number parallel with considering the generation of each fire form.

Fire animation has been controlled by current time step and time amount of each fire point required to burn all the fuel. Therefore, the program will toggle the drawing function if it is not necessary or if it is not at the right time. However, when the large amount of burning area has been considered, the program has the slower performance, which it affects the animation of fire. The animation of fire path goes slower when the fire path becomes very large. Thus, the program needs to speed up the animation by double increase the drawing spot from one point increment per time to two point increments per time.

6 RESULTS AND ANALYSIS

The contents in this section would show the results of final project output and analyze the project efficiency, subject to project's purpose. The aim of this project is to create the prototype of fire path by using diffuse-limited aggregation (DLA) structure and predict the fire spread by considering the surrounding of burning area.

Furthermore, the instructions to use the program have been explained in Program User Guide in Appendix B.

6.1 Project performance analysis

The project has been created in OpenGL and C++ language with using the NGL library. The fire path creates in 3D scene and user can adjust the variables of the fire area to create different fire path growth. Fire spreads over the fuel area by calculating the speed growth from variety of factors such as, moisture value of the area, temperature of the area, wind direction, and slope of the terrain.

Program interface

The program allows user to adjust the necessary variables to create the different fire path. Moreover, the program can compute the new fire path from entire scene one at a time, with regarding to particular amount of fire seed, landscape type, landscape weather, wind and number of fuel objects.

This project does not concern the Human-Computer Interface. The adaptable input has been added to the program by using GL Form. The reason of putting this feature in is just for making the changing of each variable easier than in the code. And user would see the different fire path prediction when the satisfied variable is adjusted.

Diffuse-limited aggregation structure

Fire path has been created with DLA structure. If the wind is assigned to the scene, the fire path would be growth along the wind direction. Furthermore, the fire would not go pass the non-flammable area or go out of the scene area. The output of these issues has been created in

satisfied result. DLA form represents the good fire path burning in the flammable area and follows the wind direction. However, there is the small issue of DLA form, which is in the form, as shown in figure 19, there are less fire points around origin point than the fire point that far from origin, which it conflicts with the real fire behavior, the fire area surrounding fire seed should have more intensity or density of fire point more than the fire are in further distance.

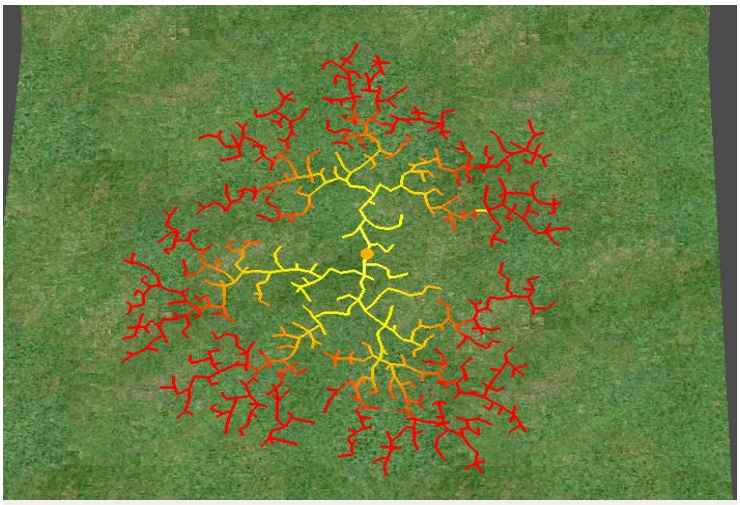


Figure 19: The fire path created by DLA structure

Collided detection with other DLA form

When more than fire seed has been assigned, program will create the fire path regards to those origin positions. The arrangement of each point will be first randomly assigned and user can move particular to the satisfy location. Nevertheless, if the origin points are located too closed together, the collision detection function has not computed the correct result, while the burning state occurred. Thus, the fire line still collides to each other and the burning area will also receive the heat from fire path and re-ignited again. However, the program still does not detect the collision between different DLA form in efficient way, as shown in figure 17, because when the DLA forms approach together, the point line still collides.

Fire growth behavior

For the prediction part, program has arranged the fire growth and speed rate, regards to the scene environments. Rate of growth would be decreased if the scene has high moisture value.

In addition, the fire paths that move against the wind direction would grow slower than the fire paths that move towards the wind direction. The program has also generated the fire growth animation in accurate result, referred to the study of fire behavior and factors of fire spread.

Apart from the wind direction that influences the fire rate of growth, slope area also affects the fire path. Fire will travel to the uphill slope faster than the growth in flat area and downhill slope. Program has computed the slope around the fire path and speed up the fire growth that would travel uphill. Finally, the program has created output of fire growth, regards to this issue. Fire path that goes up to the top of terrain would go faster than fire path that goes on the flat area.

Fire path does not spread all over the fuel area

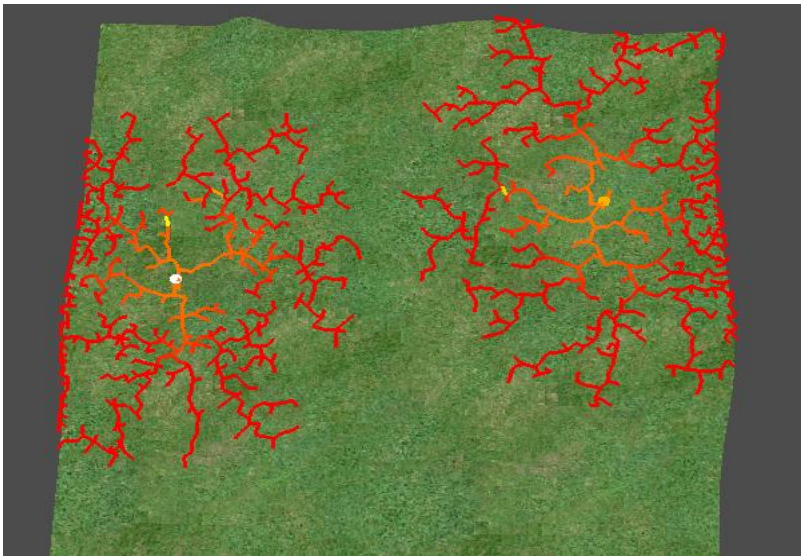


Figure 20: Fire does not spread all over the available fuel area

From the figure 20, it shows the issue of fire does not spread all over the place. When the fire path from DLA form has been shows all the fire point, the fire stop growing the fire. In real world, fire should not stop if there are still the fuel object surrounds, unless, the strong wind has added. Moreover, only area of non-flammable object could stop the fire. Thus, the program occur this issue because the DLA form has limited the size before it creates to reduce the computing time. However, this size limited should be adjust to solve the problem.

Fuel object

When fire path hits the fuel object, that object will start the fire. The burning time of the fuel object depends on moisture and fuel amount of particular object. Thus, different object fuel amount and different object moisture would have different time needed for each object to be burnt down. After the object received the heat from fire path, the moisture of object will be decreased. Moreover, when the object has less moisture, which will be able to start the fire. That object would be ignited. Afterwards, the fuel amount of the object would be decreased along the time period and fire growth rate. Program has set the colour fire object from green to red when it receives enough heat and start burning. The colour would be changing from red to black by depending on the fuel amount. The result of the program has regards to the fuel amount. Whenever, all the fuel has gone, the object will present as the black ash in the scene and cannot start the fire again.

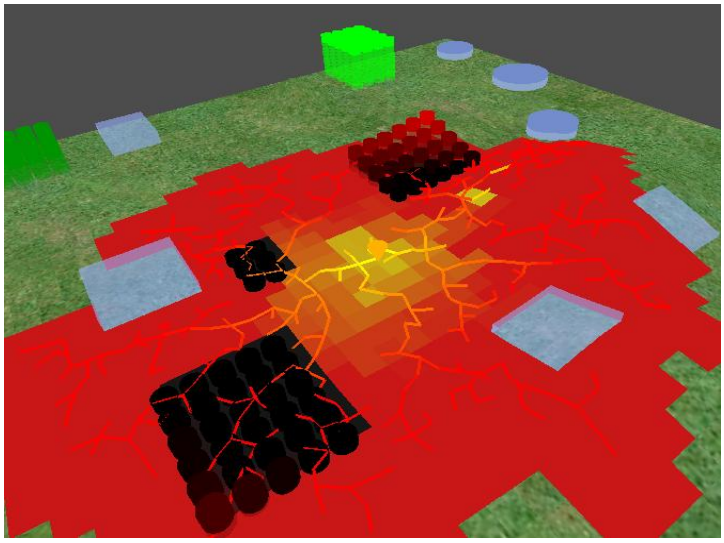


Figure 21: The burning down behavior of fuel object

Wind direction

Fire has gone along with the assigned wind direction. Moreover, the speed growth of each fire branch would affects from the wind speed and also the direction of both wind and fire branch. Figure 22 has shown the program output that created the fire form, which has the result from the wind.

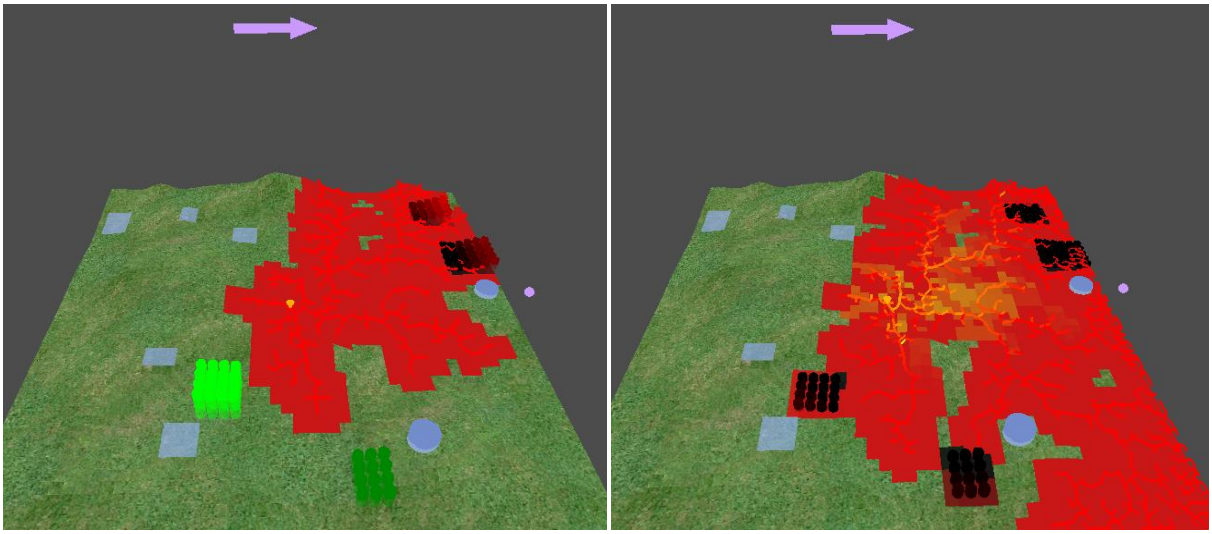


Figure 22: The wind added to fire scene with specific direction and speed

Heat transfer animation

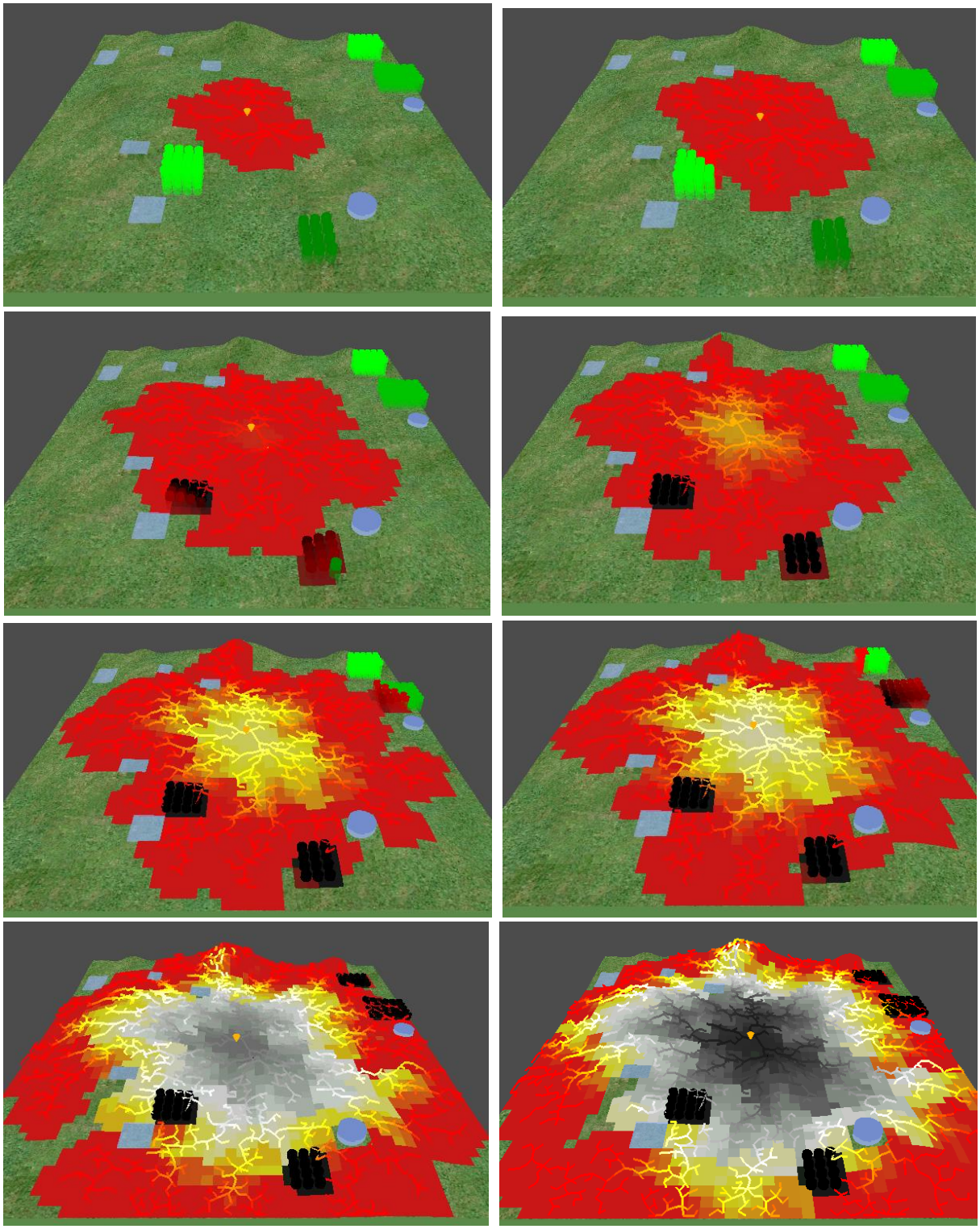
The fire path has assigned the different colour because the different heat amount of particular burning point. Colour of fire path looks fine with the heat variables. However, the grid point that is located near the fire path should also have the effect from burning point because it will receive the heat from the fire. Thus, the texture mapping to control the colour of ground in burning area is added to the program. The appearance of ground, which receive heat from burning path, looks strange and not delicate. After the several checking with amount of transferred heat, the problem would be in the shading function of texture mapping. Moreover, this shading function should be solved to get the smooth and delicate burning area surface.

Fire path animation

Fire animation has been controlled by current time step and time amount of each fire point required to burn all the fuel. Therefore, the program will toggle the drawing function if it is not necessary or if it is not at the right time. However, when the large amount of burning area has been considered, the program has the slower performance, which it affects the animation of fire. The animation of fire path goes slower when the fire path becomes very large. Thus, the program needs to speed up the animation by double increase the drawing spot from one point increment per time to two point increments per time.

6.2 Final project output

The following figures would show the fire growth, which has been generated by the program.



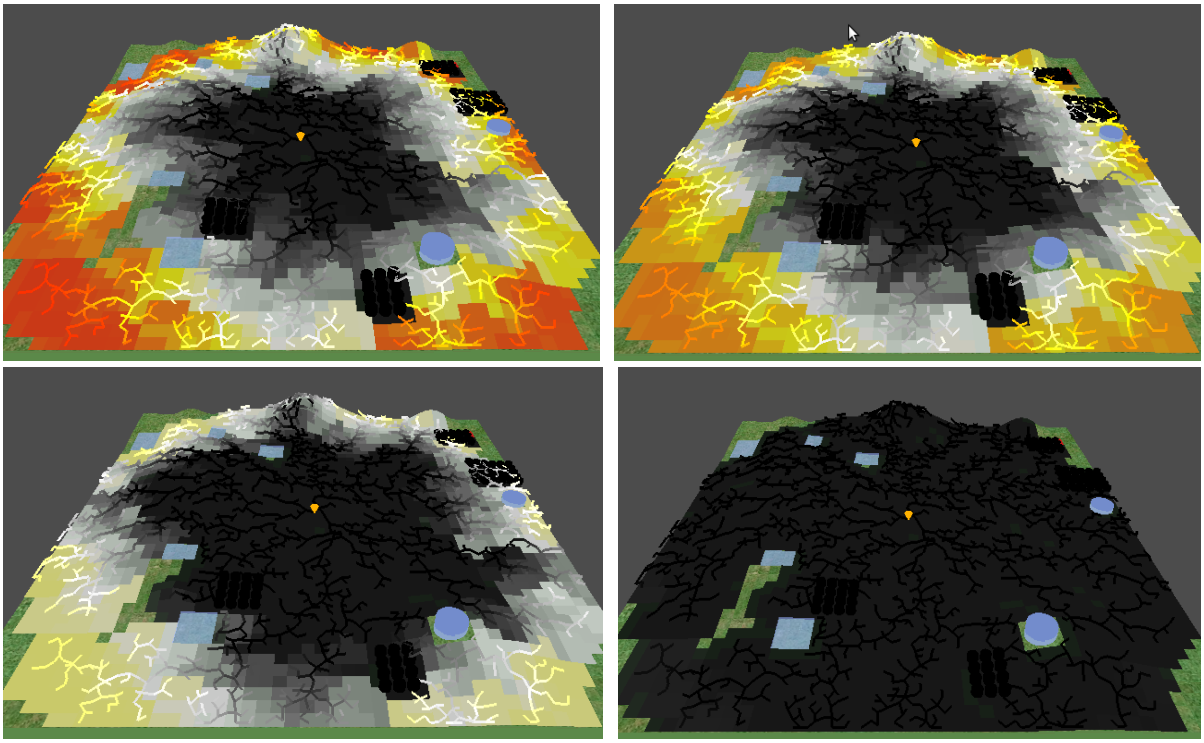


Figure 22: Final output from the program

7 CONCLUSION

7.1 Final output discussion

Fire path has been created in DLA structure and growth regards to the influence from environment area. The fire structure has achieved the purpose of creating the fire path regards to diffuse-limited aggregation structure. Moreover, the fire growth behavior has been generated by considering the influence from surrounding, which it is another main purpose of doing this project.

Although, the conditions of fire growth and the fire path generating have been working efficiently, the program has occur the issues with controlling animation from the created fire list and control the shading colour of burning area. The program still has the un-stable speed control when the fire path reaches the high number of generation. For the issue of colour shading, this does not affect to the fire growth rate. However, the problem of controlling colour of heat amount just becomes the smaller issue that have an influence to aesthetic appearance of the program. Therefore, for the future, this small issue should have been first improved because the aesthetic look of program appearance gives the powerful and attractive look to the program.

For the final output, all the purpose has been achieved. However, just only some of the program objectives do not have the great performance yet. The following contents explain where the small issue of the program should be developed and improved in the future.

The issue that should be improved and developed in the future

1. The shaders of heat emitting function should be developed by improved the texture mapping area.

2. The generation of DLA that would arrange the fire growth index should be improved by re-arranged the generation number, regards to time born of each fire point. Thus, the fire growth animation would be stable and runs faster when it reaches the far distance from fire seed.

3. The calculation of creating DLA structure that would be affected from wind direction and wind speed should be improved to have more limited area for fire spread, subject to the wind speed (which can be considered as wind strength).

4. The animation of fire spread should be developed to having more efficiency growth, subject to the fire growth in real world. It should not growth stop or pause the growth if there is not non-flammable fuel around.

5. The ground of fire area should assigned the fuel amount variables, like fuel object. Hence, the colour changing would be more notice that it is different from each other.

6. The moisture of fuel object should have more effects to the fire growth of each burning fuel object, instead of only effecting on decelerate the speed.

7. The ignited fuel object should have effects on main fire path spread, such as speed up the fire path that surrounds burning fuel object.

8. The fire intensity and fire length of each burning point should have more influence in the fire prediction growth because they are the significant variables of fire behavior.

9. DLA form that will represent the fire path should be improved because the points from different DLA form number still collide to each other.

10. DLA form should be improved by adding the stickiness function or possibilities function to each DLA branch. Hence, before the traveller sticks to the form, it would consider these functions first to check if it should become the new point on DLA form or not.

11. The drawing status of burning fire point should be improved to avoid the head fire missing issue and only the tail of fire appears in the scene.

12. The scene should have added the size adjusting function. Thus, the scene can be bigger or slower according to the satisfaction of user.

13. Distance between each fire path should be able to adjust each time the scene is edited. Therefore, the program will be improved and having more fire pattern for user to see.

14. When the program needs to create more than one origin points, program should be improved the DLA fire path generating to avoid the un-stable distance between each fire point and to develop the collision detected function of current form and others form. The program should not create the DLA form respectively the order of index number of origins point. It should create each DLA form in parallel.

15. The animation of fire growth should be improved to avoid growing un-stable speed rate and avoiding the growing of each branch as the step, that would make the animation does not look smooth.

16. After the fire growth has stopped growing, there are several un-ignited spots left over in the scene. This issue should have been fixed because among the bon fire, flammable point in those areas should have been ignited.

7.2 Future work

1. Program should be first developed all the small issue that stated previously.

2. Apart from the 3D terrain scene, fire path should be applied to 3D object model to make the program be more efficiency.

3. Apart from prototype of fire growth prediction, program should be developed to visualize the fire along with the predicted fire path. Therefore, the real fire path would be appeared in the program and it makes the program to be more attractive.

4. After the prototype of fire prediction, it should be able to export to 3D software, such as Houdini or Maya. Therefore, those programs can import the predicted fire path and the scene environments to generate the realistic fire scene that would grow regards to the predicted rate of growth. Moreover, those cylinders and boxes of fuel object can be replaced as some flammable and non-flammable object to make the scene looks more realistic and gains the aesthetic of the program.

5. The flammable and non-flammable object should be able to be replaced with the real 3D object that is created in 3D software. Thus, the program would look more aesthetic and attractive.

REFERENCES

Anonymous, 2001. *Fire behavior*. USA: Auburn university. Available from: https://fp.auburn.edu/fire/ [Accessed 18 June 2011]

Anonymous, 2003. *Introduction to forest fire*. Thailand: Forest fire control division department. Available from: http://www.dnp.go.th/forestfire/Eng/indexeng.htm [Accessed 15 June 2011]

Anonymous, 2004. *Special edition: Fire behavior*. USA: FirefighterCloseCalls.com. Available from: http://www.firefighterclosecalls.com/site/firefighterclosecalls/drills/Special%20Edition%20Fire%20Behavior.pdf [Accessed 10 June 2011]

Anonymous, 2011. 11 Facts about wildfires.USA: Do Something Inc. Available from: http://www.dosomething.org/tipsandtools/11-facts-about-wildfire [Accessed 12 June 2011]

Anonymous, 2011. *Fuel types and fire behavior*. USA: Idaho firewise. Available from: http://www.idahofirewise.org/science/fuel-types/ [Accessed 12 June 2011]

Bergemann, T., 27 April 2011. Relative humidity. *Wikipedia*. Available from: http://en.wikipedia.org/wiki/Relative_humidity [Accessed 18 June 2011]

Bourke, P., 2004. DLA - Diffusion Limited Aggregation. Australia: Paulbourke.net. Available from: http://paulbourke.net/fractals/dla/ [Accessed 20 June 2011]

Emily, Helen, and Mohamed, 2000. Forest fires: causes of fires. *Forces of Nature*. Australia: ThinkQuest organizer. Available from: http://library.thinkquest.org/C003603/english/forestfires/causesoffire.shtml [Accessed 15 June 2011]

Farmer, S., 2009. Facts about forest fires. *Environment and conservation*. USA: Factoidz. Available from http://factoidz.com/facts-about-forest-fires/ [Accessed 15 June 2011]

Hanson, H.P., 2011. *Wildfire*. USA: hphanson.com. Available from: http://www.hphanson.com/fiction/environment/SERE/wildfire.shtml# [Accessed 12 June 2011]

Hatton, R., 2004. *Fire behavior*. USA: Portland Community College. Available from: http://spot.pcc.edu/~rhatton/fp202Spotfirebehv.pdf [Accessed 15 June 2011]

Jenkins, M., 2008. *Unit 11: Fire Behavior Prediction Systems* [photograph]. USA: Utah State University. Available from: http://ocw.usu.edu/Forest__Range__and_Wildlife_Sciences/Wildland_Fire_Management_and_Planning/Unit_11__Fire_Behavior_Prediction_Systems_2.html [Accessed 1 August 2011]

Kennard, D., 2008. Fire Intensity. *Fire behavior*. USA: Forest encyclopedia network. Available from: http://www.forestencyclopedia.net/p/p486/view [Accessed 21 June 2011]

Lam, C., 2010. *Diffusion Limited Aggregation (a fractal growth model).* Hong Kong: Hong Kong Polytechnic University. Available from: http://apricot.polyu.edu.hk/~lam/dla/ [Accessed 22 June 2011]

Little, S., 2011. Let's Talk Fire: Backdraft, A Firefighter Nightmare. *Employment and Jobs.* USA: hubpages.com. Available from: http://hubpages.com/hub/Lets-Talk-Fire-Backdraft-A-Firefighter-Nightmare [Accessed 12 June 2011]

Little, S., 2011. *The fire tetrahedron, the backbone of teaching fire behavior* [photograph] USA: hubpages.com. Available from: http://hubpages.com/hub/Lets-Talk-Fire-Backdraft-A-Firefighter-Nightmare [Accessed 12 June 2011]

Madrzykowski, D., 2011. *Fire behavior*. USA: NIST. Available from: http://www.nist.gov/fire/fire_behavior.cfm [Accessed 18 June 2011]

Morais, M., 2001. *Predicting fire spread*. USA: University of California, Santa Barbara. Available from: http://www.physics.ucsb.edu/~complex/research/hfire/fbehave/fbehave_predict.html [Accessed 21 June 2011]

Natural history museum, 2011. *Forest fires*. UK: Natural history museum. Available from: http://www.nhm.ac.uk/nature-online/earth/volcanoes-earthquakes/forest-fire/index.html [Accessed 10 June 2011]

NYCEMT86, 17 July 2007. Basic Principles of Fire Behavior. *JREF Forum.* USA. Available from: http://forums.randi.org/showthread.php?t=87587 [Accessed 12 June 2011]

Pietronero, L.,1989. Fractals' physical origin and properties. New York: Plenum press. Available from: http://books.google.com/books?id=ks49t6QcfY8C&pg=PA232&lpg=PA232&dq=dla+fractal&source=bl&ots=KnUYDwiDTs&sig=dYzFyKwfUVumMEeu9GooMMrSy1s&hl=en&ei=-DcdTvr7H8Kk8QOE9OieCA&sa=X&oi=book_result&ct=result&resnum=4&ved=0CCwQ6AEwAzgU#v=onepage&q=dla%20fractal&f=false [Accessed 23 June 2011]

Putnam, A.H., and Karels, J.R., 2004. *Chapter VII: Fire behavior*. USA: Florida forest service. Available from: http://www.fl-dof.com/wildfire/wf_pdfs/ibpf_ch8_fire_behavior.pdf [Accessed 18 June 2011]

Reas, C., and McWilliams, C., 2011. *Simulate: Diffusion-limited aggregation*. USA:formandcode.com. Available from: http://formandcode.com/code-examples/simulate-dla [Accessed 26 June 2011]

Reavis, B., 2010. Diffuse Limited Aggregation Rendered with Sunflow. USA. Available from: http://thirdroute.com/projects/dla/ [Accessed 7 July 2011]

Rothermel, R.C., 1972. *A Mathematical model for predicting fire spread in wildland fuels*. Utah: USDA Forest service. Available from: http://www.fs.fed.us/rm/pubs_int/int_rp115.pdf [Accessed 25 June 2011]

Rothermel, R.C., 2011. *How to predict the spread and intensity of forest and range fires*. Arizona: National advanced resource technology center. Available from: http://cnre.vt.edu/for2514/webarticles/intro-ch1.pdf [Accessed 25 June 2011]

Sottile, M., 2010. *3D Diffusion limited aggregation in Haskell*. USA: WordPress.com. Available from: http://syntacticsalt.com/2010/08/22/3d-diffusion-limited-aggregation-in-haskell/ [Accessed 20 June 2011]

Stock, M., 2009. *DLA3D*. USA: Mark Stock. Available from: http://markjstock.org/dla3d/ [Accessed 22 June 2011]

Tubtub, 2006. *Lightning* [Photograph]. Desktopexchange.com. Available from: http://www.desktopexchange.com/gallery/Nature-Wallpapers/Lightning_pictures [Accessed 22 June 2011]

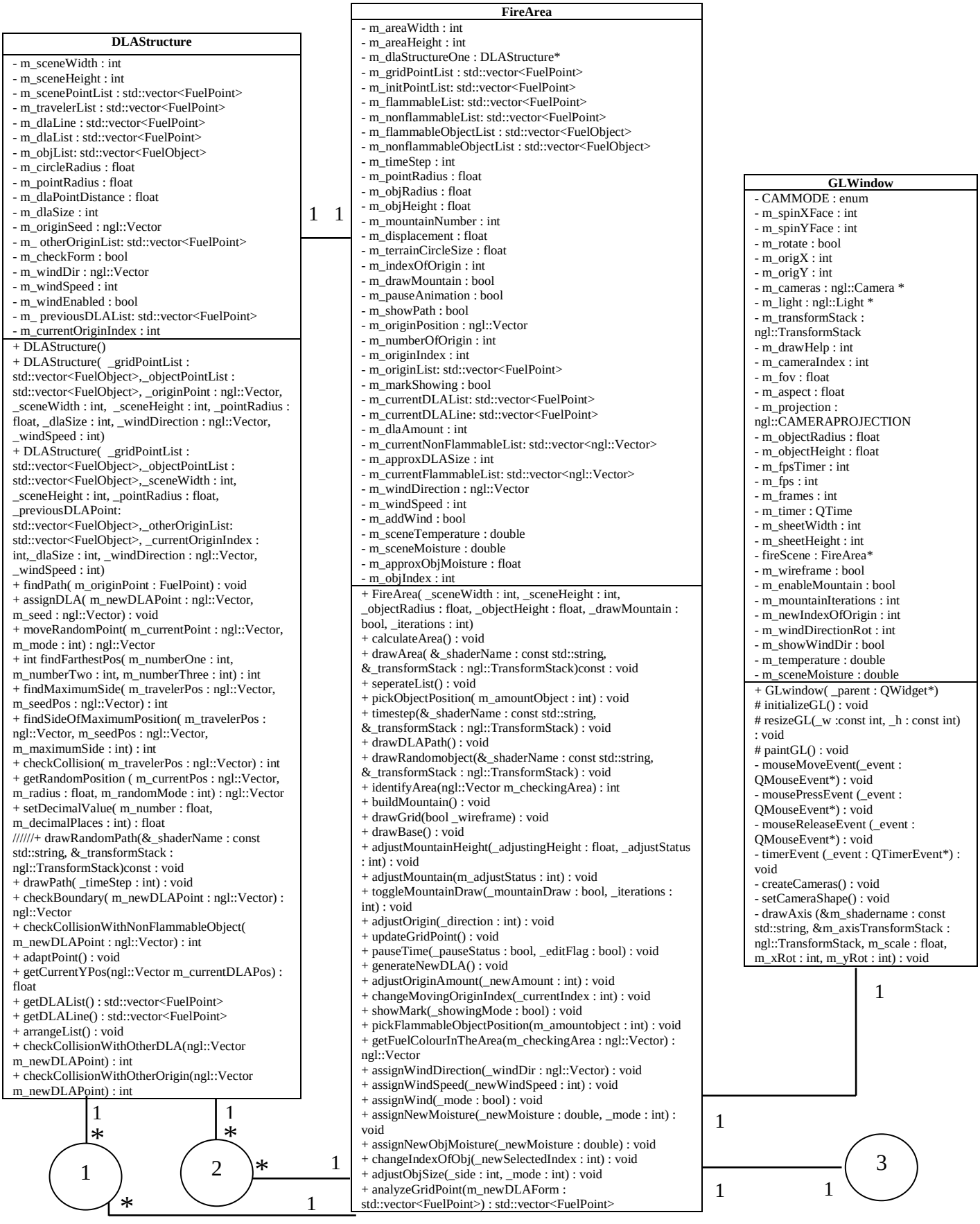
Vallieres, M., 2011. *Diffusion Limited Aggregates (DLA).* Philadelphia: Drexel University. Available from: http://www.physics.drexel.edu/~valliere/PHYS305/Monte_Carlo/Monte_Carlo_story/node6.html [Accessed 26 June 2011]

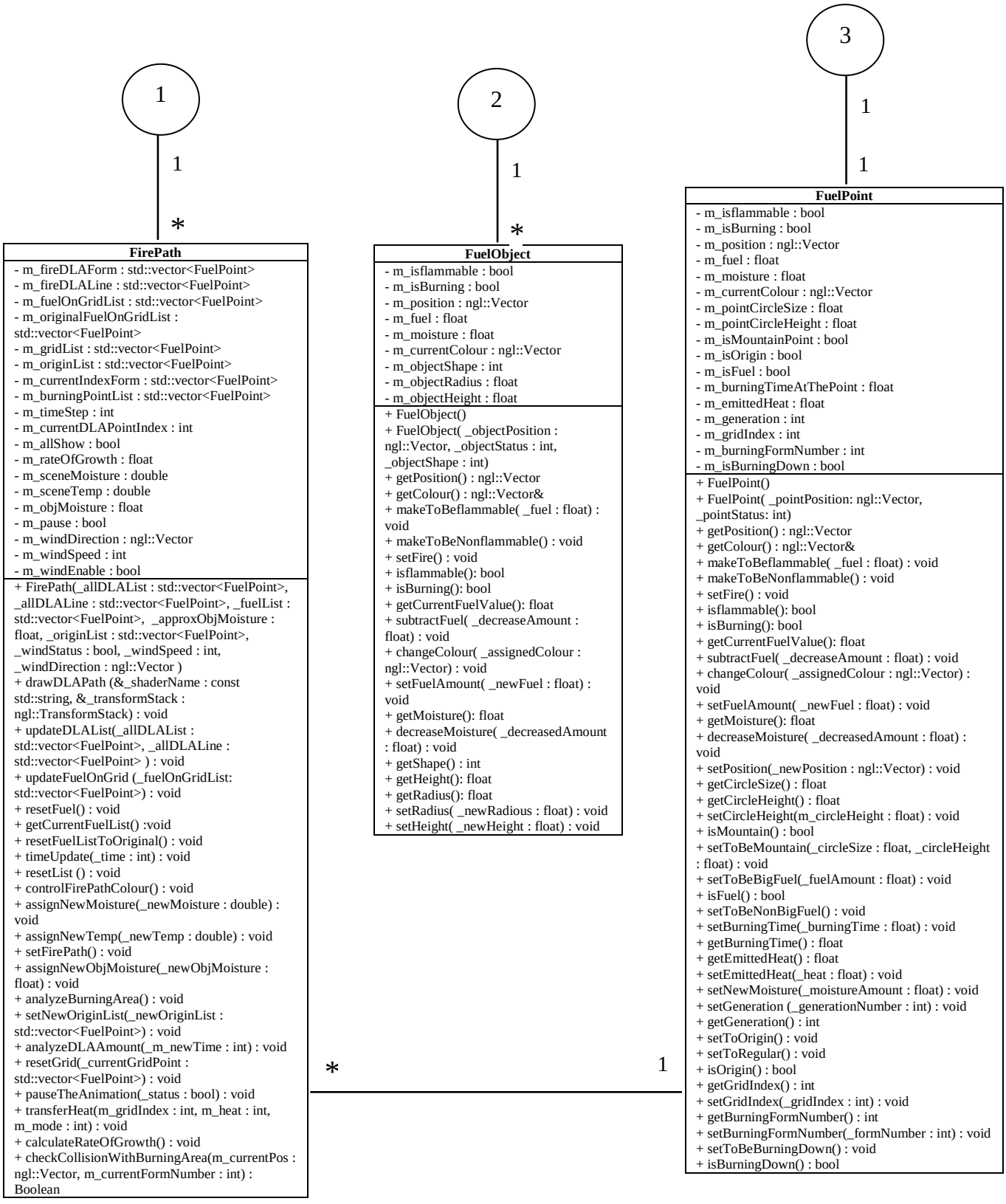
Wurm, P., Instone, L., and Parr, K., 2011. *Fire behavior*. Australia: Tropical Savannas CRC & Bushfire CRC. Available from: http://learnline.cdu.edu.au/units/sbi263/fundamentals/behaviour.html [Accessed 15 June 2011]

Xia, Q., and Unger, D., 2008. Diffusion-limited aggregation driven by optimal transportation. USA: UC Regents, Davis ampus. Available from: http://www.math.ucdavis.edu/~qlxia/Research/dla.pdf [Accessed 22 June 2011]

APPENDIX A.

Class Diagram and Details


DLAStructure
- m_sceneWidth : int
- m_sceneHeight : int
- m_scenePointList : std::vector<FuelPoint>
- m_travelerList : std::vector<FuelPoint>
- m_dlaLine : std::vector<FuelPoint>
- m_dlaList : std::vector<FuelPoint>
- m_objList: std::vector<FuelObject>
- m_circleRadius : float
- m_pointRadius : float
- m_dlaPointDistance : float
- m_dlaSize : int
- m_originSeed : ngl::Vector
- m_ otherOriginList: std::vector<FuelPoint>
- m_checkForm : bool
- m_windDir : ngl::Vector
- m_windSpeed : int
- m_windEnabled : bool
- m_ previousDLAList: std::vector<FuelPoint>
- m_currentOriginIndex : int
+ DLAStructure()
+ DLAStructure(_gridPointList : std::vector<FuelObject>,_objectPointList : std::vector<FuelObject>, _originPoint : ngl::Vector, _sceneWidth : int, _sceneHeight : int, _pointRadius : float, _dlaSize : int, _windDirection : ngl::Vector, _windSpeed : int)
+ DLAStructure(_gridPointList : std::vector<FuelObject>,_objectPointList : std::vector<FuelObject>,_sceneWidth : int, _sceneHeight : int, _pointRadius : float, _previousDLAPoint: std::vector<FuelObject>,_otherOriginList: std::vector<FuelObject>, _currentOriginIndex : int,_dlaSize : int, _windDirection : ngl::Vector, _windSpeed : int)
+ findPath(m_originPoint : FuelPoint) : void
+ assignDLA(m_newDLAPoint : ngl::Vector, m_seed : ngl::Vector) : void
+ moveRandomPoint(m_currentPoint : ngl::Vector, m_mode : int) : ngl::Vector
+ int findFarthestPos(m_numberOne : int, m_numberTwo : int, m_numberThree : int) : int
+ findMaximumSide(m_travelerPos : ngl::Vector, m_seedPos : ngl::Vector) : int
+ findSideOfMaximumPosition(m_travelerPos : ngl::Vector, m_seedPos : ngl::Vector, m_maximumSide : int) : int
+ checkCollision(m_travelerPos : ngl::Vector) : int
+ getRandomPosition (m_currentPos : ngl::Vector, m_radius : float, m_randomMode : int) : ngl::Vector
+ setDecimalValue(m_number : float, m_decimalPlaces : int) : float
//////+ drawRandomPath(&_shaderName : const std::string, &_transformStack : ngl::TransformStack)const : void
+ drawPath(_timeStep : int) : void
+ checkBoundary(m_newDLAPoint : ngl::Vector) : ngl::Vector
+ checkCollisionWithNonFlammableObject(m_newDLAPoint : ngl::Vector) : int
+ adaptPoint() : void
+ getCurrentYPos(ngl::Vector m_currentDLAPos) : float
+ getDLAList() : std::vector<FuelPoint>
+ getDLALine() : std::vector<FuelPoint>
+ arrangeList() : void
+ checkCollisionWithOtherDLA(ngl::Vector m_newDLAPoint) : int
+ checkCollisionWithOtherOrigin(ngl::Vector m_newDLAPoint) : int
1 1
FireArea
- m_areaWidth : int
- m_areaHeight : int
- m_dlaStructureOne : DLAStructure*
- m_gridPointList : std::vector<FuelPoint>
- m_initPointList: std::vector<FuelPoint>
- m_flammableList: std::vector<FuelPoint>
- m_nonflammableList: std::vector<FuelPoint>
- m_flammableObjectList : std::vector<FuelObject>
- m_nonflammableObjectList : std::vector<FuelObject>
- m_timeStep : int
- m_pointRadius : float
- m_objRadius : float
- m_objHeight : float
- m_mountainNumber : int
- m_displacement : float
- m_terrainCircleSize : float
- m_indexOfOrigin : int
- m_drawMountain : bool
- m_pauseAnimation : bool
- m_showPath : bool
- m_originPosition : ngl::Vector
- m_numberOfOrigin : int
- m_originIndex : int
- m_originList: std::vector<FuelPoint>
- m_markShowing : bool
- m_currentDLAList: std::vector<FuelPoint>
- m_currentDLALine: std::vector<FuelPoint>
- m_dlaAmount : int
- m_currentNonFlammableList: std::vector<ngl::Vector>
- m_approxDLASize : int
- m_currentFlammableList: std::vector<ngl::Vector>
- m_windDirection : ngl::Vector
- m_windSpeed : int
- m_addWind : bool
- m_sceneTemperature : double
- m_sceneMoisture : double
- m_approxObjMoisture : float
- m_objIndex : int
+ FireArea(_sceneWidth : int, _sceneHeight : int, _objectRadius : float, _objectHeight : float, _drawMountain : bool, _iterations : int)
+ calculateArea() : void
+ drawArea(&_shaderName : const std::string, &_transformStack : ngl::TransformStack)const : void
+ seperateList() : void
+ pickObjectPosition(m_amountObject : int) : void
+ timestep(&_shaderName : const std::string, &_transformStack : ngl::TransformStack) : void
+ drawDLAPath() : void
+ drawRandomobject(&_shaderName : const std::string, &_transformStack : ngl::TransformStack) : void
+ identifyArea(ngl::Vector m_checkingArea) : int
+ buildMountain() : void
+ drawGrid(bool _wireframe) : void
+ drawBase() : void
+ adjustMountainHeight(_adjustingHeight : float, _adjustStatus : int) : void
+ adjustMountain(m_adjustStatus : int) : void
+ toggleMountainDraw(_mountainDraw : bool, _iterations : int) : void
+ adjustOrigin(_direction : int) : void
+ updateGridPoint() : void
+ pauseTime(_pauseStatus : bool, _editFlag : bool) : void
+ generateNewDLA() : void
+ adjustOriginAmount(_newAmount : int) : void
+ changeMovingOriginIndex(_currentIndex : int) : void
+ showMark(_showingMode : bool) : void
+ pickFlammableObjectPosition(m_amountobject : int) : void
+ getFuelColourInTheArea(m_checkingArea : ngl::Vector) : ngl::Vector
+ assignWindDirection(_windDir : ngl::Vector) : void
+ assignWindSpeed(_newWindSpeed : int) : void
+ assignWind(_mode : bool) : void
+ assignNewMoisture(_newMoisture : double, _mode : int) : void
+ assignNewObjMoisture(_newMoisture : double) : void
+ changeIndexOfObj(_newSelectedIndex : int) : void
+ adjustObjSize(_side : int, _mode : int) : void
+ analyzeGridPoint(m_newDLAForm : std::vector<FuelPoint>) : std::vector<FuelPoint>
GLWindow
- CAMMODE : enum
- m_spinXFace : int
- m_spinYFace : int
- m_rotate : bool
- m_origX : int
- m_origY : int
- m_cameras : ngl::Camera *
- m_light : ngl::Light *
- m_transformStack : ngl::TransformStack
- m_drawHelp : int
- m_cameraIndex : int
- m_fov : float
- m_aspect : float
- m_projection : ngl::CAMERAPROJECTION
- m_objectRadius : float
- m_objectHeight : float
- m_fpsTimer : int
- m_fps : int
- m_frames : int
- m_timer : QTime
- m_sheetWidth : int
- m_sheetHeight : int
- fireScene : FireArea*
- m_wireframe : bool
- m_enableMountain : bool
- m_mountainIterations : int
- m_newIndexOfOrigin : int
- m_windDirectionRot : int
- m_showWindDir : bool
- m_temperature : double
- m_sceneMoisture : double
+ GLwindow(_parent : QWidget*)
# initializeGL() : void
# resizeGL(_w :const int, _h : const int) : void
# paintGL() : void
- mouseMoveEvent(_event : QMouseEvent*) : void
- mousePressEvent (_event : QMouseEvent*) : void
- mouseReleaseEvent (_event : QMouseEvent*) : void
- timerEvent (_event : QTimerEvent*) : void
- createCameras() : void
- setCameraShape() : void
- drawAxis (&m_shadername : const std::string, &m_axisTransformStack : ngl::TransformStack, m_scale : float, m_xRot : int, m_yRot : int) : void
1
1
*
1
*
1
2
*
1
*
1
1
1
1
3
1
1


1
1
*
2
1
*
3
1
1
*
1
FirePath
- m_fireDLAForm : std::vector<FuelPoint>
- m_fireDLALine : std::vector<FuelPoint>
- m_fuelOnGridList : std::vector<FuelPoint>
- m_originalFuelOnGridList : std::vector<FuelPoint>
- m_gridList : std::vector<FuelPoint>
- m_originList : std::vector<FuelPoint>
- m_currentIndexForm : std::vector<FuelPoint>
- m_burningPointList : std::vector<FuelPoint>
- m_timeStep : int
- m_currentDLAPointIndex : int
- m_allShow : bool
- m_rateOfGrowth : float
- m_sceneMoisture : double
- m_sceneTemp : double
- m_objMoisture : float
- m_pause : bool
- m_windDirection : ngl::Vector
- m_windSpeed : int
- m_windEnable : bool
+ FirePath(_allDLAList : std::vector<FuelPoint>, _allDLALine : std::vector<FuelPoint>, _fuelList : std::vector<FuelPoint>, _approxObjMoisture : float, _originList : std::vector<FuelPoint>, _windStatus : bool, _windSpeed : int, _windDirection : ngl::Vector)
+ drawDLAPath (&_shaderName : const std::string, &_transformStack : ngl::TransformStack) : void
+ updateDLAList(_allDLAList : std::vector<FuelPoint>, _allDLALine : std::vector<FuelPoint>) : void
+ updateFuelOnGrid (_fuelOnGridList: std::vector<FuelPoint>) : void
+ resetFuel() : void
+ getCurrentFuelList() :void
+ resetFuelListToOriginal() : void
+ timeUpdate(_time : int) : void
+ resetList () : void
+ controlFirePathColour() : void
+ assignNewMoisture(_newMoisture : double) : void
+ assignNewTemp(_newTemp : double) : void
+ setFirePath() : void
+ assignNewObjMoisture(_newObjMoisture : float) : void
+ analyzeBurningArea() : void
+ setNewOriginList(_newOriginList : std::vector<FuelPoint>) : void
+ analyzeDLAAmount(_m_newTime : int) : void
+ resetGrid(_currentGridPoint : std::vector<FuelPoint>) : void
+ pauseTheAnimation(_status : bool) : void
+ transferHeat(m_gridIndex : int, m_heat : int, m_mode : int) : void
+ calculateRateOfGrowth() : void
+ checkCollisionWithBurningArea(m_currentPos : ngl::Vector, m_currentFormNumber : int) : Boolean
FuelObject
- m_isflammable : bool
- m_isBurning : bool
- m_position : ngl::Vector
- m_fuel : float
- m_moisture : float
- m_currentColour : ngl::Vector
- m_objectShape : int
- m_objectRadius : float
- m_objectHeight : float
+ FuelObject()
+ FuelObject(_objectPosition : ngl::Vector, _objectStatus : int, _objectShape : int)
+ getPosition() : ngl::Vector
+ getColour() : ngl::Vector&
+ makeToBeflammable(_fuel : float) : void
+ makeToBeNonflammable() : void
+ setFire() : void
+ isflammable(): bool
+ isBurning(): bool
+ getCurrentFuelValue(): float
+ subtractFuel(_decreaseAmount : float) : void
+ changeColour(_assignedColour : ngl::Vector) : void
+ setFuelAmount(_newFuel : float) : void
+ getMoisture(): float
+ decreaseMoisture(_decreasedAmount : float) : void
+ getShape() : int
+ getHeight(): float
+ getRadius(): float
+ setRadius(_newRadious : float) : void
+ setHeight(_newHeight : float) : void
FuelPoint
- m_isflammable : bool
- m_isBurning : bool
- m_position : ngl::Vector
- m_fuel : float
- m_moisture : float
- m_currentColour : ngl::Vector
- m_pointCircleSize : float
- m_pointCircleHeight : float
- m_isMountainPoint : bool
- m_isOrigin : bool
- m_isFuel : bool
- m_burningTimeAtThePoint : float
- m_emittedHeat : float
- m_generation : int
- m_gridIndex : int
- m_burningFormNumber : int
- m_isBurningDown : bool
+ FuelPoint()
+ FuelPoint(_pointPosition: ngl::Vector, _pointStatus: int)
+ getPosition() : ngl::Vector
+ getColour() : ngl::Vector&
+ makeToBeflammable(_fuel : float) : void
+ makeToBeNonflammable() : void
+ setFire() : void
+ isflammable(): bool
+ isBurning(): bool
+ getCurrentFuelValue(): float
+ subtractFuel(_decreaseAmount : float) : void
+ changeColour(_assignedColour : ngl::Vector) : void
+ setFuelAmount(_newFuel : float) : void
+ getMoisture(): float
+ decreaseMoisture(_decreasedAmount : float) : void
+ setPosition(_newPosition : ngl::Vector) : void
+ getCircleSize() : float
+ getCircleHeight() : float
+ setCircleHeight(m_circleHeight : float) : void
+ isMountain() : bool
+ setToBeMountain(_circleSize : float, _circleHeight : float) : void
+ setToBeBigFuel(_fuelAmount : float) : void
+ isFuel() : bool
+ setToBeNonBigFuel() : void
+ setBurningTime(_burningTime : float) : void
+ getBurningTime() : float
+ getEmittedHeat() : float
+ setEmittedHeat(_heat : float) : void
+ setNewMoisture(_moistureAmount : float) : void
+ setGeneration (_generationNumber : int) : void
+ getGeneration() : int
+ setToOrigin() : void
+ setToRegular() : void
+ isOrigin() : bool
+ getGridIndex() : int
+ setGridIndex(_gridIndex : int) : void
+ getBurningFormNumber() : int
+ setBurningFormNumber(_formNumber : int) : void
+ setToBeBurningDown() : void
+ isBurningDown() : bool

Class definition

1. **FireArea class**

This class would create the fire area or scene of the program. The objects and point of fire would be generated here in this class. Moreover, wind force, weather temperature, slope of landscape and other fire area's variables are also assigned in this class to identify the world scene for fire to burn.

Variables Names	Types	Definitions
m_areaWidth	Integer	Width value of the scene
m_areaHeight	Integer	Height value of the scene
m_dlaStructureOne	DLAStructure object	Object of DLAStructure class to form the DLA structure
m_gridPointList	std::vector<FuelPoint> list	List of every point position on the scene, refers to FuelPoint class.
m_initPointList	std::vector<FuelPoint> list	List of every point that is burning, refers to FuelPoint class
m_flammableList	std::vector<FuelPoint> list	List contains initial flammable point
m_nonflammableList	std::vector<FuelPoint> list	List contains initial non-flammable point
m_flammableObjectList	std::vector<FuelObject> list	List of non-flammable objects on the scene, refers to FuelObject class
m_nonflammableObjectList	std::vector<FuelObject> list	List of non-flammable objects on the scene, refers to FuelObject class
m_timeStep	Integer	Time step counting number
m_pointRadius	Float	Initial radius of points on the scene
m_objRadius	Float	Initial radius of objects in the scene
m_objHeight	Float	Initial height of objects in the scene
m_mountainNumber	Integer	Number of mountain
m_displacement	Float	Displacement for building mountain (height of mountain)
m_terrainCircleSize	Float	Radius of terrain
m_indexOfOrigin	Integer	Index number of origin point, according to grid point list
m_drawMountain	Boolean	Flag of generating terrain
m_pauseAnimation	Boolean	Flag of pausing animation
m_showPath	Boolean	Flag of showing DLA form
m_originPosition	ngl::Vector	Origin point position
m_numberOfOrigin	Integer	Amount of origin number
m_originIndex	Integer	Current index of origin number in the origin point list

m_originList	std::vector<FuelPoint> list	Origin point list
m_markShowing	Boolean	Flag of showing mark of origin point
m_currentDLAList	std::vector<FuelPoint> list	List of current DLA point
m_currentDLALine	std::vector<FuelPoint> list	List of current DLA head point
m_dlaAmount	Integer	Amount of dla structure
m_currentNonFlammableList	std::vector<ngl::Vector>	List of current non-flammable object's position
m_approxDLASize	Integer	Size of DLA that is assigned by users and calculated the approximate value (with the scene) by the program
m_currentFlammableList	std::vector<ngl::Vector>	List contains current position of flammable object
m_windDirection	ngl::Vector	Wind direction
m_windSpeed	Integer	Wind speed
m_addWind	Boolean	Wind status
m_sceneTemperature	Double	Scene temperature
m_sceneMoisture	Double	Scene moisture
m_approxObjMoisture	Float	Moisture of the fuel object
m_objIndex	Integer	Index number of fuel object

Methods Names	Types	Parameters	Definitions
FireArea(_sceneWidth : int, _sceneHeight : int, _objectRadius : float, _objectHeight : float, _drawMountain : bool, _iterations : int)	Constructor	_sceneWidth : int, _sceneHeight : int, _objectRadius : float, _objectHeight : float, _drawMountain : bool, _iterations : int	Constructor of this class. The width and height of the scene would be assigned. Moreover, the initial value of object's radius and object's height would also be assigned.
calculateArea()	Void	none	This function calculates grid point from the assigned width and height size of the scene. The initial scene will be created with flat landscape before the slope value would be assigned later.
drawArea(&_shaderName : const std::string, &_transformStack : ngl::TransformStack)	Void (Const)	_shaderName : const std::string, _transformStack : ngl::TransformStack	This function draws fire area from the calculated point position list.
seperateList()	Void	none	This function separates point to appropriate list, which are flammable and non-flammable list.
timestep(&_shaderNa	Void	&_shaderName :	This function updates the point to

me : const std::string, &_transformStack : ngl::TransformStack)		const std::string, &_transformStack : ngl::TransformStack	current state
drawDLAPath()	Void	none	This function draws the path of fire, according to DLA structure
drawRandomObject(&_shaderName : const std::string, &_transformStack : ngl::TransformStack)	Void (Const)	_shaderName : const std::string, _transformStack : ngl::TransformStack	This function draws the objects in the scene
pickObjectPosition(m_amountObject : int)	Void	m_amountObject : int	This function arranges the objects position and assigns object's variables, such as radius, height and etc.
identifyArea(ngl::Vector m_checkingArea)	Integer	ngl::Vector m_checkingArea	This function finds the place on fire area to specify as non-flammable area. Thus, fire path would not go through this part.
buildMountain()	Void	none	This function builds the mountain from the specific mountain number and assign the point to be centre of mountain.
drawGrid(bool _wireframe)	Void	bool _wireframe	This function draws the grid
drawBase()	Void	none	This function draws base of the scene, including the ground.
adjustMountainHeight(_adjustingHeight : float, _adjustStatus : int)	Void	_adjustingHeight : float, _adjustStatus : int	This function adjusts the size of mountain.
adjustMountain(m_adjustStatus : int)	Void	m_adjustStatus : int	This function builds the mountain with the current value.
toggleMountainDraw(_ mountainDraw : bool, _iterations : int)	Void	mountainDraw : bool, _iterations : int	This function toggles the drawing or generating the mountain or terrain
adjustOrigin(_direction : int)	Void	_direction : int	This function moves the position of origin with the specified direction.
updateGridPoint()	Void	none	This function updates the new grid point after the new amount of obstacle is created.
pauseTime(_pauseStatus : bool, _editFlag : bool)	Void	_pauseStatus : bool, _editFlag : bool	This function pause the animation of fire path.
generateNewDLA()	Void	none	This function would be called when the new scene is edited and the

			generate button is clicked for creating the fire path.
adjustOriginAmount(_ newAmount : int)	Void	_newAmount : int	This function adjusts the amount of fire origin of the whole scene.
changeMovingOriginIn dex(_currentIndex : int)	Void	_currentIndex : int	This function selects which origin point would be able to move.
showMark(_showingM ode : bool)	Void	_showingMode : bool	This function would be called when the mark's check box has been toggled.
pickFlammableObjectP osition(m_amountobjec t : int)	Void	m_amountobject : int	This function randomly assigns the fuel object's position.
getFuelColourInTheAr ea(m_checkingArea : ngl::Vector)	ngl::Vector	m_checkingArea : ngl::Vector	This function returns the colour of point that is located in the fire area.
assignWindDirection(_ windDir : ngl::Vector)	Void	_windDir : ngl::Vector	This function assigns the new wind direction from user.
assignWindSpeed(_new WindSpeed : int)	Void	_newWindSpeed : int	This function assigns the new wind speed from user.
assignWind(_mode : bool)	Void	_mode : bool	This function toggles the wind function.
assignNewMoisture(_n ewMoisture : double, _mode : int)	Void	_newMoisture : double, _mode : int	This function assigns the new moisture and temperature of the fire scene.
assignNewObjMoisture (_newMoisture : double)	Void	_newMoisture : double	This function assigns the new moisture and temperature of the fuel object.
changeIndexOfObj(_ne wSelectedIndex : int)	Void	_newSelectedIndex : int	This function assigns the new index of the fuel object that will be adjusted the size.
adjustObjSize(_side : int, _mode : int)	Void	_side : int, _mode : int	This function adjusts the radius and height of selecting fuel object.
analyzeGridPoint(m_n ewDLAForm : std::vector<FuelPoint>)	std::vector <FuelPoint >	m_newDLAForm : std::vector<FuelPoint >	This function returns the grid area of current dla form.

2. DLAStructure class

This class would form the DLA structure from the assigned size and boundary area.

Variables Names	Types	Definitions
m_ sceneWidth	Integer	Width value of the scene

m_ sceneHeight	Integer	Height value of the scene
m_ scenePointList	std::vector<FuelPoint>	List of every point on the fire area, refers to FuelPoint class
m_ travelerList	std::vector<FuelPoint> list	List of traveler point position, refers to FuelPoint class.
m_ dlaList	std::vector<FuelPoint> list	List of every point on DLA structure, refers to FuelPoint class
m_ dlaLine	std::vector<FuelPoint> list	List of every point on DLA structure to perform the line path, refers to FuelPoint class
m_ object	std::vector<FuelObject> list	List of objects on the scene, refers to FuelObject class
m_ circleRadius	Float	Initial radius of sphere in DLA process
m_pointRadius	Float	Initial radius of points on the scene
m_ dlaPointDistance	Float	Initial distance between DLA points
m_ dlaSize	Integer	Size of DLA structure
m_originSeed	ngl::Vector	Origin position
m_ otherOriginList	std::vector<FuelPoint>	List of origin points
m_checkForm	Boolean	Flag of checking other DLA structure
m_windDir	ngl::Vector	Wind direction
m_windSpeed	Integer	Wind speed
m_windEnabled	Boolean	Wind flag
m_ previousDLAList	std::vector<FuelPoint>	List of point on previous DLA structure
m_currentOriginIndex	Integer	The current index of origin point

Methods Names	Types	Parameters	Definitions
DLAStructure	Constructor	none	Default constructor of this class.
DLAStructure(_gridPointList : std::vector<FuelObject >,_objectPointList : std::vector<FuelObject >, _originPoint : ngl::Vector, _sceneWidth : int, _sceneHeight : int, _pointRadius : float, _dlaSize : int, _windDirection : ngl::Vector, _windSpeed : int)	Constructor	_gridPointList : std::vector<FuelObject>,_objectPointList : std::vector<FuelObject>, _originPoint : ngl::Vector, _sceneWidth : int, _sceneHeight : int, _pointRadius : float, _dlaSize : int, _windDirection : ngl::Vector, _windSpeed : int	Constructor of this class. The width and height of the scene would be assigned to perform the DLA structure within the bounding area. The object list is assigned to identify the possibility area to be burnt.
DLAStructure(_gridPointList :		_gridPointList : std::vector<FuelObj	Another constructor of this class. This function is for generating more than one

std::vector<FuelObject >,_objectPointList : std::vector<FuelObject >,_sceneWidth : int, _sceneHeight : int, _pointRadius : float, _previousDLAPoint: std::vector<FuelObject >,_otherOriginList: std::vector<FuelObject >, _currentOriginIndex : int,_dlaSize : int, _windDirection : ngl::Vector, _windSpeed : int)		ect>,_objectPointLis t : std::vector<FuelObj ect>,_sceneWidth : int, _sceneHeight : int, _pointRadius : float, _previousDLAPoint : std::vector<FuelObj ect>,_otherOriginLi st: std::vector<FuelObj ect>, _currentOriginIndex : int,_dlaSize : int, _windDirection : ngl::Vector, _windSpeed : int	DLA structure.
findPath(m_originPoint : FuelPoint)	Void	m_originPoint : FuelPoint	This function finds the new DLA clusters to form the structure
assignDLA(m_newDLAPoint : ngl::Vector, m_seed : ngl::Vector)	Void	m_newDLAPoint : ngl::Vector, m_seed : ngl::Vector	This function adds the new DLA point to the list.
moveRandomPoint(m_currentPoint : ngl::Vector, m_mode : int)	ngl::Vector	m_currentPoint : ngl::Vector, m_mode : int	This function moves the traveler point and returns the new position of traveler.
findFarthestPos(m_numberOne : int, m_numberTwo : int, m_numberThree : int)	Integer	m_numberOne : int, m_numberTwo : int, m_numberThree : int	This function finds which orthogonal and side that is farthest from the fire starter point
findMaximumSide(m_travelerPos : ngl::Vector, m_seedPos : ngl::Vector)	Integer	m_travelerPos : ngl::Vector, m_seedPos : ngl::Vector	This function finds which orthogonal and side that is farthest from assigned seed point
findSideOfMaximumP osition(m_travelerPos : ngl::Vector, m_seedPos : ngl::Vector,	Integer	m_travelerPos : ngl::Vector, m_seedPos : ngl::Vector, m_maximumSide :	This function finds the side of current traveler point that should not move

m_maximumSide : int)		int	
checkCollision(m_travelerPos : ngl::Vector)	Integer	m_travelerPos : ngl::Vector	This function checks collision between DLA points and traveler point that is closed together or not.
getRandomPosition (m_currentPos : ngl::Vector, m_radius : float, m_randomMode : int)	ngl::Vector	m_currentPos : ngl::Vector, m_radius : float, m_randomMode : int	This function finds the random point in the area of specific sphere around the origin point
setDecimalValue(m_number : float, m_decimalPlaces : int)	Float	m_number : float, m_decimalPlaces : int	This function sets the specific number to have the specific decimal places as decided
drawPath(_timeStep : int)	Void	none	This function draws path of DLA structure
checkBoundary(m_newDLAPoint : ngl::Vector)	ngl::Vector	m_newDLAPoint : ngl::Vector	This function checks collision between DLA points and edge of fire area
checkCollisionWithNonFlammableObject(m_newDLAPoint : ngl::Vector)	Integer	m_newDLAPoint : ngl::Vector	This function checks collision between DLA points and objects
adaptPoint()	Void	none	This function adjusts the position of DLA point on flat grid with the terrain height.
getCurrentYPos(ngl:: Vector m_currentDLAPos)	Float	ngl::Vector m_currentDLAPos	This function compares current point on flat grid to the point list of terrain and then the new Y-position will be returned.
getDLAList()	std::vector <FuelPoint >	none	This function returns the DLA point list.
getDLALine()	std::vector <FuelPoint >	none	This function returns the DLA head point list.
arrangeList()	Void	none	This function sorts the DLA List with respecting to its generation number.
checkCollisionWithOtherDLA(ngl::Vector m_newDLAPoint)	Integer	ngl::Vector m_newDLAPoint	This function checks the collision with other DLA structure and returns the toggle number.

checkCollisionWithOtherOrigin(ngl::Vector m_newDLAPoint)	Integer	ngl::Vector m_newDLAPoint	This function checks the collision with other Origin Point and returns the toggle number.

3. FuelObject class

This class is the constructor class of fuel object. It would identify fuel object with its variables, such as type, position, shape and etc.

Variables Names	Types	Definitions
m_ isflammable	Boolean	Type of fuel that is flammable or not
m_ isBurning	Boolean	Status of fuel that is burning or not
m_ position	ngl::Vector	Position of fuel
m_ fuel	Float	Amount of fuel
m_ moisture	Float	Moisture of fuel
m_ currentColour	ngl::Vector	Colour of fuel
m_ objectShape	Integer	Shape of fuel that is cylinder or cube
m_ objectRadius	Float	Radius of fuel
m_ objectHeight	Float	Height of fuel

Methods Names	Types	Parameters	Definitions
FuelObject()	Constructor	none	Default constructor of this class.
FuelObject(_objectPosition : ngl::Vector, _objectStatus : int, _objectShape : int)	Constructor	_objectPosition : ngl::Vector, _objectStatus : int, _objectShape : int	Constructor of this class. The fuel would be assigned its initial position, status, and shape.
getPosition()	ngl::Vector	none	This function returns current position of fuel.
getColour()	ngl::Vector	none	This function returns current colour of fuel.
makeToBeflammable(_fuel : float)	Void	_fuel : float	This function makes the fuel to be flammable with assigning fuel amount.
makeToBeNonflammable()	Void	none	This function makes the fuel to be non-flammable.
setFire()	Void	none	This function changes the status of fuel to be burning.
isflammable()	Boolean	none	This function returns type of fuel object that is flammable or not
isBurning()	Boolean	none	This function returns status of fuel that is burning or not.

getCurrentFuelValue()	Float	none	This function returns current fuel amount.
subtractFuel(_decreaseAmount : float)	Void	_decreaseAmount : float	This function decreases fuel amount with the assigned value.
changeColour(_assignedColour : ngl::Vector)	Void	_assignedColour : ngl::Vector	This function set the colour of fuel
setFuelAmount(_newFuel : float)	Void	_newFuel : float	This function set fuel amount
getMoisture()	Float	none	This function returns moisture value of fuel
decreaseMoisture(_decreasedAmount : float)	Void	_decreasedAmount : float	This function decreases moisture of fuel with assigned value
getShape()	Integer	none	This function returns shape of fuel.
getHeight()	Float	none	This function returns height of fuel.
getRadius()	Float	none	This function returns radius of fuel.
setRadius(_newRadious : float)	Void	_newRadious : float	This function sets the radius of fuel.
setHeight(_newHeight : float)	Void	_newHeight : float	This function sets the height of fuel.

4. FuelPoint class

This class is the constructor class of fuel point. It would identify each point on the fire area with its variables, such as current position, colour and etc.

Variables Names	Types	Definitions
m_ isflammable	Boolean	Type of point that is flammable or not
m_ isBurning	Boolean	Status of point that is burning or not
m_ position	ngl::Vector	Position of point
m_ fuel	Float	Amount of fuel
m_ moisture	Float	Moisture of point
m_ currentColour	ngl::Vector	Colour of point
m_isBurningDown	Boolean	Burnt status
m_pointCircleSize	Float	Radius of terrain
m_pointCircleHeight	Float	Height of terrain
m_isMountainPoint	Boolean	Flag of terrain generating
m_isOrigin	Boolean	Origin status
m_isFuel	Boolean	Fuel object status
m_burningTimeAtThePoint	Float	Current burning time

m_emittedHeat	Float	Heat emitting amount
m_generation	Integer	Generation number
m_gridIndex	Integer	Grid index number
m_burningFormNumber	Integer	DLA structure number

Methods Names	Types	Parameters	Definitions
FuelPoint ()	Constructor	none	Default constructor of this class.
FuelPoint(_pointPosition: ngl::Vector, _pointStatus: int)	Constructor	_pointPosition: ngl::Vector, _pointStatus: int	Constructor of this class. The point would be assigned its initial position and status.
getPosition()	ngl::Vector	none	This function returns current position of point.
getColour()	ngl::Vector	none	This function returns current colour of point.
makeToBeflammable(_fuel : float)	Void	_fuel : float	This function makes the point to be flammable with assigning fuel amount.
makeToBeNonflammable()	Void	none	This function makes the point to be non-flammable.
setFire()	Void	none	This function changes the status of point to be burning.
isflammable()	Boolean	none	This function returns type of point object that is flammable or not
isBurning()	Boolean	none	This function returns status of point that is burning or not.
getCurrentFuelValue()	Float	none	This function returns current fuel amount.
subtractFuel(_decreaseAmount : float)	Void	_decreaseAmount : float	This function decreases fuel amount with the assigned value.
changeColour(_assignedColour : ngl::Vector)	Void	_assignedColour : ngl::Vector	This function set the colour of point
setFuelAmount(_newFuel : float)	Void	_newFuel : float	This function set fuel amount
getMoisture()	Float	none	This function returns moisture value of point
decreaseMoisture(_decreasedAmount : float)	Void	_decreasedAmount : float	This function decreases moisture of fuel with assigned value
setPosition(_newPosition : ngl::Vector)	Void	_newPosition : ngl::Vector	This function update the current position to the point
getCircleSize()	Float	none	This function returns the circle radius

			of terrain point.
getCircleHeight()	Float	none	This function returns the height of terrain point
setCircleHeight(m_circleHeight : float)	Void	m_circleHeight : float	This function assigns the height to the terrain point
isMountain()	Boolean	none	This function returns status of point that is the terrain point or not
setToBeMountain(_circleSize : float, _circleHeight : float)	Void	_circleSize : float, _circleHeight : float	This function sets the point to be terrain point with assigned circle size and height.
setToBeBigFuel(_fuelAmount : float)	Void	_fuelAmount : float	This function sets the point to be the point of fuel object.
isFuel()	Boolean	none	This function returns type of point that is located in fuel object area or not
setToBeNonBigFuel()	Void	none	This function sets the point to be regular point that is not located in the area of fuel object.
setBurningTime(_burningTime : float)	Void	_burningTime : float	This function sets the burning time of the point. It would explain how long the point would be burnt down.
getBurningTime()	Float	none	This function will return the current burning time of point.
getEmittedHeat()	Float	none	This function returns the heat of the point that is received from burning fire path.
setEmittedHeat(_heat : float)	Void	_heat : float	This function sets the heat that received from fire path.
setNewMoisture(_moistureAmount : float)	Void	_moistureAmount : float	This function sets the moisture of point.
setGeneration (_generationNumber : int)	Void	_generationNumber : int	This function assigns the generation number of point, referred to the created fire path.
getGeneration()	Integer	none	This function returns the number of generation of point.
setToOrigin()	Void	none	This function sets the point to be origin point.
setToRegular()	Void	none	This function sets the point to be regular point, not origin.
isOrigin()	Boolean	none	This function returns type of point that is an origin point or not
getGridIndex()	Integer	none	This function returns index of grid point from the specific point list.
setGridIndex(_gridIndex : int)	Void	_gridIndex : int	This function sets the index number of grid point.
getBurningFormNumb	Integer	none	This function returns form number of

er()			fire path.
setBurningFormNumber(_formNumber : int)	Void	_formNumber : int	This function sets the form number of fire path that the point is belong.
setToBeBurningDown()	Void	none	This function sets the status of point that is burning down.
isBurningDown()	Boolean	none	This function returns the burnt status of point.

5. GLWindow class

This class is the main GLWindow widget that would run NGL applications. The initialize, drawing, updating, timing, and mouse controlling of the application are generated here. It will control all the input variables to interact with the program. Moreover, it has the object variables that refers to FireArea class to build the scene as the fire area.

Variables Names	Types	Definitions
m_ spinXFace	Integer	X-rotation mouse value
m_ spinYFace	Integer	Y-rotation mouse value
m_ rotate	Boolean	Flag to indicate if the mouse button is pressed when dragging
m_ origX	Integer	Previous x mouse value
m_ origY	Integer	Previous y mouse value
m_ cameras	std::vector <ngl::Camera>	List of cameras
m_ transformStack	ngl::TransformStack	Transformation stack for GL transformation and other transformations
m_ drawHelp	Integer	Flag to toggle help information
m_ cameraIndex	Integer	Index of camera list to active the specific camera
m_ fov	Float	FOV value for the camera
m_ aspect	Float	Aspect ratio of the camera
m_ projection	ngl::CAMERAPROJECTION	Projection mode of the camera
m_ objectRadius	Float	Initial radius of fuel object
m_ objectHeight	Float	Initial height of fuel object
m_ fpsTimer	Integer	Flag for the fps timer
m_ fps	Integer	Number of fps (frames per second) to draw the scene
m_ frames	Integer	Frame number
m_ timer	QTime	Time counter
m_ sheetWidth	Integer	Width of scene
m_ sheetHeight	Integer	Hiehgt of scene
fireScene	FireArea	Object of FireArea class
m_wireframe	Boolean	Flag to toggle wireframe

m_enableMountain	Boolean	Flag to generate the terrain
m_mountainIterations	Integer	Terrain iterations number
m_newIndexOfOrigin	Integer	Index number of origin point
m_windDirectionRot	Integer	Rotation value of wind direction
m_showWindDir	Boolean	Flag to show the arrow of wind
m_temperature	Double	Scene temperature
m_sceneMoisture	Double	Scene moisture

Methods Names	Types	Parameters	Definitions
GLwindow(_parent : QWidget*)	Constructor	_parent : QWidget	Default constructor of this class.
initializeGL()	Void	none	This function creates the window in the initial implemented method that could be called once when application starts.
resizeGL(_w :const int, _h : const int)	Void	_w :const int, _h : const int	This function is for organizing windows of application when it is resized. The width and height size of modified windows will be sent to this function to compute.
paintGL()	Void	none	This function is the main GL drawing routine.
mouseMoveEvent(_event : QMouseEvent*)	Void	_event : QMouseEvent	This function is called when the particular mouse button is moved. It will interact with the rotation of the scene.
mousePressEvent (_event : QMouseEvent*)	Void	_event : QMouseEvent	This function is called when the particular mouse button is pressed. It will interact with the rotation of the scene.
mouseReleaseEvent (_event : QMouseEvent*)	Void	_event : QMouseEvent	This function is called when the particular mouse button is released. It will interact with the rotation of the scene.
timerEvent (_event : QTimerEvent*)	Void	_event : QTimerEvent	This function counts the time of the program and controls the time counter.
createCameras()	Void	none	This function creates camera to the list with different viewing.
setCameraShape()	Void	none	This function set the camera value and the scene to be in the current state.
drawAxis(&m_shaderName : const std::string, &m_axisTransformStack :	Void	&m_shaderName : const std::string, &m_axisTransformStack :	This function draws arrow of the wind, which will head to the wind direction.

ck : ngl::TransformStack, m_scale : float, m_xRot : int, m_yRot : int)	ngl::TransformStack, m_scale : float, m_xRot : int, m_yRot : int

6. FirePath class

This class would calculate the fire growth rate and control the animation along the built DLA form. Fire path will be generated here with respecting to fire behaviors and variables, such as moisture, temperature, slope, wind and fuel object.

Variables Names	Types	Definitions
m_fireDLAForm	std::vector<FuelPoint>	List contains dla point, which it concerns as the head of fire point
m_fireDLALine	std::vector<FuelPoint>	List contains dla point, which it concerns as the tail of fire point
m_fuelOnGridList	std::vector<FuelPoint>	List contains position and variables of flammable object
m_originalFuelOnGridList	std::vector<FuelPoint>	List contains the initial fuel variables of flammable object
m_gridList	std::vector<FuelPoint>	List contains point on the grid or fire scene
m_originList	std::vector<FuelPoint>	List contains origin point
m_currentIndexForm	std::vector<FuelPoint>	List contains current position of fire path drawing function
m_burningPointList	std::vector<FuelPoint>	List contains burning dla points
m_timeStep	Integer	Current drawing time step
m_currentDLAPointIndex	Integer	Current drawing index
m_allShow	Boolean	Fire path drawing status that is drawn all of the dla point or not
m_rateOfGrowth	Float	Rate of fire growth
m_sceneMoisture	Double	Scene moisture
m_sceneTemp	Double	Scene temperature
m_objMoisture	Float	Fuel object moisture
m_pause	Boolean	Animation pause status
m_windDirection	ngl::Vector	Wind direction
m_windSpeed	Integer	Wind speed
m_windEnable	Boolean	Wind status

Methods Names	Types	Parameters	Definitions
FirePath(_allDLAList : std::vector<FuelPoint> , _allDLALine : std::vector<FuelPoint> , _fuelList : std::vector<FuelPoint> , _approxObjMoisture : float, _originList : std::vector<FuelPoint> , _windStatus : bool, _windSpeed : int, _windDirection : ngl::Vector)	Constructor	_allDLAList : std::vector<FuelPoint>, _allDLALine : std::vector<FuelPoint>, _fuelList : std::vector<FuelPoint>, _approxObjMoisture : float, _originList : std::vector<FuelPoint>, _windStatus : bool, _windSpeed : int, _windDirection : ngl::Vector	Constructor of this class. The fire path will be generated from the given dla list, respecting to moisture, wind speed, wind direction, and origin list.
drawDLAPath (&_shaderName : const std::string, &_transformStack : ngl::TransformStack)	Void	&_shaderName : const std::string, &_transformStack : ngl::TransformStack	This function draws fire path from the calculated from burning time.
updateDLAList(_allDLAList : std::vector<FuelPoint> , _allDLALine : std::vector<FuelPoint>)	Void	_allDLAList : std::vector<FuelPoint>, _allDLALine : std::vector<FuelPoint>	This function updates the dla point list whenever the new fire path is generated.
updateFuelOnGrid (_fuelOnGridList: std::vector<FuelPoint>)	Void	_fuelOnGridList: std::vector<FuelPoint>	This function updates the fuel object variables list.
resetFuel()	Void	none	This function resets the fuel object colour to initial colour.
getCurrentFuelList()	std::vector <FuelPoint >	none	This function returns the current fuel object variables list.
resetFuelListToOriginal()	Void	none	This function resets the fuel object variables to initial state.
timeUpdate(_time : int)	Void	_time : int	This function updates the fire path along the time.
resetList ()	Void	none	This function resets all the list to initial state.
controlFirePathColour ()	Void	none	This function controls the colour of fire path.
assignNewMoisture(_newMoisture : double)	Void	_newMoisture : double	This function adjusts the moisture value of fire scene.

assignNewTemp(_newTemp : double)	Void	_newTemp : double	This function adjusts the temperature of fire scene.
setFirePath()	Void	none	This function sets the dla path to be ready to fire.
assignNewObjMoisture(_newObjMoisture : float)	Void	_newObjMoisture : float	This function adjusts the moisture value of fuel object.
analyzeBurningArea()	Void	none	This function would analyze the current burning area with the fuel object and heat transferring.
setNewOriginList(_newOriginList : std::vector<FuelPoint>)	Void	_newOriginList : std::vector<FuelPoint>	This function updates the origin list whenever the new fire path is generated or adjusted the origin amount.
analyzeDLAAmount(_m_newTime : int)	Void	m_newTime : int	This function would analyze the current burning spot respecting to burning time of each fuel at the current time step.
resetGrid(_currentGridPoint : std::vector<FuelPoint>)	Void	_currentGridPoint : std::vector<FuelPoint>	This function updates the grid point.
pauseTheAnimation(_status : bool)	Void	_status : bool	This function toggles the animation.
transferHeat(m_gridIndex : int, m_heat : int, m_mode : int)	Void	m_gridIndex : int, m_heat : int, m_mode : int	This function transfers the heat to surrounding area.
calculateRateOfGrowth()	Void	none	This function calculates the fire spread rate.
checkCollisionWithBurningArea(m_currentPos : ngl::Vector, m_currentFormNumber : int)	Boolean	m_currentPos : ngl::Vector, m_currentFormNumber : int	This function would check collision with burning area with the current new born fire point.

APPENDIX B.

Program User Guide

Program User Guide

Program allows users to adjust several variables to see the different of fire path. The different fire growth behavior would be created from the different environment factors. The following variables are the necessary factor that users would be able to adjust the value.

1 Weather of area

1.1 Scene moisture

This variable would affect the fire rate of growth. Thus, whenever user wants to speed up the fire spread, user needs to decrease the scene moisture until it reaches the minimum value, which it is zero. Nevertheless, if user wants to see the slower fire spread, the scene moisture should be increase.

1.2 Scene temperature

This variable would work similar to the scene moisture variables. However, it gives the smaller effects to the fire scene when the single step is adjusted. In addition, if user adjust the temperature every 20 step, the moisture would automatically change with half of the latest amount moisture value because these two variables has related together, followed by the humidity rules in real world. For example, if the temperature has decreased from 40 to 20 degrees, the moisture would also be decreased from 20 to 10 percents.

2 Appearance and Camera section

These two sections do not relate to the fire growth prediction. They are just the viewing adjustment of the program.

The appearance section is for toggle the scene appearance from wire frame to texture mapped scene and toggle the origin mark showing status. Thus, if user does not want the mark of origins in the scene, user needs to uncheck the ‘show mark’ check box.

In the camera section, the different camera from different view will be available to choose the viewing of the program. Moreover, user can zoom in and zoom out the scene by clicking the zooming button.

3 Terrain enable

The terrain scene would be initially implemented when the program first runs. Thus, to see the different fire growth from the area with slope and flat area, user should disable the terrain function and click the generate button.

If the terrain is enabled, user would be able to adjust these following variables.

3.1 Terrain iteration numbers

This variable states to the iterations of terrain within the scene. More number of iterations, more mountain or terrain would be generated.

3.2 Random button

This button is for creating the new terrain scene with the same value of iterations number.

3.3 Terrain height increase and decrease buttons

User can adjust the height of terrain by clicking the increase (+) or decrease (-) button inside the terrain box. Thus, the terrain would be higher or lower than the initial creation.

4 Wind enable

The scene would add the wind function to be considered with fire path prediction, if the wind enable box is checked. If the wind is enabled, user would be able to adjust the direction and wind speed value.

5 Fire seed

This variable refers the origin of DLA structure that would represent as the fire path. The amount of fire seeds can be increased up to 5 points. Thus, the program would generate 5 fire path at a time. When the amount of origins has been adjusted, the new origin will be assigned

with the random position. Moreover, user can move the fire seed to the satisfied position by choosing the index of fire seeds and clicking the direction buttons. When the index number of fire seed has adjusted, user can notice which fire seeds is about to be moved in the scene by looking for the yellow fire seed.

6 Fuel object

This sections would adjust the amount and variables of fuel object, both flammable fuel and non-flammable fuel. Amounts of both fuel types can be changed. When user adjusts each fuel amount, program would clear the scene and randomly locates the fuel into the scene, respecting to the new assigned amount number.

For flammable object, user can select the specific object and adjust its size. Height and radius of the fuel object is available for user to adjust. Moreover, the moisture of the fuel object can also be changed. When the object moisture is changed, program would decrease the value of transparency, therefore, those fuel objects would have opaque surface, and vice versa.