
# Development of Three-dimensional Model of Avocado

A dissertation submitted to The University of Manchester for the degree of Masters in Advanced Computer Science in the Faculty of Science and Engineering.

2019

By

Danlin Wang 10288004

School of Computer Science

Dr Fumie Costen

# Contents

# List of Figures

# List of Tables

# Abstract

In the current day and age, with the deepening research on avocado, its extremely high nutritional value and great commercial potential have been discovered by various industries. Meanwhile, there are many different varieties of avocado available for consumers to choose from. However, due to its biological properties, avocado is very susceptible to disease. Currently, research teams start to develop devices that use electromagnetic waves to detect avocado disease but lack relevant data on avocado. The goal of this project is to establish a digital phantom of healthy or rotten avocados, providing reliable data for the detection device to accurately simulate how electromagnetic waves can effectively spread through the diseased avocados. In this project, models of different cultivars of healthy avocados and rotten parts of both diseases were established, and through the human-computer interaction interface, users can select different varieties of avocado models with different diseases, these models can be shown as the two-dimensional images through three directions. After performing functional tests and performance tests on the program, it can be concluded that most of the basic requirements of the project were well implemented, but there are still some details, functions and algorithms need to be improved.

**Keywords**: Avocado, three-dimensional model, Stem-end rot, Anthracnose.

# Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

# Copyright Notice

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialization of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/ DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/ library/aboutus/regulations) and in The University’s policy on presentation of Theses.

# Acknowledgements

I am very grateful to my family and friends for their support and dedication. It is their help that made me complete this wonderful adventure.

I am very grateful to the support and help of Dr. Fumie Costen. Throughout the project, she continued to provide me with guidance and assistance in all aspects.

# 1.Introduction

## 1.1 Problem Statement

The fruit we usually call avocado is the fruit of the plant Persea Americana (Mill), which may have originated in Mexico and is an evergreen dicot that belongs to the Lauraceae family [1][2][3]. The skin of the avocado is mostly green, and the fruit can be round, pear-shaped or oval. It is rich in nutrients, and, in the final appendix section, the nutritional value of avocado will be more specifically explored.

In recent years, avocado has received more attention, and the demand for avocados in various industries has increased considerably. But most avocado trees are grown in tropical and subtropical regions. Mexico is by far the largest avocado grower in the world, with annual production several times that of the second-largest producer (United States) [4]. Therefore, the transportation of avocado to international markets is essential. However, unlike other fruits, avocados are taken from the tree before they ripen [5]. This happens because avocados have a higher respiratory rate than other fruits, especially in the second half of their maturity period. As a result of the large increase in respiration rate, more ethylene is released, and the carbon dioxide and heat produced during respiration will accelerate the ripening of the fruit [6]. Therefore, suppliers often choose immature, hard avocados to transport and sell, while consumers want to buy ready-to-eat avocados to avoid buying rotten fruit and to be able to store them for a longer period time [7][8]. Therefore, avocado is a food that travels directly from farm to market. The properties of avocado's natural skin reduce the need for external packaging, and healthy avocados can be sent directly to market after harvesting, without the need for more processing, preservatives, flavor enhancers, and so on [9]. However, it is precise because of these characteristics of avocado that its transportation becomes complicated.

Avocado transportation companies must strictly control transportation time. If the storage time is too long, avocados will deteriorate and rot, affecting sales and causing huge economic losses. Also, the nutritional value of avocados is very high. Once avocados become infected with a disease, the economic impact could be devastating. Most of the pathogens of diseased avocados have an incubation period, and, usually, lesions will occur over a large area after the fruits are picked. Therefore, it is necessary to judge the health of avocados before transportation. Once the disease breaks out during transportation, the loss will be difficult to estimate. Because avocados are becoming so widely used, once there is a problem in the transportation process, it is not only the food industry, but also the cosmetic industry, the medical industry, and the avocado oil industry that bear the brunt of the losses. Consequently, a study of avocado health patterns becomes critical to inform on ways to prevent contamination of crops.

Our research team currently to develop a program that detects diseases in avocados by sending electromagnetic waves to the fruits before harvesting, thereby determining the state of decay of a crop and detecting disease by obtaining reflection signals from the fruits. To do this, it is necessary to be able to accurately simulate how electromagnetic waves can effectively propagate through the avocado. In order to develop algorithms for such wave detection devices, it is necessary to be able to receive signals either reflected or transmitted by avocados at different stages, whether healthy or diseased. Even though avocados have become so important to agriculture, a paucity of data exists on avocados and their natural cycle.

The aim of our study is to build a generate the three-dimensional model of avocado that will provide reliable measurements. Numerical simulations were used to generate data. The focus is to carry out the algorithms to create a digital phantom for those health, disease or decay of avocados, and then program a user interface that can extract the information entered by the user. For example,

the user chooses a disease and the extent to which the fruit rots after the illness, then the digital phantom of the avocado can be shown on the interface. The realistic geometry of the avocado in the digital phantom makes the simulation reflect reality. In this project, the digital models of five typical avocados will be created by using a simulation tool and will provide data on wave detection, and a basis for future disease detection, treatment and prevention.

## 1.2 Aim and Objectives

### 1.2.1 Aim

The aim of this project is to build a virtual model of an avocado fruit and use it for disease detection using algorithms generated by analyzing electromagnetic wave patterns emitted by the fruit. After investigating the biological characteristics of avocado, a three-dimensional model of the fruit was generated by using MATLAB software. Then, later the human-computer interface was established. On this interface, users can select different varieties of avocado, their common diseases, and degrees of avocado decay as variables. By doing so, the cross-sectional view of different states of avocados, in three different orientations, can be obtained.

### 1.2.2 Objectives

- Learn about the biological structure of avocados, including shape, thickness, seed and skin surface, by investigating a large amount of literature.
- An investigation of typical avocado decay patterns and the process by which they rot.
- An understanding of the image data structure and .pgm files.

- The development of algorithms to produce different kinds of healthy and diseased avocados, implementing them in MATLAB language to form a three-dimensional avocado model, which is represented by the .pgm images.
- The generation of a human-computer interface that allows the users to select images of avocados in different states.
- An evaluation of the levels of accuracy with respect to reality of the model generated and a test of program performance.

## 1.3 Organization of the document

There are six components to this document and the approximate content of each part is as follows:

Chapter 2 describes the background and significance of this project; 2.1 describes the biological characteristics of avocados, including the structure and different varieties of avocados; 2.2 lists several diseases encountered during avocado production, and a detailed analysis and elaboration of two diseases; Section 2.3 describes the image data structure and the PGM files. Section 2.4 tells what the three dimensions is specifically. And the Graphical User Interface (GUI) is outlined in Section 2.5. Only with sufficient theoretical knowledge and data can the avocado be simulated as closely as possible to reality, providing the foundation for realistic geometric models. Lastly, the chapter describes existing information about three-dimensional avocado models.

Chapter 3 details the algorithms used in this project: the algorithm simulating a healthy avocado, and two simulating the rotten parts.

Chapter 4 describes the algorithms above-mentioned algorithms and the implementation of the final human-computer interface, including the

implementation of the programming language, the code and the human-machine interface.

Chapter 5 tests (functional testing and performance testing and evaluates the results by contrasting the resulting images and the realistic avocado picture, thereby evaluating the performance of the software.

Chapter 6 summarizes the entire project and the test results and discusses recommendations for the future development of this project.

# 2. Background

Avocados, which used to be called vegetable butter, midshipman's butter, or as butter pear, are the only important edible fruit of the Lauraceae family.[10] As its name suggests, avocado has played an important role as a food in the past. Also, an environment suitable for growing avocado trees can be found in almost all tropical and subtropical regions [10]. It is precisely because of its popularity that the demand for avocados in different industries around the world is rising. In recent years, with the continuous exploration of avocado as an important crop, it has been found to have a very high nutritional value. Consumers in many countries have regarded it as an indispensable part of their diet in their daily lives. Not only that, it is rich in a variety of fats and has many unexpected uses in cooking and in cosmetic industries [11].

This project provides an interesting solution to curb the economic loss caused by avocado diseases, therefore it is important. The building of a three-dimensional avocado model provides basic data for future research. Before entertaining a three-dimensional model for avocados, the geometry of the fruit, the types of disease it is prone to, and the pathological processes involved in the disease need to be properly understood. Only by doing so, a more accurate model can be built. In the following section, the physiological structure and the known diseases of avocados will be described in detail.

## 2.1 Healthy avocado

### 2.1.1 Physiological structure

Generally speaking, all fruits can be divided into two categories, one is dry and the other is fleshy. Avocados belong to the latter. The avocado fruit is actually the mature ovary of the flower [12] (Figure 2.1 A cross-sectional view of avocado).

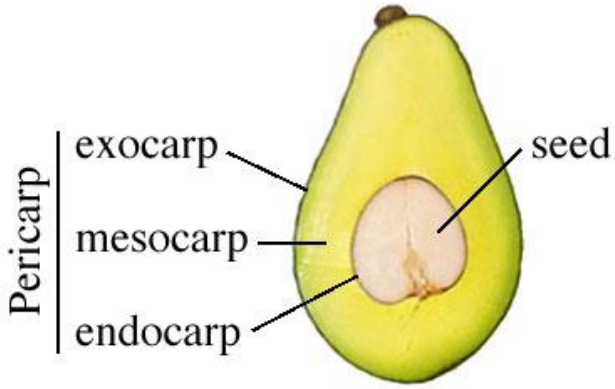



Figure 2.1 A cross-sectional view of avocado

As seen in figure 2.1, the peel of an avocado is usually composed of pericarp (oval wall) and a wrapped seed, the weight of which can account for 10%-15% of the total weight of the fruit. The pericarp is divided into three parts, exocarp, mesocarp and endocarp. The exocarp is often called skin, the mesocarp accounts for a large part of pericarp, and the flesh we often eat is the mesocarp. The endocarp wraps the seed. The endocarp of other fruits may be soft or fleshy, but the endocarp of an avocado is very tough [12]. Because of this structure, avocados are easy to peel and convenient to eat.

### 2.1.2 Cultivars

Avocados may be produced all over the world. However, due to the different climates in different regions, the avocados grown in different places can also belong to different varieties. They can have different shapes, textures, tastes, colors, and even odors. In general, the shape of common avocados is mostly round, pear-shaped and oval-shaped. The colors of their skin can be divided into black, purple, red-purple and yellow-green. [13]. Although there are many cultivars that have been identified, more than 500, some of these varieties may take a long time to mature, the amount of protein or fat that can be detected in the fruit is very low, or, because of some characteristics, it is difficult to store them during transport and they are easily damaged. Therefore, not every

cultivar is suitable for commercialization, and most of them have not yet been marketed [14]. Some of the existing varieties are discussed below.

- **Hass**

Figure 2.2 Hass Avocado

This is the most common avocado variety on the market. According to a survey, among the avocados cultivated around the world, the Hass variety account for about 80% [15][16]. It is a fruit native to California, green when immature, black after ripening, with moderate thickness, narrowly obovate to obovate, with an average weight of 140-340 ounces. This is a hybrid type from Guatemala. It can withstand temperatures of -1 ° C (30 ° F).

- **Bacon**

Figure 2.3 Bacon Avocado

Bacon's place of origin is also California. The fruit is medium in size, but the seeds are large, the skin is green when immature, and it remains green after maturity, but the color becomes darker, the skin is thin and oval, and the flesh tastes very light. The average weight is 255-570 ounces.

- **Fuerte**

Figure 2.4 Fuerte Avocado

Its name comes from the fact that it survives after experiencing frost. It can still survive at -3 ° C (27 ° F) [19]. Fuerte's fruit is medium in size and is pear-shaped. It is green before and after maturity. The skin is easily peeled off, the skin is moderately thick, and the average weight is 255-455 ounces. The trees are 6 by 4 meters in size [14][17].

- **Gwen**

Figure 2.5 Gwen Avocado

Gwen is produced in California and is more productive and smaller than the Hass variety. The color is always green, which could be deepen after maturity. The fruit is oval in shape and the seeds are medium in size with an average weight of 170-425 ounces. The flesh has a rich nutty taste.

- **Lam Hass**

Figure 2.6 Lam Hass Avocado

The origin of Lam Hass is California, and this is a hybrid. The fruit is obovate, the seed is moderately sized, the skin is pebbly, and black in color before maturity, with an average weight of 280-510 ounces [14].

- **Pinkerton**

Figure 2.7 Pinkerton Avocado

Pinkerton was grafted using the varieties Hass and Rincon and planted in Saticoy, California in the early 1970s. The green skin deepens with maturity,

the flesh has a creamy texture, the fruit is pyriform, the core is small, and the average weight is 255-511 ounces [14][17].

- **Reed**

Figure 2.8 Reed Avocado

This variety comes from California and there is a variety from Guatemala, with large seeds, large and round fruits, green skin and moderate thickness. The flesh is delicate and has a nutty taste. The average weight is 480-680 ounces.

- **Zutano**

Figure 2.9 Zutano Avocado

Zutano is of the Mexican type and is produced in California. Its fruit is obovate, the skin is yellow-green, thin and smooth, the seeds are medium in size, the flesh is light, and the average weight is 310-400 ounces [14].

## 2.2 Rotten avocado

Just like other tropical and subtropical fruits, avocados are plagued by various diseases. The prevention of diseases is essential to ensure the economic success of avocado growers, and new technologies focuses of improving crop yield and disease prevention are critical. Only by fully understanding the type of diseases and its root causes can scientists develop a mechanism of prevention in a targeted manner.

Avocado is very susceptible to viruses, bacteria, and fungi, even such organisms can even trigger the loss of some key nutritional factors. Diseases can manifest on avocado flesh, as rot, discoloration, ulcers and spots [18]. In general, the diseases that avocado often suffer can be divided into four types. Bacterial diseases, Fungal diseases, Virus like diseases, Miscellaneous diseases and disorders. The corresponding representative of each disease type is listed in Table 2.1.

Table 2.1 The types of diseases in avocados [19]

| **Disease** | **Typical representative** |
|---|---|
| Bacterial diseases | Bacterial canker<br>Blast and bacterial fruit spot<br>Crown gall |
| Fungal diseases | Anthracnose<br>Fruit rot (includes stem end rot & fruit spots)<br>Rusty blight |
| Virus like diseases | Sunblotch<br>Trunk pitting |
| Miscellaneous diseases and disorders | Dieback<br>Tipburn<br>Blackstreak |

The disease may manifest in different ways often altering the appearance of the fruit at maturity. The fruit may develop depressed spots, deteriorate as the disease progresses. Also, black lesions will appear on the leaves of the plant

[20]. Because this project only models the fruit, other parts of the avocado plant, such as roots, stems, and leaves, will not be discussed in detail.

When an avocado plant is affected by a disease, the fruit does not have obvious lesions. Only when avocados ripen to maturity, the symptoms will be evident. In general, the fruit will form a large, concave rotten area, and the color will change from normal to dark brown or even black. When the disease develops, the decaying part of the fruit begins to extend from the epidermis to the flesh, and eventually the fruit will crack [21]. Therefore, when a disease spreads in a crop, and is not controlled on time, it will cause serious economic losses. Also, because diseases may not be easily detectable before the fruit matures, fruit growers cannot judge the health of the fruit at the time of harvesting, and some diseased fruits may look fresh and healthy before transport and storage [21]. Therefore, some fruits that look fresh and healthy will develop lesions and decay during transportation or storage, which highlights the need for this project. Two common avocado diseases were selected for modeling, namely Anthracnose and Stem-end rot. The following content will introduce each disease separately, and the appearance of diseased avocado will be shown.

### 2.2.1 Stem-end rot

As mentioned earlier, avocado is susceptible to a variety of diseases, of which Stem-end rot (SER) is one of the most common. According to a study by Sanders and Korsten [22], nearly 13% of South African avocados exported in 1983 were affected by SER.

SER is caused by a series of fungal pathogens [23]. The main route of infection for these fungi may be avocado flowers [24]. When avocado flowers are infected, these pathogens spread through the stems of the plants, and the fruit is infected through the stem ends during the early growth phase but shows no abnormalities. Symptoms of short rot of the stem begin to appear when the fruit matures [23]. Therefore, if the relevant personnel want to prevent this disease,

they must discover it and intervene before the fruit matures. According to a study by Demoz and Korsten [25], if Bacillus subtilis B246 is sprayed on avocado before fruit picking, the pathogen can be suppressed in situ, and the antagonist to the pathogen will invade the flower. Then, the rate at which avocados continue to rot will be slowed. This treatment is currently used on both a commercial and semi-commercial scale [26]. Generally, as the fruit matures, the decaying area begins to spread and gradually rot invades the entire fruit, tuning it dark and making it shrink (Figure 2.10). The decay of avocado can be roughly divided into three stages.

- Early stage: only slightly rot in the head.

Figure 2.10(a) Early stage of SER

Figure 2.10(a) Early stage of SER

- Mid stage: the stem rot is aggravated, and a slight wedge-shaped decay (xylem) has changed from the bottom. (Figure 2.11).

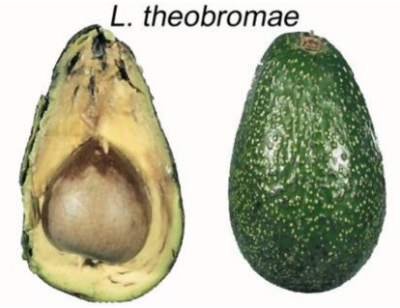



Figure 2.11(a) Mid stage of SER

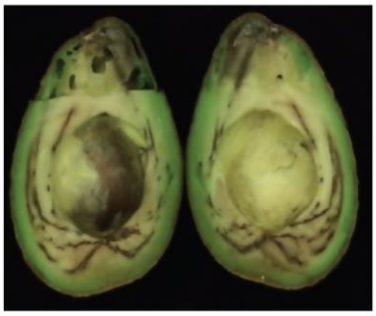


Figure2.11(b) Mid stage of SER

- Late stage: the head and tail rot are connected, the core rots (Figure 2.12).

Figure 2.12 Late stage of SER

### 2.2.2 Anthracnose

Anthracnose is caused by pathogens that infects the leaves and fruits alike [27]. The pathogen is Colletotrichum gloeosporioides [28]. According to a study of South African avocados exported in 1983, 36% were infected with Anthracnose [29]. In an article, Mabbett [27] mentioned that healthy fruits cannot be infected with Colletotrichum gloeosporioides during storage, transportation and marketing. In other words, avocados are infected before harvest. As long as

the fruit is not picked, the avocado can be infected at any stage of growth. When the fruit is still on the tree, the incubation period of the bacteria can reach several weeks or several months, so Anthracnose has a "secret life", and the infection can continue until after the avocado is harvested.

Avocados, which generally receive insect bites or damage (For example, during the picking process, the picked tools may damage the skin, or during the transportation process, the avocado may collide with each other due to the bumps in the road), are more susceptible to Anthracnose. After being invaded by the pathogen, the affected area of the fruit will gradually expand and spread. Before the avocado is harvested, the skin will naturally crack and produce some lenticels, so that the plants are also susceptible to disease, and some small, dark spots appear around the opening. In addition, for those avocados that remain intact, the pathogens will remain in a latent state until the fruit is fully mature and harvested, without any signs of disease. Once picked, the fungal pathogens in the fruit will continue to grow, killing the pulp cells of the avocado, and manifesting as a large black necrotic area. This disease seriously affects the quality of the fruit during sales, and due to the influence of the fungal pathogen, the shelf life of the product is greatly shortened, affecting its marketability [30].

Anthracnose progresses as follows: at the beginning, the lesion area has a diameter of < 5 mm, a circular shape, a slight depression, and a brown or black color. When the condition is aggravated, the lesion will expand rapidly, the sag becomes obvious, and there may be radial cracking in the center of the lesion (Figure 2.13). Eventually, all lesions merge (Figure 2.14).

Figure 2.13 Schematic diagram of radial cracking

Figure 2.14(a) Avocado suffering from Anthracnose


Figure 2.14(b) Avocado suffering from Anthracnose

## 2.3 Image data structure

A 3-D model of an avocado (the fruit) was created and stored using a Potable Gray Map (PGM) file format. PGM is a simple image format, which represents two-dimensional data in gray scale, and is used for storing and exchanging image data. The 3-D model of an avocado was created using Netpbm, an open-source package of graphics programs and a programming library. [31][32]. Because Netpbm's resources are open to everyone, it can be found and used in almost all open source operating systems. However, in some closed source operating systems, it can also be found. For example, Microsoft Windows and macOS [33]. To be more specific, if a graphic object is generated by using Netpbm, its format is called Netpbm formats. There is not only one type of Netpbm formats, it is divided into two categories, one is PAM (Portable Arbitrary Map) format, which is relatively complicated and cumbersome; the other is PNM (Portable portable anymap format), sometimes Also known as PBM format, it contains three formats: PBM (portable bitmap format), PGM (portable graymap format), PPM (portable pixmap format). The existence of these three formats facilitates the image of different file formats. Different platform nails can be converted at will. In this project, in order to distinguish different parts of avocado fruit with different gray scales, the PGM format is used to generate picture objects.

The PGM format is also the lowest common denominator grayscale file format [34]. A PGM image represents a grayscale graphic image, and the PGM file represents one or more files of the grayscale image saved in the PGM format [31]. There is no other data, separators, or other padding before, after, or between these images [34]. It contains header information and a numeric grid that can represent multiple values for shades of gray, from black (0) to white (maximum 65,536). There are two storage formats for PGM files: ASCII text, which is the most frequently used format, and binary [31].

The following figure (Figure 2.15) shows an example of a PGM-format picture. As illustrated by the picture, the PGM format file can easily display two-dimensional pictures using different shades of gray.


Figure 2.15 Example of a PGM-format image

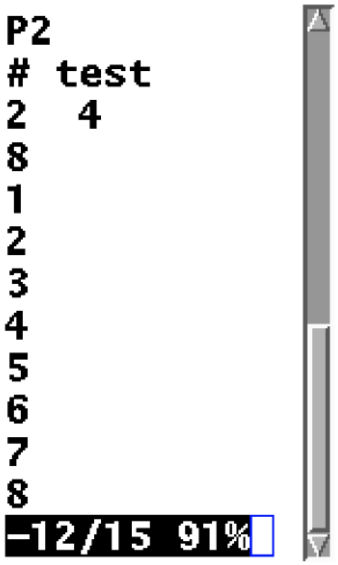



Figure 2.16 A file in PGM format

A file in PGM format is illustrated in Figure 2.16. The first line, P2, indicates that the data is in PGM format when using ASCII text encoding, and this is called the " magic number", which is two bytes in size, and it is used to distinguish the type of file. When Binary is used, the magic number at the beginning of the file would be P5. In the binary format, 8 bits are required for each pixel [35]. At the same time, Magic number can also be expressed in ASCII formats. When using this format, it is more convenient for humans to read, highly readable, and more easily converted to other platforms, represented by other forms. In the third row of numbers in figure 6, "2" indicates that the x-axis has two pixels, "4" indicates

that the y-axis has 4 pixels, and, in the next row, "8" indicates that the largest valid value in the data is 8. The rest of the data is the actual data for each pixel.

## 2.4 Three-dimensional space

In daily life, all objects in the surrounding environment that can be picked up, moved and touched are three-dimensional, and all three-dimensional objects can be rotated in space [36]. The most significant difference between three-dimensional and two-dimensional objects is that the former has one more dimension than the two-dimensional object: depth. The three-dimensional space is in fact a geometric setting. In three-dimensional space, the position of a point must be determined by three parameters. If the two-dimensional coordinate system $(x - y)$ is extended and a z-axis is added, the vectors in these three directions will not be in the same plane (two-dimensional) at the same time, and therefore a three-dimensional coordinate system can be created; this is illustrated in Figure 2.17.

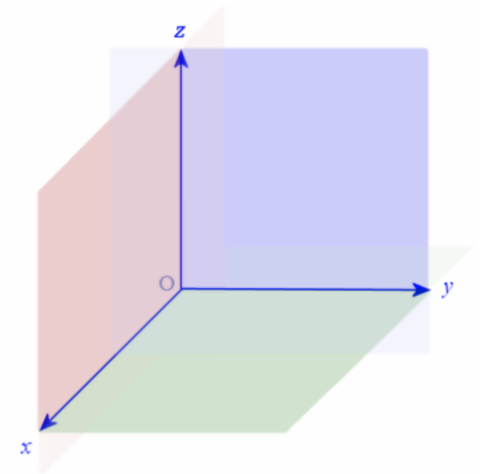



Figure 2.17 Three-dimensional coordinate system

In this way, the three dimensions of width, height and depth can serve to describe the objects in the three-dimensional space.

## 2.5 GUI

GUI is an abbreviation of graphical user interface. This offers the user an interface for operation by displaying graphics on the computer display. In other words, the GUI provides a human-computer interaction interface. Previously, if the user wished to modify the data selection, this had to be done by using the command line, which is undoubtedly unfriendly to users without a basis in coding. However, once a graphical interface such as a GUI appeared, the range of users that can be operated became broader, and it became more visually acceptable to the user. In the graphical user interface, the user's actions are performed on the window. The user’s basic operation on the window is done using the mouse. The user can clearly see where the mouse pointer is and what options the program can provide for visual operation.

After creating the GUI in this project, it is possible for the user to can select different varieties, diseases and decay states on the window with a simple click of the mouse to view the profile of the avocado in three different directions. Its buttons and their combined options are equivalent to different parameters. When the user clicks the mouse and makes a selection, corresponding parameters are passed to the program, and it is instructed to deliver various calling functions or provide different function equations. In this way, the parameters thus generate different avocado profiles.

By investigating and analyzing the relevant literature, it is clear that many studies have been conducted into the various aspects of avocado, such as its nutritional value, variety and various diseases. Meanwhile, the related technology for establishing the three-dimensional model of avocado has also been known. However, there is currently no data regarding the geometry of many avocado fruits. The success of this project may help to enhance this area of research in the future.

# 3. Algorithm

Through the introduction of the second section, different varieties of avocado and different health or disease states are clearly mastered, and three-dimensional models can be prepared. This section outlines the foundation for the completion of the entire project: the algorithm. By reviewing and surveying the existing literature, the geometry of the five varieties of avocado can be clearly understood; furthermore, the realization of all decay shapes is based on healthy avocados. The overall idea of implementing the project will be roughly described and the algorithms employed during the different stages of the project are described in detail below. They relate to the geometry of the fruit and the score of the different varieties, the realization of the rugged peel, the realization of the Anthracnose which decays in block form and the Stem-end rot which decays in strip form.

## 3.1 Mathematical model of healthy avocado

Three-dimensional modelling of healthy avocados provides the basis for the entire project. To describe the avocado in MATLAB software to achieve a more realistic depiction, the avocado must be mathematically modelled. Whether the 3D model can be successfully established and established is similar to the avocado in real life. They all depend on whether the mathematical model is established successfully and whether the algorithm of the model is appropriate. When we want to use mathematical concepts and language to describe a system more accurately and rigorously, mathematical models can be used to achieve this. In other words, the mathematical model can also be understood as: When the internal law of a particular object in the real world is discovered, this law can effectively distinguish it from other objects. To achieve some of the necessary goals, we can make assumptions about the unique characteristics of the law and then, based on those assumptions, borrow the relevant appropriate mathematical tools to obtain the mathematical model of the

object.[37] The entire process of developing mathematical models is referred to as mathematical modelling [38].

Mathematical modelling can be described as an art which transforms complex problems in the application domain into a series of easy-to-handle mathematical formulas. Developing mathematical models also helps people to understand the objects being studied more clearly and in greater depth, which makes some complicated problems in real life more regular; this is advantageous to researchers and more conducive to related research and work. Mathematical models are important in all areas of science, including archaeology (reconstruction of original objects based on retained debris reconstruction or classification of ancient art), architecture (virtual reality, drawings), astronomy (detecting planetary systems, observations, the trajectory of the planet, studying the origin of the universe) and linguistics (automatic translation). Mathematical models are indeed very closely linked to our everyday life. As Figure 3.1 shows, the mathematical model is closely related to the real model. [39]

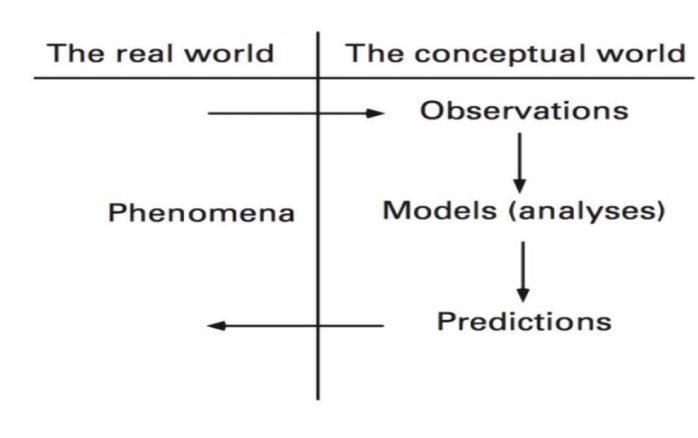



Figure 3.1 Depiction of the scientific mathematical model

Indeed, mathematical modelling does not simply involve mathematically expressing real things. It requires certain principles to be followed, and some of these basic ideas form the foundation and core of the mathematical model. The essential points are defining which object to model and what criteria and

needs must be met for the establishment of this model; establishing what information and data needs to be obtained about the modelled object for the model to be successfully established; using the acquired information to judge what data is beneficial for modelling and what kind of scenes the data can be applied to; determining the method of the govern model; and organizing the mathematical equations to be used in this model and performing the corresponding calculations to get the answers. Assuming that the model has been established, it is necessary to determine some methods to verify whether the it adheres to all the assumptions made to it before the model is built, whether it meets all the requirements before it is established, and if not, whether it corrects and modifies the defects, retests the verification, modifies any remaining defects, forms a loop until the model is perfect and then judge and determine the usable conditions of the model.[40]

“Mathematical models provide insights, answers, and guidance for theoretical and numerical analysis that are useful for the original application. [39]” This project seeks to treat a variety of different avocados as mathematical models. In order to develop these mathematical models, it is necessary to pass multiple sets of variables and the complex relationship between the variables, which is the algorithm mentioned below, in order to describe the process involved in building the entire model. Once the avocado has been mathematically modelled, its structure can be understood more clearly and comprehensively, and computing power can also be used more effectively in the process. [39]

The following figure is a flowchart of mathematical modelling (Figure 3.2). Each node represents various information including what to collect, what to sort, what to evaluate and what to organize. The connection between the nodes indicates the two-way communication between various information. [39]

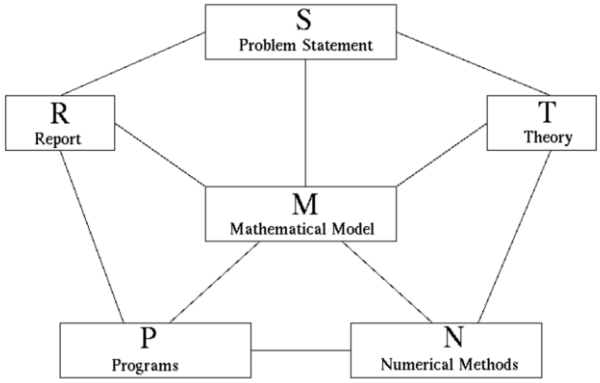



Figure 3.2 Modelling diagram

In this project, the information gathered from the existing literature is linked to the information of theory and problem statement. The mathematical methods and information of the programs to must be considered in order to perform mathematical modelling.

When investigating the geometry of avocados, it was found that regardless of the variety of avocado, the shape of the whole fruits and the scores are composed of a sphere or ellipsoid or a combination of these two mathematical geometry models.

There are many varieties of avocado, but this project is limited in terms of time. As such, only five varieties of avocado were modelled. Figure 3.3 illustrates that the fruit and score of Reed can be basically determined as the shape of a sphere, while the geometry of other varieties of avocado is composed of a sphere and an ellipsoid. From Gwen to Zutano, the ratio of length to width increases from left to right.

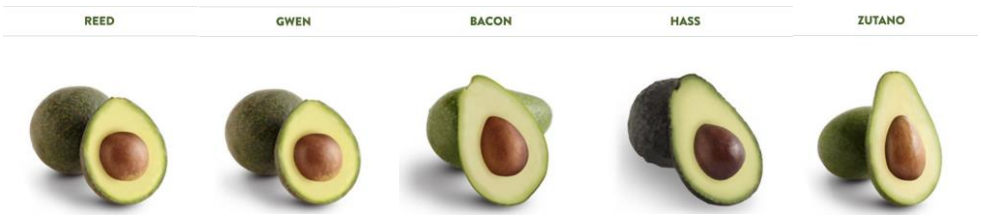



Figure 3.3 Five varieties of avocado

In the following sections, information and knowledge points about sphere and ellipsoid will be described in detail.

### 3.1.1 Sphere

The word “sphere” comes from the Greek σφαῖρα, which means “ball” in English. [41] A sphere is described as “a perfectly round geometrical object in three-dimensional space that is the surface of a completely round ball [42]”. In mathematics, a sphere is defined as the situation where, given a point, the set of points in the three-dimensional space with a distance from that given point constitutes a shell of sphere [43]. Within this, the given point $o$ is the center of the sphere, and the distance to the center of the sphere $r$ is the radius of the entire sphere.

In the process of completing the project, the sphere can be used to implement the Reed species, while the mathematical models of other varieties also require the participation of the sphere. The avocado is not just a spherical shell; the flesh of the fruit must also be reflected, so when seeking to draw a solid sphere, it is necessary to control the number of the sphere radius between $0$ - $r$. Of course, this solid ball could also be imagined as a three-dimensional object formed by a circle, with the center point as $o$ , rotating 360 degrees with an arbitrary diameter as the axis in a three-dimensional space. Figure 3.4 is a three-dimensional graphic of the sphere.

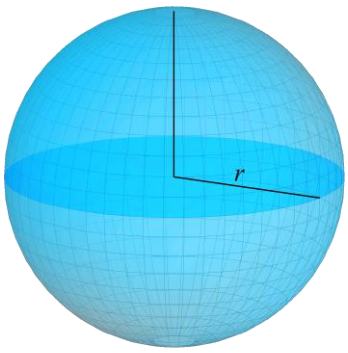



Figure 3.4 Three-dimensional graphic of a sphere

In MATLAB, equations can be used to draw spheres. The mathematical expression of a sphere in the Cartesian coordinate system is as follows:

$$(x - x_{centre})^2 + (y - y_{centre})^2 + (z - z_{centre})^2 \leq r^2$$

In this equation, $(x_{centre},\ y_{centre},\ z_{centre})$ is set as the center of the sphere, and r is its radius. As previously mentioned, since the avocado is solid, the interior of the sphere also needs to be expressed, so the distance from the point to the center of the sphere $(x - x_{centre})^2 + (y - y_{centre})^2 + (z - z_{centre})^2$ is less than or equal to the radius r.

### 3.1.2 Ellipsoid

“An ellipsoid is a quadric surface; that is, a surface that may be defined as the zero set of a polynomial of degree two in three variables. [44]” A quadric surface can also be defined clearly: it is based on a three-dimensional extension of some conic curves, including (ellipses, parabolas and hyperbolas) [45]. At the same time, the equations of the graphs belonging to the quadric surface can be placed in the following general form of equations [46]:

$$Ax^2 + By^2 + Cz^2 + Dxy + Exz + Fyz + Gx + Hy + Iz + J = 0$$

Each letter in the equation, from A to J, is defined as a constant.

Given that an ellipse belongs to one of the quadric surfaces, it contains all the features of a quadric surface. First, each cross-section is elliptical or circular, or sometimes even a point [47]. Figure 3.5 illustrates that the figure on the three-dimensional coordinate axis is an ellipsoid in which the cross-sectional areas A, B, and C are both circular or elliptical.

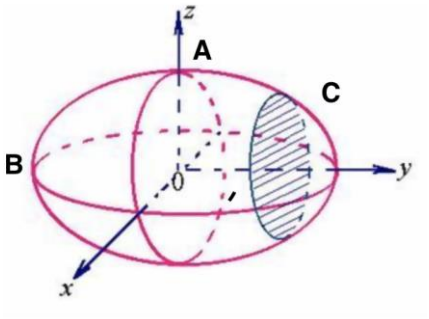



Figure 3.5 Ellipsoid with cross-sections

Second, it can be enclosed in a sufficiently large sphere. [45] Figure 3.5 also illustrates that if the ellipsoid is placed in a Cartesian coordinate system, it will have three pairs of symmetry axes perpendicular to each other in the figure, which are the $x$ axis, the $y$ axis and the $z$ axis respectively. The symmetry center point at which the three axes intersect is also the center of the ellipsoid. Further, the line segment that divides an ellipsoid into two parts on a coordinate axis can be described as the principle axis, or simply an axis of the ellipsoid.

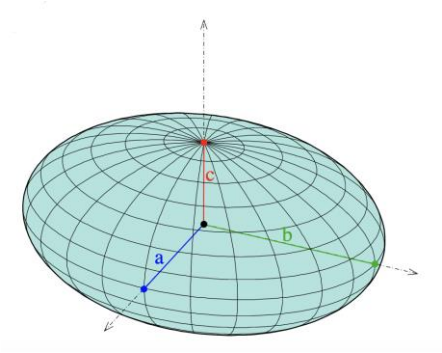



Figure 3.6 Tri-axial ellipsoid

As shown in Figure 3.6, $a$ , $b$ and $c$ are the three different axial lengths of an ellipsoid. An ellipsoid like this, i.e. with three different shaft lengths, can be described as a tri-axial ellipsoid. The axial direction of this kind of ellipsoid is not arbitrarily convertible; this means that the three axes are unique and fixed. [44] However, when two of the three axes are of equal length ( $b = a$ ), for example as illustrated in Figure3.7, the ellipsoid is called a spheroid. In this

case, the appearance of the ellipsoid does not appear to change from all angles as long as it rotates around the only different axis with an axis length of c.

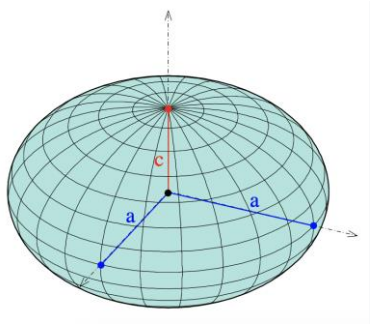



Figure 3.7 Spheroid

If the value of a is less than c , then this will be an oblate spheroid; conversely, if the value of a is greater than c , then it will be a prolate spheroid. In addition, when the value of a is equal to c , as shown in Figure 3.8, it will be a sphere, which is the shape previously mentioned in 3.1.1.

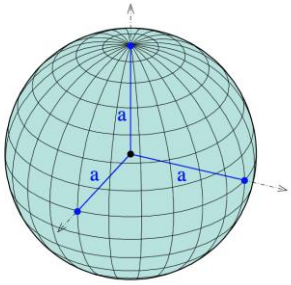



Figure 3.8 Sphere

If the Cartesian coordinate system is used, the system is coincident with the three axes of the ellipse and the coordinates of the center point of the ellipsoid $(x_{centre}, y_{centre}, z_{centre})$, then the mathematical expression of the ellipsoid is:

$$\frac{(x - x_{centre})^2}{a^2} + \frac{(y - y_{centre})^2}{b^2} + \frac{(z - z_{centre})^2}{c^2} \leq 1$$

The aim of this module is to create a 3D model. The basis of said model came from a pixel matrix, in which grey levels within a pixel matrix gave evidence of the pictured avocado's details. The general process of the project is as follows:

the first stage involved creating a three-dimensional model of a rectangular parallelepiped of 300*300*200 pixels and setting its initial grayscale to pure black. Then in the middle of the cuboid, a mathematical model of the ellipsoid and sphere is used to depict the avocado's peel and pulp. Here, differences are picked out by various grey values. From a logical point of view, at the beginning of the program, after a series of parameters were initialized, the approximate model of the fruit and its internal rotten area were determined. The three-dimensional geometric shape was then refined by a random generation algorithm. Each pixel in the resulting geometric three-dimensional image is then painted to distinguish the different parts of the avocado. Essentially, it was necessary to create a mental image of the shape of the avocado and the rotten area and its distribution, and then use this to form a model and add color to it. The process is similar to that involved in building an object in real life. The algorithm for implementing each part is described in detail below.

## 3.2 Flesh and score

Mastering the equations for sphere and ellipsoid, a healthy three-dimensional model of an avocado can be drawn in MATLAB. For the Reed variety, the fruit’s core and flesh can be treated, in the most direct possible fashion, as two spheres. If one selects the center point of the cuboid as the center point of the ball of the two spheres, this is completely logical. The sphere with the smallest radius of two acts as the score of the Reed, and the slightly larger sphere acts as the pulp portion of the Reed, as shown below (Figure 3.9).

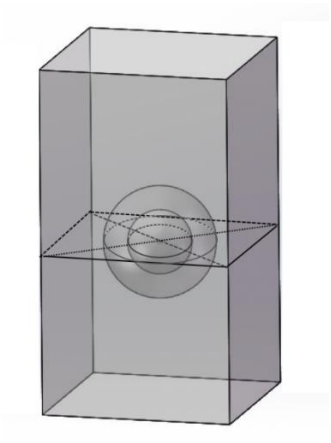


Figure 3.9 The flesh and score of Reed Avocado

The next step is to use the method described in 3.3 to draw the skin for the Reed. After this, a believable and quite accurate 3D model of the Reed avocado can be built. For other varieties of avocado, geometric accuracy may not always be achieved with just one sphere. This would therefore require the participation of an ellipsoid instead. For example, the Bacon variety's (see Figure 2.3 for the physical map) geometry can be similarly used shown with combination of a sphere and an ellipsoid.

Figure 3.10 The geometry of Bacon

As illustrated in Figure 3.10, the entire fruit shape of Bacon can be graphically shaped into a pattern as shown. Alternatively, if seeking to draw Bacon in equations in MATLAB, the ellipsoid and sphere can be combined to achieve the same purpose. To do this, the cuboid must be divided into upper and lower parts, the center point of the cuboid is selected as the origin, the half of the ellipsoid is drawn in the upper part and the half of the sphere could be drawn in

the lower part. The equations of the hemisphere and the semi-ellipsoid should always be the same as those of the sphere and the ellipsoid respectively. This is true except that the hemisphere and the semi-ellipsoid add various restrictions to the values in the height direction of the shape generally when drawing. Specifically, it is assumed that the z-axis is the height axis of avocado and the positive direction represents what is above the avocado. Next, when there is $z > z_{center}$ , the ellipsoid equation is used, and when there is $z \leq z_{center}$ , the sphere equation is used. Thus, the upper part of the avocado is ellipsoidal, and the lower part is spherical. The same method can be used to generate the score of Bacon. The 3D model drawn at this time is shown in the Figure 3.11.

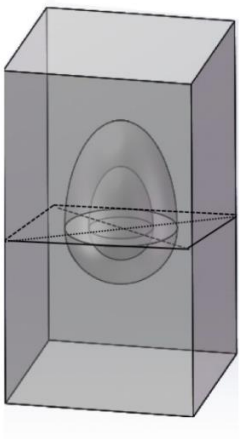


Figure 3.11 The flesh and score of Bacon Avocado

The algorithm in Section 3.3 can then be used to generate a peel for Bacon. Following these steps will allow a user to depict the geometry of the avocado fruit of the Bacon species. As for Zutano, Hass and Gwen, the modelling methods for these three varieties of avocado are the same; the only difference is that the sphere radius of the sphere differs from the axial length of the ellipsoid. According to the different characteristics of each variety, the three-dimensional model of the health of each variety can be easily obtained by changing the values of radius and axial length.

## 3.3 Skin

The thickness of the acre of the avocado cannot be ignored. As in most fruit, the color of the peel is always different from the color of the flesh. It also must be noted that the avocado's skin is not smooth, as shown in Figure 3.12.

Figure 3.12 Avocado skin

The skin of the avocado is pitted, with a variety of different sized bumps across its surface. If the model were to treat the skin as a smooth curve, then the established 3D model would differ greatly from the real thing. In order to simulate the uneven texture of the avocado peel, other algorithms need to be considered. Through observation, and querying the relevant data, the geometry of the peel can be illustrated as a shape similar to Figure 3.13.

Figure 3.13 Mathematical model of the peel’s textured surface

The Overlapped Sphere Model can be used to achieve a result that reflects the target model. The overlapping spheres of various large and small sizes can realistically simulate the intricate bulges on the avocado peel. The specific implementation steps are as follows. The exocarp and score were drawn to

simulate the avocado employing the algorithm described in Section 3.2. Next, using the same method from Section 3.2, the sphere radius and an ellipsoid axis length slightly smaller than that of the exocarp were selected to map the peel area of the avocado. For the avocados adhering to the Reed and Bacon variety, the standard three-dimensional models were obtained. The Overlapped Sphere model was then used. In the peel section, using the Randi() function that comes preinstalled with MATLAB to select random numbers, random points were determined in the three-dimensional space, and the spheres drawn with these random points as their centers. The radius of the sphere was also selected by the Randi() function. The value of the sphere radius was controlled within a selected range, and the Randi() function is used to select and draw each sphere radius within a determined range of values. This process is depicted in Figure 3.14.

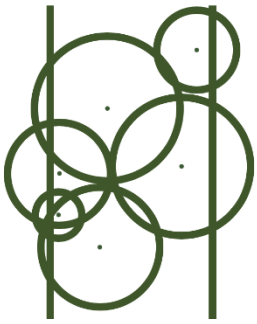


Figure 3.14 The principle of skin drawing

This process of drawing small spheres with various dimensions was continued until said spheres took up the entire skin area. At this point, the uneven texture of an avocado was achieved and appeared on both sides of the peel area. After successfully building a three-dimensional model of a healthy avocado it is possible to begin coloring areas. First, the pixels of all the small spheres are painted with 0.5 (dark grey). The flesh area was then filled with light grey, and finally the grey level of the score part was adjusted to 1, white.

The reason for coloring in this order is because only the outer skin of avocado is rough. The inside of the fruit, on the other hand, is smooth and flat. For this

reason, the peel should be colored first and then the flesh. Once the small balls generated by the random algorithm appeared in the flesh, it was covered by the second fill color and became light grey. This method ensures that the outer skin of the avocado is rough, and the inner epidermis is smooth every time the operation is undertaken. The figure below shows the process of filling in a small piece of peel.

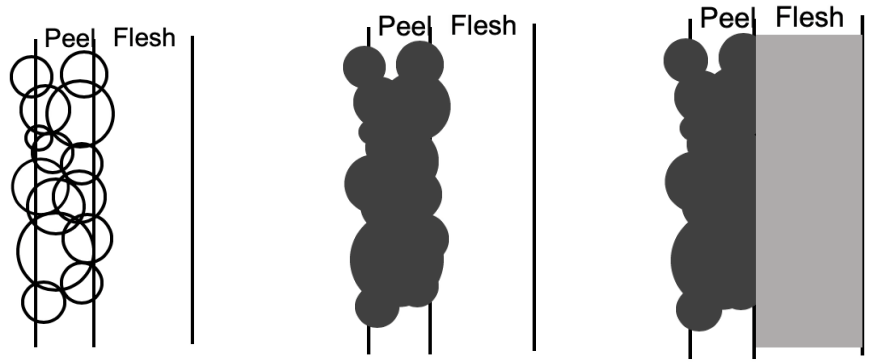



Figure 3.15 Peel colouring process

## 3.4 Rotten part

After building five varieties of healthy avocado, two types of disease can be modelled. Based on previously gathered information, it can be determined that the most obvious difference in the avocados after suffering from two diseases is that their decay patterns are different. Therefore, the most prominent feature of modelling a diseased avocado is to distinguish the decay model of the two diseases. It is worth noting that although the decay pattern of the same disease is virtually identical, the diseased area of each diseased avocado is not. To simulate the abundance of avocados in a more realistic manner, the computer's pseudo random number generator (PRNG) is mainly used in both disease classifications. By using this algorithm, the results of each run of the project will be different, and so too will be the specific shape of the decayed part and the size. The following section will provide a detailed description of the pseudo random number generator.

### 3.4.1 Pseudo random number generator

This algorithm was initially used for its ability to select numbers without any regularity. However, the true random number is the result that must be obtained by way of a special random test. These random numbers are often used in different techniques employed in statistics. For example, when using Monte Carlo simulation to calculate, when looking to extract representative individuals from a statistically good population, it is necessary to have the help of random numbers. It is important and indeed interesting to note that the most distinctive features of random numbers are [48]:

a) The distribution of the generated values within the defined interval or set (the probability of occurrence of each value is the same)
b) The values generated in the future are not predicted based on past or current values.

Each generation of numbers is entirely random and unique. In order to obtain a sequence of random numbers, many methods can be used; these are called random number generators. However, to obtain real random numbers, i.e. where all results are unrelated, need to be derived from physical phenomena. A common example of this in life is rolling a dice and a turntable game. Based on research conducted by statisticians, scientists and mathematicians, the generation of high-frequency noise samples through circuits is one of the best methods to use to obtain random numbers of random numbers [48]. These random number generators are called physical random number generators, but they have higher technical requirements or physical instrument prices. For instance, a ComScire random number generator costs around 900 US dollars. Some projects do not produce high returns, and therefore simply cost large amounts.

In fact, in some practical applications, the generation of pseudo-random numbers is often sufficient for research. A pseudo-random number is a random

number generated by way of a software method. The resulting values seem to be uncorrelated and to have been randomly selected, when in fact they can be calculated because they have statistical characteristics similar to those of random numbers. The resulting data is calculated using a fixed, reusable calculation method. The number of values obtained after the random process of software generation is very large, but all of the values are generated using mathematical recursion formulas. These numbers can appear in a certain probability distribution and pass the corresponding randomness test. Correspondingly, the method of generating a pseudo-random number is called a pseudo-random number generator.

A pseudo-random number generator (PRNG) is also known as a deterministic random bit generator (DRBG) [49]. It is an algorithm that employs mathematical formulae to derive and generate a sequence of random numbers. [50]. The sequence of values obtained by means of PRNG is not truly random (although it may contain a sequence of real random numbers), and the generation of those pseudo-random numbers is entirely dependent on the initial values in the input algorithm, which is itself the seed of the PRNG. However, the resulting pseudo-random number sequence does not have the values generated by those hardware random number generators which are closer to the true random number. Despite this, whether considering its speed of generation or its reproducibility, the existence of pseudo-random number generators in different areas of practice is still greatly significant. [51] If the mathematical equations of uniformly distributed pseudo-random numbers are changed, then pseudo-random numbers which satisfy different distributions can be obtained. The algorithm used to generate a uniformly distributed pseudo-random number can be considered as the basis of the pseudo-random number generation algorithm. Common uniform distributed pseudo-random number algorithms include the linear congruence algorithm and Mersenne twister algorithm.

The mathematical knowledge used in the algorithm may be complicated, but it can be split roughly into two steps [52]. The first is to provide arbitrary seeds for the operation of the PRNG algorithm and the second is to propose the generation of the next random number claim. In addition, to generate a pseudo-random number, it is necessary to select a seed and then calculate based on a particular generation algorithm and then obtain another pseudo-random number based on the seed; the result tends to be a number in a decimal system. The new pseudo-random number should then be used as the seed, or the initial value, and the next pseudo-random number should be recursively obtained according to the algorithm until all of the pseudo-random numbers that do not appear repeatedly have been calculated. The set of values obtained each time in this process is the sequence of pseudo-random numbers. These calculations can be calculated simultaneously, or one at a time. Regardless of the process implemented by the pseudo-random number generator, once the seed and its mathematical equations and generation algorithms have been determined, the results of the final sequence are unique. In fact, the random number of pseudo-random numbers is that although the value obtained after the end of one operation is fixed, the pattern in which the value appears, and the sequence of the last generated pseudo-random number can pass most of the randomness tests. This demonstrates that the performance of pseudo-random numbers is similar to that of random numbers.

The PRNG algorithm has three important characteristics [52]. The first is high efficiency. PRNG is computationally efficient and can generate a large number of numbers in a short period of time. When an application has a demand for the number of digits, the PRNG application proves extremely useful. The second is certainty. If the seed at the time of execution of the PRNG has been determined and so too has the modulo operation, then the resulting pseudo-random number column is also determined. The third and final characteristic is periodicity. Although the PRNG has periodicity, the sequence of numbers will

eventually stop. However, its cycle is long enough to handle most practical applications in life.

PRNG has many applications, most of which are in cryptography, and also in some simulation methods such as the Monte Carlo method. In this project, it will be used to randomly generate the centre coordinates of the sphere, constructing a three-dimensional model of the avocado peel and the decayed part. In MATLAB, the rand() function, randi() function and randn() function is able to generate pseudo-random numbers. These functions are all made with the help of a single shared random number generator and can be called by other functions to form a more complex random number generation algorithm.

However, PRNG cannot be applied to experiments and fields at will because it is able to pass some randomness tests. The most basic and indeed essential requirement of PRNG is its excellent statistical properties. That is to say, after conducting rigorous data analysis and testing, it is certain that the output after applying it will be similar to the result of the natural random number. John von Neumann once stated that "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin. [53]". Deterministic generators of this kind can be considered to still have various drawbacks. Many generated values may still suffer from uneven distribution, which conflicts with the purpose of initially generating random numbers. It is true that PRNG can produce the similar effect as a random number generator, but it cannot be used if the flaw is obvious.

In this project, the optional interval of the pseudo-random number is small, and MATLAB comes with a pseudo-random number generation function which is convenient to call and is therefore used directly. The following two sections will address the particular application scenarios of PRNG.

### 3.4.2 Anthracnose

As Figures 2.14 illustrate, when an avocado is suffering from Anthracnose, the rotted area is flaky, the contour edge is smooth and there is no angularity, which means that it resembles a cloud-like shape. At the same time, the color of the decayed part is deepened. Mathematical modelling of the decayed part of the avocado in real life can be performed, and the resulting geometry will appear as indicated in Figure 3.16 below.

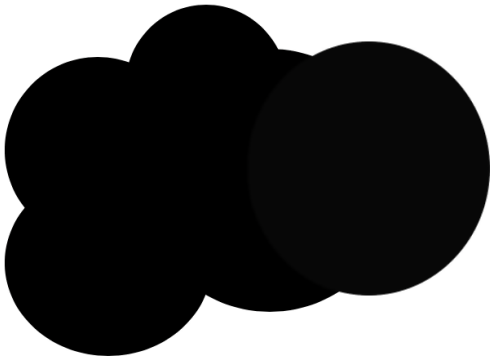


Figure 3.16 Rotten shape of Anthracnose

The random generation algorithm runs throughout the process of implementing this cloud object. First, select a point coordinate $o_1$, in the flesh portion of healthy avocado as the seed of PRNG, take point $o_1$ as the center of the sphere, select $r_1$ as the radius of the sphere and use the equation introduced in 3.1.1 to draw the ball. Using the pseudo-random number generation function in MATLAB, the point $o_1$ is used as the seed and the next random number is selected from the three directions of the $x$-axis, $y$-axis and $z$-axis and combined to form a new point coordinate $o_2$. At this time, select $r_2$ as the sphere radius of the new sphere, and draw the ball with $o_2$ as the sphere. Next, take $o_2$ as the seed of PRNG, select the next pseudo-random coordinate point $o_3$ in the three-dimensional space as the center of the new ball and $r_3$ as the new sphere radius, and then draw the third sphere. Loop through this process until a satisfactory shape has been drawn. It is important to note that the

selection process of the sphere radius of each random ball also uses the pseudo-random number generation algorithm. Employing the random function in MATLAB, and when the seed radius is used, the value of the sphere radius is selected from a fixed range. When the length of a seed radius $r_1$ is established, the radius $r_2$ of the next sphere is selected “at random” by the computer. When the second sphere is drawn, "random" selects the next radius $r_3$ , and this process then continues in a loop until no new spheres need to be generated. Figure 3.17 shows the internal mathematical model when generating a cloud object.

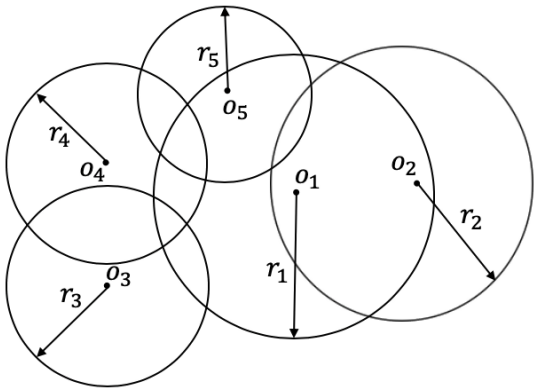



Figure 3.17 Cloud-shaped decay mathematical model

It is important to acknowledge that controlling the number of random balls generated can also control the degree of avocado rot. Real-life images and research surveys indicate that the more severe the Anthracnose, the larger the area of decay. That is to say, during the three-dimensional modelling process, the number of randomly generated balls is controlled. The higher the number, the more spheres are generated, and the larger the volume of the superimposed spheres, the larger the decay area displayed. As a result, in the process of achieving different degrees of decay, the approximate range of the number of spheres corresponding to different degrees can be established by continuously adjusting the parameters.

### 3.4.3 Stem-end rot

As can be seen from Figures 2.10, when an avocado is suffering from Stem-end rot, the main performance is that it starts to rot from the head, and the decayed part is strip-shaped. For this part of the implementation, two algorithms are designed. In theory, if the first algorithm can be implemented, then the 3D model produced will be more similar to the avocado fruit in real life. However, due to the limited amount of time for this project and the weakness of my programming ability, the first algorithm was not implemented. After this, another second algorithm was designed. Although its expected effect is not as good as that of the first algorithm, it can essentially achieve a strip-like decay shape. Regardless of the first or second algorithm, a random generation algorithm is still used. Next, both algorithms will be described in detail.

1) First algorithm (not implemented)

   Following careful observation of Figures 2.10, Figure 2.11 and Figure 2.12, the strip-like decay portion of avocado can be analogized to the shape shown in Figure 3.18.

Figure 3.18 The strip-like decay of Stem-end rot

   The generation of this pattern is similar to but not identical to the generation of the Anthracnose decay. Given that the outer contour of the cloud-like decay is not certain, in this disease, the general shape must be strip-shaped, but it also needs to be twisted. For this feature, the following method is designed: According to the thickness of the strip-like decay, the

value of the parameter $r$ , the radius of all the balls used to generate the decay, was carefully chosen. The center of the first ball $o_1$ was determined to generate the decayed area. The PRNG is also used to select the pseudo-random number as the spherical center point $o_2$ of the second ball in the three coordinate system directions respectively. However, this random number selection adds conditions. The coordinates of the second point can only be selected within the range of $45^o$, as shown in Figure 3.19 (a).

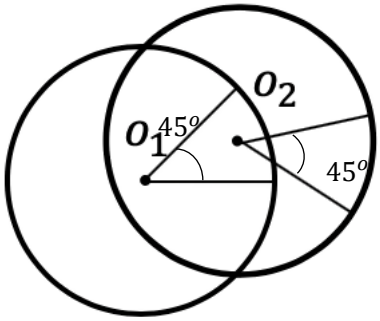



Figure 3.19(a) The process of the first algorithm for Stem-end rot

At this stage, $o_2$ is used as a random seed, and the next random number point $o_3$ is selected within the range of $45^o$ angle, while $r$ is employed as the sphere radius to draw the ball, as shown in Figure 3.19 (b).

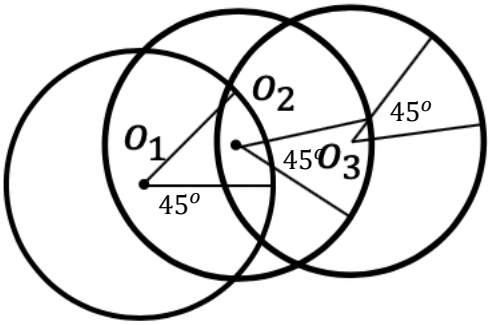



Figure 3.19(b) The process of the first algorithm for Stem-end rot

A value must be established for the number of generated pseudo-random numbers. Providing that the number of random numbers to be obtained

does not arrive, the action of randomly generating the center point of the ball and drawing the sphere will be cycled until the initially set condition is satisfied. The magnitude of this value will then be adjusted, thus enabling the length of the decay strip to be controlled.

Moreover, since the selection range of the center point is narrowed down to the angle of $45^o$, the center of the whole strip does not fluctuate significantly. Every time the next ball is generated, it is controlled within a small range. However, given that the model is three-dimensional, the angle cannot be well controlled in space, and therefore the cumbersome random generation method is not implemented.

2) Second algorithm (implemented)

The second algorithm relevant for this project is based on Anthracnose's algorithm. This is the algorithm used to complete the project. Essentially, this made it possible to recreate the intricacies of the rotten section of avocado. The strip-shaped decayed area is not treated in the computer program as a curved and irregular line, but as a smooth curve. Those curves start from the head of the avocado, similar to the curvature of the avocado peel, and the geometry is shown in Figure 3.20.

Figure 3.20 The geometry of rotten part for Stem-end rot

This image is, of course, just a two-dimensional cross-section. Within the actual fruit, however, it is impossible to have only a one-sided rotted area.

If it is assumed that the various sections of the rotten avocado exist as shown, then the line forms a surface. Because the curves are not closed, and are in a cross-section, the left and right halves of the avocado have curves. When a mathematical model is created for these strips of decay in three dimensions, it is possible to include both the left and right sides. In this way, the two decaying areas are linked to form a hollow semi-ellipse, as shown in Figure 3.21.

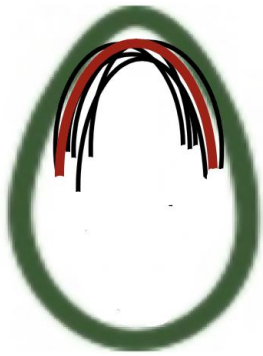


Figure 3.21 The linked decaying areas of Stem-end rot

This, in the three-dimensional space, is equivalent to forming a hollow semi-ellipsoid. Assuming that the strip-shaped decay regions on the left and right sides can be joined into a hollow semi-ellipsoid with different axial lengths, the decayed area of the fruit can be realized in the form of a hollow semi-ellipsoid.

The same method is used here as in the previous disease type. To carry it out, a range of values for the center of the ellipsoid must be set. Through the pseudo-random number generation algorithm, the spherical center of the semi-ellipsoid with "random" was then selected, and three values were almost randomly selected (these will be referred to as a, b and c) from the value interval of the axial length. Using the length of the shaft, a semi-ellipsoid was drawn. They were painted grey-black to represent the area of decay. The next step varies depending on the diameter of the decay. The

appropriate parameter w was chosen, but the center point of the ellipsoid could have been used. The next step was to draw a semi-ellipsoid with $a - w$, $b - w$, and $c - w$ as the axis length, and fill it with light grey. This is represented visually as normal flesh. The specific process is shown in Figure 3.22 below (for the sake of clarity, only the two-dimensional picture is used as an example to explain the coloring process, and the principle in three-dimensional space is the same as it is).

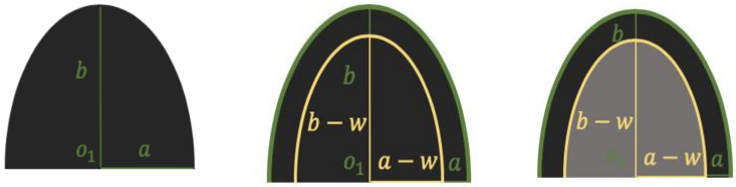



Figure 3.22 The coloring process for the Stem-end rot decay area

The resulting decay graph is as follows. (Figure 3.23)

Figure 3.23 A typical representative of the Stem-end rot decayed area

The semi-ellipsoid should be drawn in the same way as mentioned in 3.1.2 Assuming that the z-axis represents the ellipsoid's height, a condition should be added to the selection of the z-value ($z>z_{center}$). The object obtained after MATLAB executes the statement is a semi-ellipsoid. It can then be seen that as long as the parameter size of w is changed, the thickness of the rot bar can be selected. After that, the new ellipsoid centre coordinates are continuously generated by the randi() function, and a hollow ellipsoid will be continuously generated. A three-dimensional model of the decayed area of the avocado with Stem-end rot can be recreated in this way.

# 4.Implementation

Once the algorithm has been designed for the three-dimensional model of an avocado, these algorithms need to be implemented using MATLAB software. The code for this program was written in the MATLAB R2018a environment using the MATLAB language, and the previous theoretical ideas were visualized by way of graphical objects based on the algorithms of each part mentioned in the previous chapter. The following sections provide a detailed explanation of the language used in programming, the establishment of the three-dimensional model of avocados and the specific implementation steps of the final human-computer interaction interface.

## 4.1 Language

MATLAB, also known as Matrix Laboratory, is a multi-paradigm numerical computing software developed by MathWorks Inc. in the United States. It is also a high-level programming language that is employed in computer science. [54] This language was originally written with a view to accessing the matrix software, which was developed by the LINPACK and EISPACK projects. [55] The environment is easy to use and comprises environments primarily used for numerical calculations, visualization and programming. [54] When there are problems and scenarios to be solved, these are represented by familiar mathematical symbols. [55] Meanwhile, MATLAB is also an interactive system “whose basic data element is an array that does not require dimensioning [55]”. This feature makes it possible for MATLAB to provide an effective solution for problems that require the use of vector and matrix formulas. Based on its nature, MATLAB’s primary and most basic function is to perform various operations on the matrix, create usable interfaces, draw functions and seek to express data. However, it can also be employed to develop algorithms, generate new models or applications and analyze and process the data given. [54] The MATLAB language does not exist in isolation. Other high-level programming languages,

such as Fortran and Java, have interfaces that are supported by MATLAB language. Given that it also provides a wide range of inbuilt mathematical functions, it can also be applied to 2D and 3D data visualization, image processing and so on. MATLAB has a wide range of applications. It offers extremely efficient support in mathematics, statistics, simulation, electronics, industry, bioinformatics, finance and testing.

The MATLAB system is comprised of five elements: MATLAB language, MATLAB working environment, image processing, MATLAB math function library and MATLAB application program interface (API). When programming in MATLAB, users can quickly create simple applets and complete, cumbersome applications that are made up of many functions and algorithms. The working environment refers to tools that manage a large amount of data in the workspace, for example tools employed to generate, analyze and manage .m files. MATLAB is a system that can be used to process a variety of graphics. The mathematical function library is comprised of a series of mathematical calculation rules ranging from addition to subtraction and multiplication and division to complex algorithms such as fast Fourier transform. The MATLAB API is a library that forms the interface between MATLAB language and other high-level programming languages. [55]

Some of the advantages of MATLAB are as follows [54]:

(1) It works independently of the compiler;

(2) It has efficient numerical calculation functions;

(3) It offers complete calculation results and programming visualization functions;

(4) It provides an inbuilt rich content library, such as power system, electromagnetics, hydraulics;

(5) It offers a feature-rich application toolbox and HELP system to provide a more feature-rich feature set;

(6) The creation of function handles makes it easier to call each other between functions, improve the reliability of function calls, reduce redundancy in program design and improve the efficiency of repeated execution;

(7) It is possible to use the Add and Remove boxes to utilize the GUI interface, which makes the operation more concise and convenient.

## 4.2 Realization of a three-dimensional model of the avocado

The entire project was completed through the creation of three files. These were a gui.m file, gui.fig file and a factor.m. As the final requirement of this project is to display a profile of an avocado from three directions on the gui interface, the profile is composed of a number of pixel matrices, and different parts of the avocado are represented by different grey values. To achieve this, a three-dimensional model of 300*300*200 pixels was created, and its initial grey level set to 0. This is pure black and provided a background for the establishment of the avocado model. Next, within the cuboid made, the avocado's nodules were painted one by one. The grey values of all the pixels in the rectangular body were in the range of 0 to 1. The levels of grey were as follows:

0.3 or 0.4: grey-black, representing the area of decay (the rot)

0.5: dark grey, representing the peel area

0.8: light grey, representing the flesh area

1: white, representing the core area

For the generation of the two different diseases, two callback functions are respectively corresponding in the gui.m file.

```
function pushbutton1_Callback(hObject, eventdata, handles)
```


Figure 4.1 Two functions for two diseases

“A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.[56]” In simple terms, this means that whenever the program runs the application program calls on pre-prepared functions in the library through the API. Some library functions, however, require the application to pass through a function first. In this case, it calls the correct function when appropriate in order to complete the intended task. The function that is passed on and later called upon is referred to as a callback function. In order to implement different clicks on the gui interface and then generate targets of different rotten types, the buttons on the two gui interfaces must correspond to two callback parameters. The specific operation of the gui will be explained in Section 4.3 of this paper. These two callback functions correspond to the implementation of my two rotting algorithms. In both callback functions, a healthy avocado model is first generated. The specific algorithm used to do this is described in full detail in 3.4.

### 4.2.1 Realisation of healthy avocado

In order to create a healthy avocado, it is necessary to initialise a 300*300*200 3D box named graphics. Then all the parameters used in the project, the spherical coordinates (circle_x, circle_y, circle_z) and its radius (r_circle) of the sphere required to draw the avocado fruit, are input. The centre coordinates of the ellipsoid (ellipse_x, ellipse_y, ellipse_z) are used. Because the ellipsoid has different axial lengths in different varieties, the following statement was used to initialize the parameters.

```
a=r_circle + handles.type;
b=r_circle;
c=r_circle;
```

Figure 4.2 Initializing the parameters

As shown in the code of the image below, the selection of different buttons corresponds to a variety of parameter values. Thus, ellipsoids of different axial lengths were generated.

```
switch str{val};
case '1 ZUTANO'
   handles.type = 30;
case '2 HASS'
   handles.type = 24;
case '3 BACON'
   handles.type = 18;
case '4 GWEN'
   handles.type = 12;
case '5 REED'
   handles.type = 6;
end
```

Figure 4.3 The code for different cultivars

The initial parameters for my model also included the square thickness of the peel, the roughness of the peel and the grey value to be used for later colouring. As previously mentioned, these were all key.

```
value1=0.5;%Dark gray, peel
value2=0.8;%Light gray, flesh
value3=1;%White, score
```

Figure 4.4 The grey value for different parts of an avocado

By looping through all the pixels in the entire cuboid, the square distance from the centre of the fruit of each point of each point from the centre of the fruit can

be judged. Through doing this, it was possible to judge whether the point belongs to the upper part or the lower part of the avocado.

```
distance2circle=(ii-circle_x)^2+(jj-circle_y)^2+(kk-circle_z)^2;

distance2ellipse=(ii-ellipse_x)^2/(a^2)+(jj-ellipse_y)^2/(b^2)+(kk-ellipse_z)^2/(c^2);
```

Figure 4.5 The code for determining the location of the target point

If the point (ii, jj, kk) is within the peel and below the avocado's halfway point, the hemisphere was drawn. At the same time, a region that produced the surface of the peel and the radius of the sphere was selected. To do this, a random function was used to select the value of the radius of the ball. Logically, the radius was not changed every time it was changed. The probability of the radius of the ball changing is 1/3 each time. When a new number is selected as the radius, the original value should be replaced by the new value.

```
if abs(distance2circle-r_circle^2)<peel && distance2circle>r_circle^2 && ii<height/2
    && (ii - ant_x)^2 + (jj - ant_y)^2 + (kk - ant_z)^2 > (ant_r_circle + 10)^2

    r_circle_mini=peel;


    if mod(randi([1,10]),3)==0
        r_circle_mini=rand*r_circle/ci;
```

Figure 4.6 The code for determining the parameters of peel

When the scan point (iii, jjj, kkk) was looped near the 20*20*20 cube area, if the point turned out to be in the drawing area of the peel ball, the ball was drawn and painted dark grey.

```
      for iii=-10:10
          for jjj=-10:10
              for kkk=-10:10
                  distance2circle_t=iii^2+jjj^2+kkk^2;
                  if distance2circle_t<r_circle_mini
                      graphics(ii+iii,jj+jjj,kk+kkk)=value1;
                  end
              end
          end
      end
  end
```

Figure 4.7 The code for coloring

The generation of the upper half of the avocado model was carried out using similar code. The code used for this upper half produces a semi-ellipsoid. The steps used to produce the peel were just like for the lower half. After the success of the upper and lower halves of the model, the score and flesh within the fruits were stained separately. The code shown in the above image is an important part of what was generated by the healthy avocado models. The next section will introduce the code to implement the decay of avocado.

### 4.2.2 Realisation of diseased avocado

The first disease Anthracnose implementation called upon the factor function. This function was stored within the factor.m file.

```
graphics = factor(ant_x, ant_y, ant_z, ant_r_circle, circle_x, circle_y, circle_z, r_circle, graphics, height, width, deepth, handles.count, handles.size);
```

Figure 4.8 The function for Anthracnose

In the factor.m file, there is a recursive function. Nx, ny and nz were initialised, and will be treated in this paper as coordinates of e balls. First, the coordinates of the decaying ball (o_x, o_y, o_z) were initialised and the sphere with the floor_o_r as the radius was drawn. Next, the randi() function was used to randomly generate new ball coordinates in this sphere. It was then necessary to determine whether the generated small sphere was in the decayed area or, in other words, to establish whether it was inside the fruit. If it was inside that area, the pixel value was set to 0.3: greyish black.

The code generated by the second disease is stored in the gui.m file, and its decay radius and decay centre were set to their initial values at the start of the work. The parameters related to the selection of decay centres have changed compared to the initial values of the previous disease.

```
ant_r_circle=10; %floor(r_circle / 20);

ant_peel_r_circle=ant_r_circle - peel;

% x
%ant_x=-randi(r_circle) + circle_x;
% y
%ant_y=2 * randi(r_circle) - r_circle + circle_y;
% z
%ant_z=sqrt(r_circle^2 - (ant_x - circle_x)^2 - (ant_y - circle_y)^2) * sign(randi(1)-0.5) + circle_z ;

ant_x = circle_x + 50;
ant_y = circle_y;
ant_z = circle_z;
```

Figure 4.9 The code for initializing parameters of rotten part of an avocado

As seen here, the number of decaying iterations and the number of decaying spheres are passed to the function that generates the decay. This is achieved using the handle() function. The specific code for generating the decayed area is similar to the method used to model the previous disease type. After the three-dimensional model of avocado is established, the cross-sectional view in three directions can be drawn through use of the following code.

```
for ii=1:width
    if ii < deepth
        axes(handles.axes1);
        image=graphics(:,:,ii);
        imshow(image)
    end
    axes(handles.axes2);
    image=graphics(ii,:,:);
    image=reshape(image,[width,deepth]);
    imshow(image)

    axes(handles.axes3);
    image=graphics(:,ii,:);
    image=reshape(image,[height,deepth]);
    imshow(image)
end
```

Figure 4.10 The code for drawing cross-sectional view pictures

## 4.3 Realization of GUI

Entering "guide" in the command line window of MATLAB brings up a fresh window. In this window, a new GUI can be created or an existing one opened. When New GUI is chosen, two files are created. These are the .m and .fig files. It should be noted that the .fig file is the design of the initial interface, and that can be completed by mouse dragging and other simple operations. The .m file, on the other hand, contains the callback code corresponding to the control in

the GUI. This is why the two callback functions created in Section 4.2 are placed in the gui.m file. The GUI interface of this project looks like the image below.

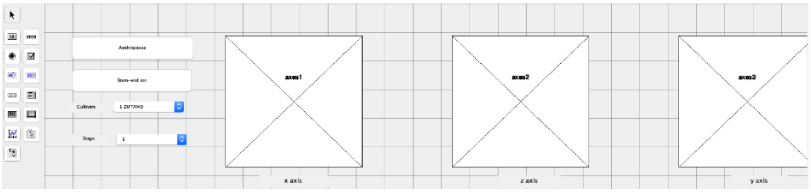


Figure 4.11 Gui's initial interface

In this program, the names of the diseases are set to two buttons and the types of avocado and degrees of decay appear in corresponding pop-up menus. The display area of the profile is set to three axis regions.

All controls added on the GUI interface can be double-clicked. After double-clicking on any control, its property inspector will pop up; this is shown here.

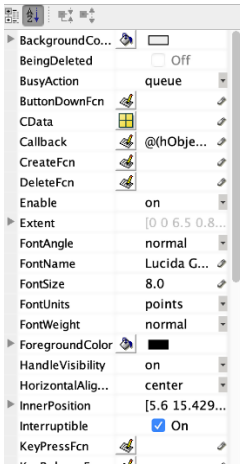



Figure 4.12 Interface that can change control properties

All initial properties of the control can be changed on this interface. This includes the text displayed on buttons, optional categories in the pop-up menu, and so on. Additionally, the tag values of this control can be viewed. This is very important to call the function on the handle in the callback program.

When the interface is run, the first program executed is the initial program. This part of the code should be added to the test_OpeningFcn(hObject, eventdata,

handles, varargin) function, which contains some initial value settings for the control.

```
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui (see VARARGIN)

handles.size = 10;
handles.count = 10;
handles.cnt = 40;
handles.len = 25;
handles.type = 30;
% Choose default command line output for gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

Figure 4.13 The code for setting the initial value of the gui

As for the button program for the two diseases, the following code corresponds to the button Anthracnose.

```
function pushbutton1_ButtonDownFcn(hObject, eventdata, handles)
```

Figure 4.14 The button program for Anthracnose

The code corresponding to the button Stem-end rot in the program is as follows.

```
function pushbutton3_Callback(hObject, eventdata, handles)
```

Figure 4.15 The button program for Stem-end rot

Pressing different buttons of course triggers different callback functions. These correspond to running different function codes, and therefore generating different types of decay within the avocado model.

Programming for two popup menus is carried out as follows. First, the pop-up menu in the GUI editing interface must be selected. In the "String" column of the property editor, the title of the option desired has to be selected, and OK must be clicked, as shown in the following figure.

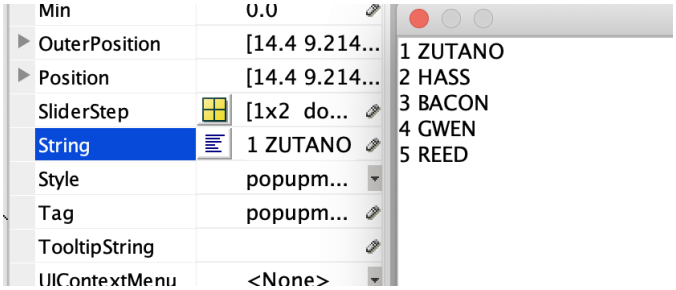



Figure 4.16 Set the display content of two pop-up menus

The corresponding codes are below.

```
function popupmenu1_Callback(hObject, eventdata, handles)

str = get(hObject, 'String');
val = get(hObject,'Value');
% Set current data to the selected data set.
switch str{val};
case '1 ZUTANO'
   handles.type = 30;
case '2 HASS'
   handles.type = 24;
case '3 BACON'
   handles.type = 18;
case '4 GWEN'
   handles.type = 12;
case '5 REED'
   handles.type = 6;
end
guidata(hObject,handles)
```

```
function popupmenu2_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
% Set current data to the selected data set.
switch str{val};
case '5'
   handles.size = 10;
   handles.count = 10;
   handles.cnt = 40;
   handles.len = 25;
case '4'
   handles.size = 8;
   handles.count = 8;
   handles.cnt = 32;
   handles.len = 16;
case '3'
   handles.size = 6;
   handles.count = 6;
   handles.cnt = 24;
   handles.len = 12;
case '2'
   handles.size = 4;
   handles.count = 4;
   handles.cnt = 16;
   handles.len = 8;
case '1'
   handles.size = 1;
   handles.count = 2;
   handles.cnt = 8;
   handles.len = 4;


end
guidata(hObject,handles)
```

Figure 4.17 The code for connecting gui and function

The code to display the pixel points in the axis area is as follows

```
for ii=1:width
    if ii < deepth
        axes(handles.axes1);
        image=graphics(:,:,ii);
        imshow(image)
    end
    axes(handles.axes2);
    image=graphics(ii,:,:);
    image=reshape(image,[width,deepth]);
    imshow(image)

    axes(handles.axes3);
    image=graphics(:,ii,:);
    image=reshape(image,[height,deepth]);
    imshow(image)
end
```

Figure 4.18 The code for displaying the pixel points

# 5.Evaluation

To complete a project in its entirety, evaluation is an essential step. This step determines whether the intended task has been carried out to a satisfactory level. Whether it has or not, the degree of completion can be assessed, and the value and significance of the results obtained through this task can be examined. [57] In addition to the ability to analyse and judge the information already available, the assessment stage has far-reaching significance. It can help people reflect on their previous work and can be used as a basis for predicting future changes or issues. [58] The program was the main part of this project. For this reason, it is important to evaluate the project mainly through the testing of this program. This will determine whether the requirements of the project have all been realised, and the efficiency (or otherwise) of the program. Functional and performance tests will be conducted on the program separately.

## 5.1 Functional testing

Functional testing, also called behavioural testing, is any software test that uses black box testing as its basis. When a functional test is carried out on a program it is effectively a form of quality assurance (or QA) [59].

The purpose of this sort of test is to ensure that the system or component's final function is consistent with the requirements that were set out at the beginning of the project [60]. When testing the project, it is important to look for any vulnerabilities within the program. Simply put, the purposes of functional testing are:

1) Checking the realization of the demand
2) Missing search function implementation
3) Checking if the application has any vulnerabilities

The requirement of this project was to design a human-computer interaction interface. In the chosen interface, users interact with the computer by selecting the variety of avocado they want to view on the interface and model their image after the onset of two different diseases. They can be viewed from three directions. The area of decay is randomly generated, and the degree of decay can be selected in order to test whether the functions are implemented when the interface is used. The following tests helped to assess the program.

### 5.1.1 Human-computer interaction interface generation

After running the gui.m file, a gui window will appear.

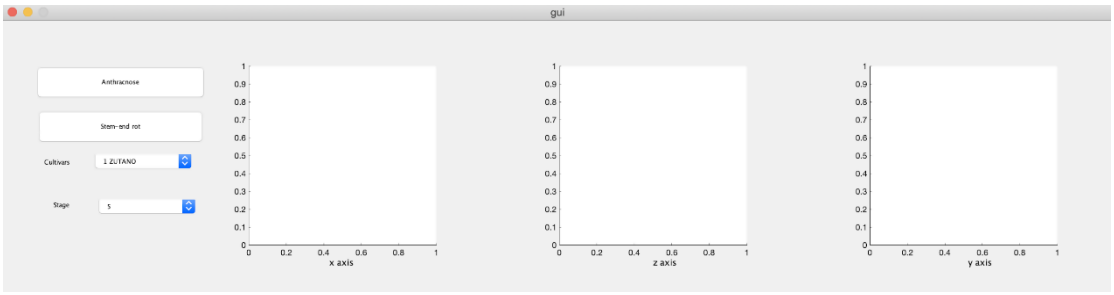



Figure 5.1 The gui window

There are two buttons on the window, Anthracnose and Stem-end rot. These are the two diseases that can be selected for the avocados. There are two drop-down menus for Cultivars and Stage, and these can be selected in 5 varieties and at 5 levels. There are also three display windows, which show the cross-section of the 3D avocado model from the x-axis, y-axis and z-axis directions.

### 5.1.2 Avocado cultivars

Regardless of which disease is chosen for a given avocado, the code and algorithm for the initial healthy model are the same. This means that, no matter which disease is chosen, the three-dimensional model of an avocado of a given variety is always the same. The pictures that follow are the physical pictures of each variety, and the pictures generated after the program is run.

- Zutano

Figure 5.2 Physical map of Zutano avocado

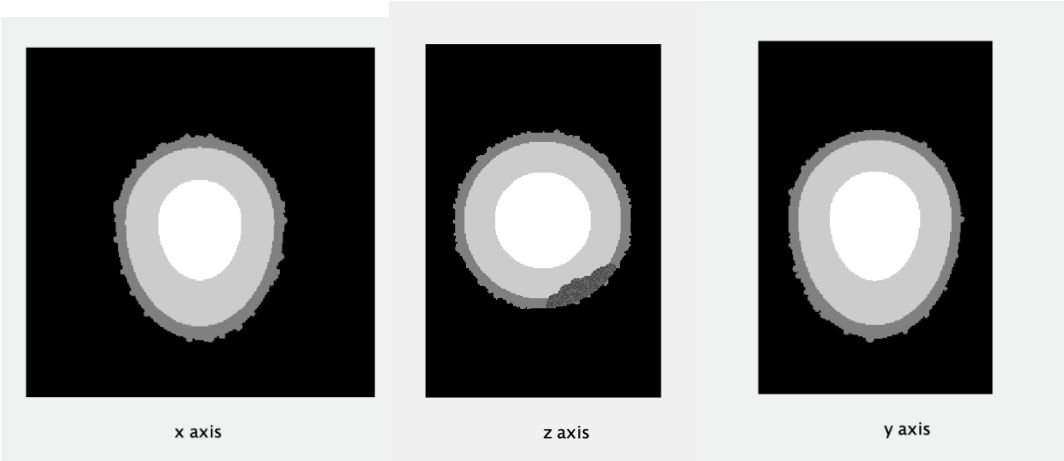



Figure 5.3 The result of Zutano avocado

- Hass

Figure 5.4 Physical map of Hass avocado

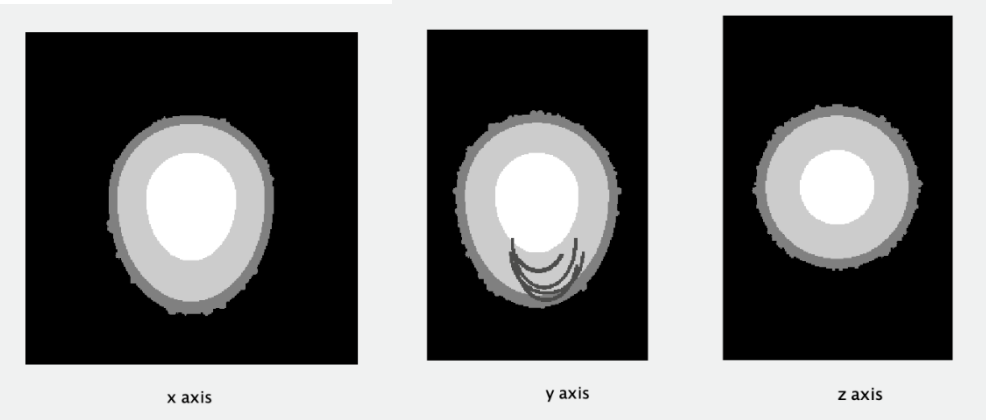



Figure 5.5 The result of Hass avocado

- Bacon

Figure 5.6 Physical map of Bacom avocado

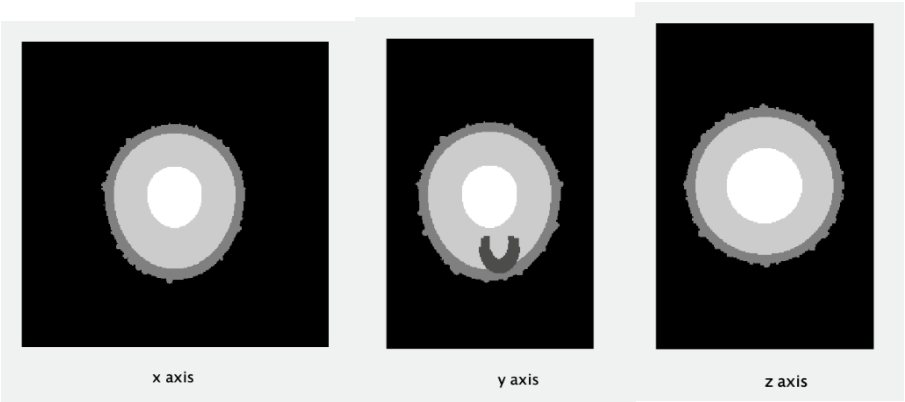



Figure 5.7 The result of Bacon avocado

- Gwen

Figure 5.8 Physical map of Gwen avocado

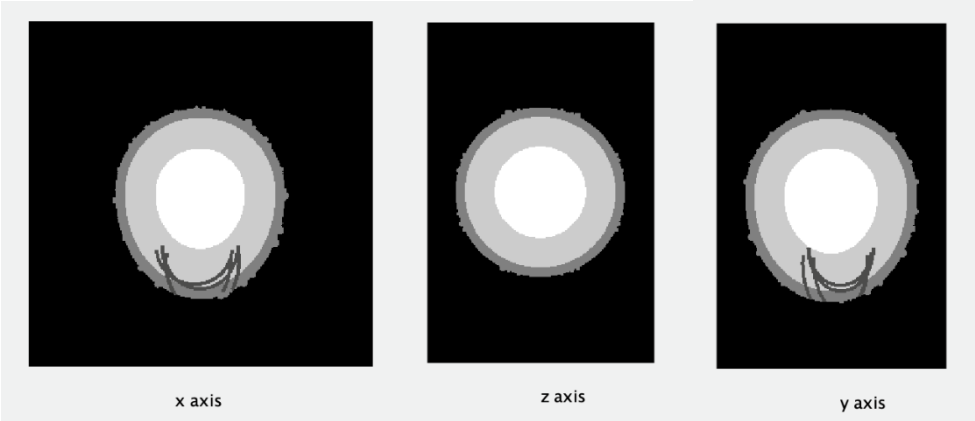



Figure 5.9 The result of Gwen avocado

- Reed

Figure 5.10 Physical map of Reed avocado

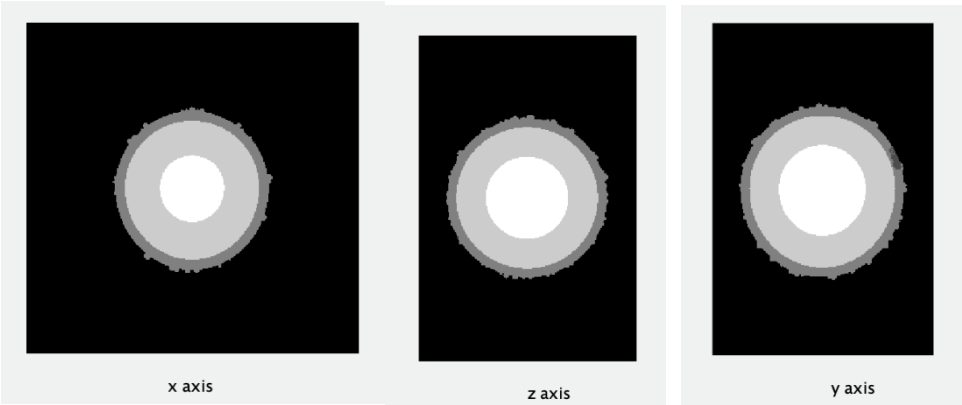



Figure 5.11 The result of Reed avocado

In comparison, it can be clearly seen that the physical maps of the Bacon variety are not similar to the cross-sections of the generated models, yet the profiles of the other four varieties are basically the same as the physical maps. This implies that, in addition to the Bacon variety, the other four varieties, Zutano, Hass, Gwen and Reed, have been successfully established in the form of 3-D models. As for Bacon, if the radius of the sphere portion is reduced and the value of the axial length of the ellipsoid portion increases in the height direction, then its three-dimensional model may be closer to the real object avocado.

### 5.1.3 Decayed avocado form

Unfortunately, not every photo of avocados of each species can be found. The data in this area of the database is neither perfect, nor complete. This is why the project needs to be established. Tests on decay patterns can only be compared to pictures of sick avocados that can be found, in order to give an accurate idea of what the decay should look like.

- Anthracnose

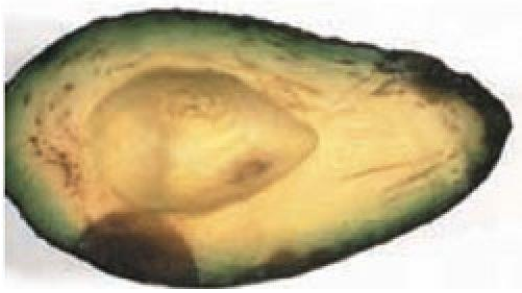


Figure 5.12 An avocado suffering from Anthracnose

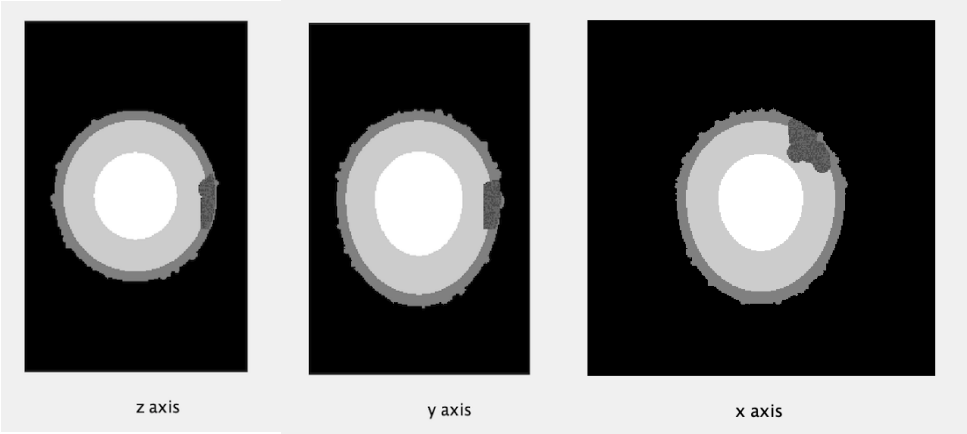



Figure 5.13 The results for an avocado suffering from Anthracnose

- Stem-end rot

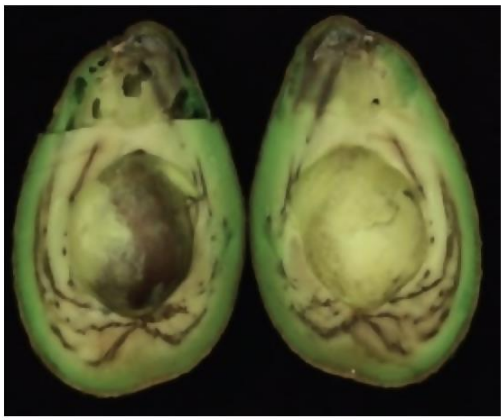


Figure 5.14 An avocado suffering from Stem-end rot

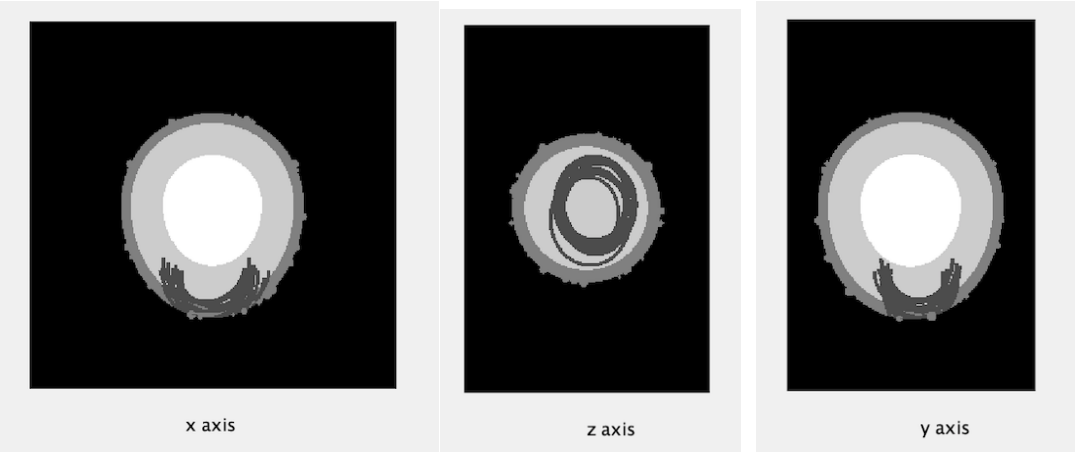



Figure 5.15 The results for an avocado suffering from Stem-end rot

The comparison of the pictures above reveals that the morphology of the decayed area of Anthracnose disease is similar to the physical picture. Although the degree of the decay area of Stem-end rot disease is not particularly high, the target decayed presents in a strip shape. The algorithm design for this disease needs to be improved, and so does the database.

### 5.1.4 Degree of rot

Because the variety of avocado does not have influence on the area or the degree of decay, only the profile of Bacon avocado was selected for analysis and comparison. The test was done a lot, since the decaying area was randomly generated, and the decay obtained each time was different. The following pictures are typical shapes, which show the results of different degrees of decay after the Bacon suffered from two diseases.

- Bacon suffering from Anthracnose

Stage 1

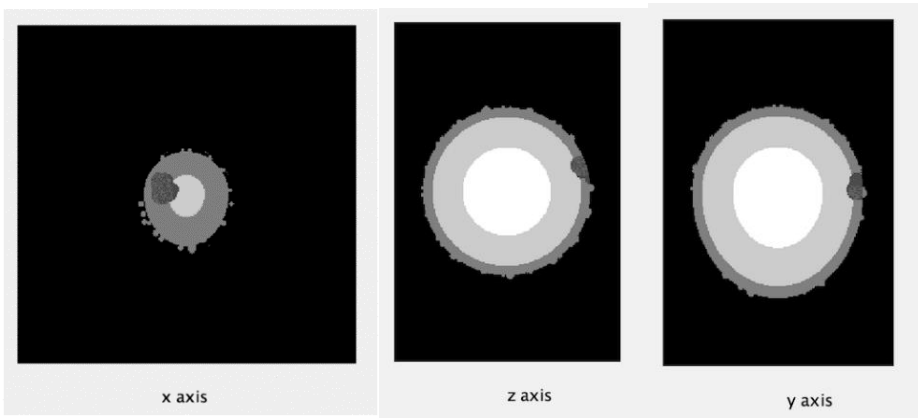



Figure 5.16 Bacon suffering from Anthracnose at stage 1

Stage 2

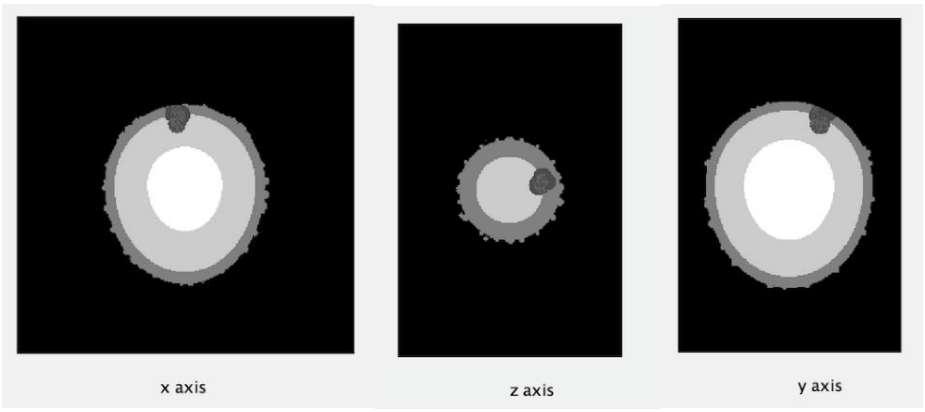



Figure 5.17 Bacon suffering from Anthracnose at stage 2

Stage 3

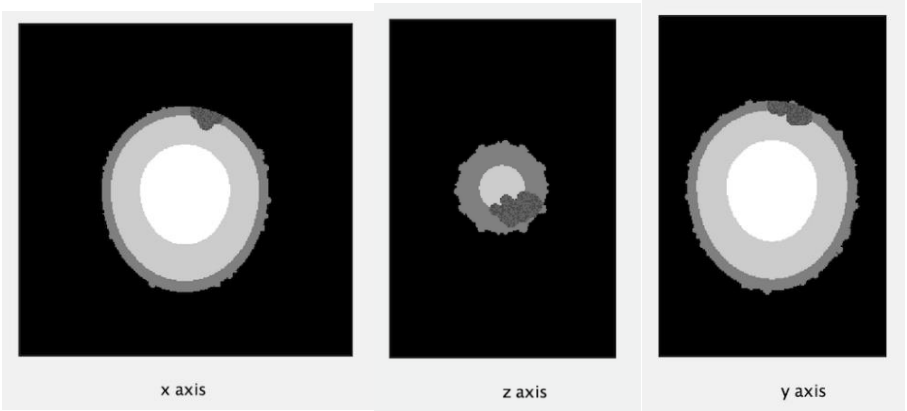



Figure 5.18 Bacon suffering from Anthracnose at stage 3

Stage 4

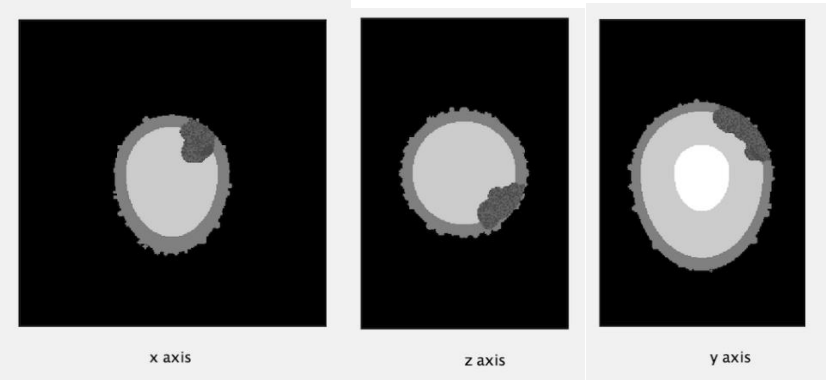



Figure 5.19 Bacon suffering from Anthracnose at stage 4

Stage 5

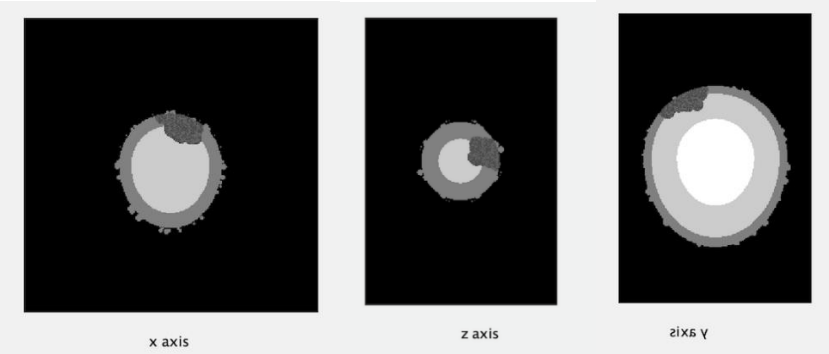



Figure 5.20 Bacon suffering from Anthracnose at stage 5

- Bacon suffering from Stem-end rot

Stage 1

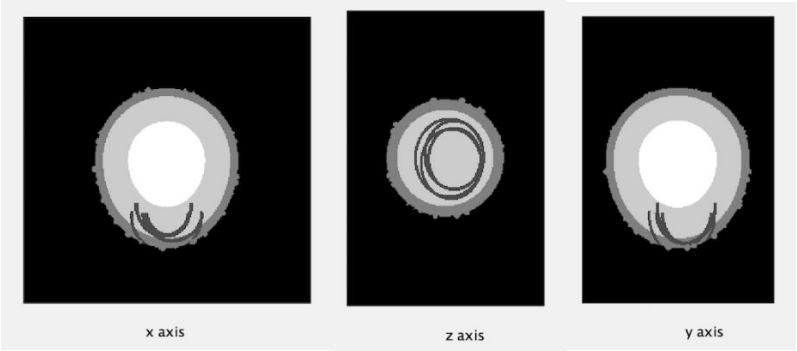



Figure 5.21 Bacon suffering from Stem-end rot at stage 1

Stage 2

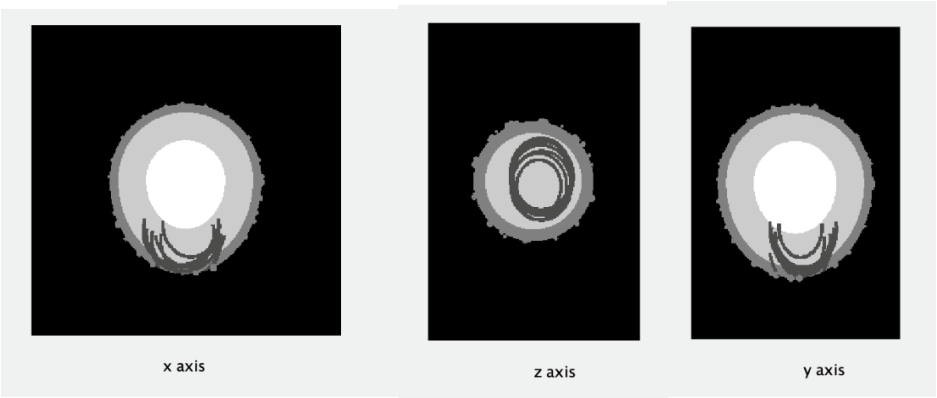



Figure 5.22 Bacon suffering from Stem-end rot at stage 2

Stage 3

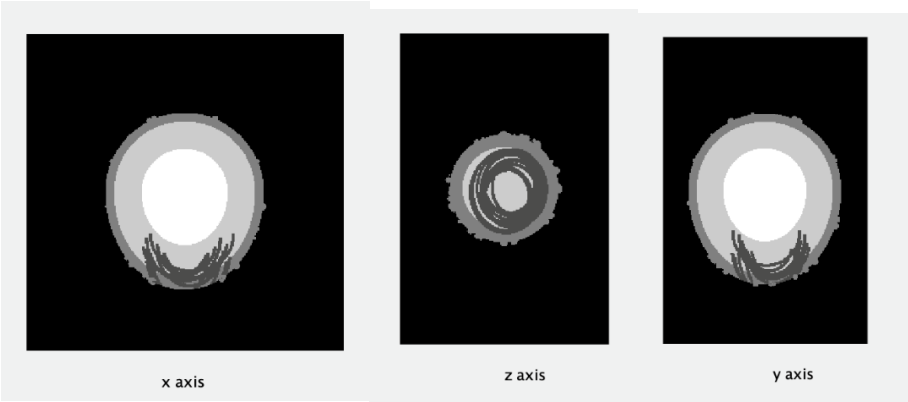



Figure 5.23 Bacon suffering from Stem-end rot at stage 3

Stage 4

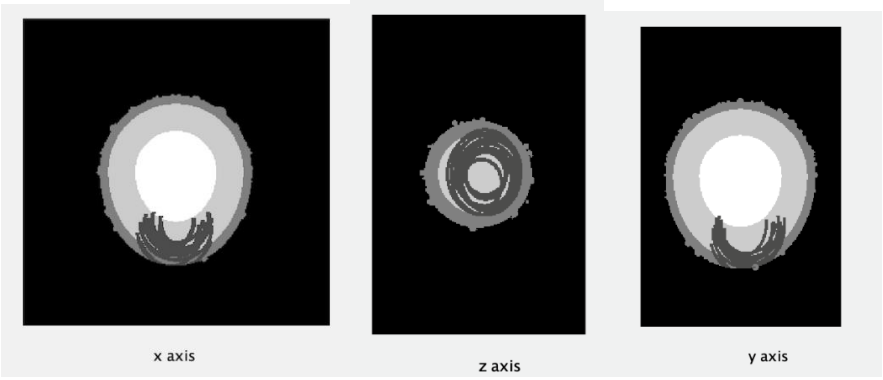



Figure 5.24 Bacon suffering from Stem-end rot at stage 4

Stage 5

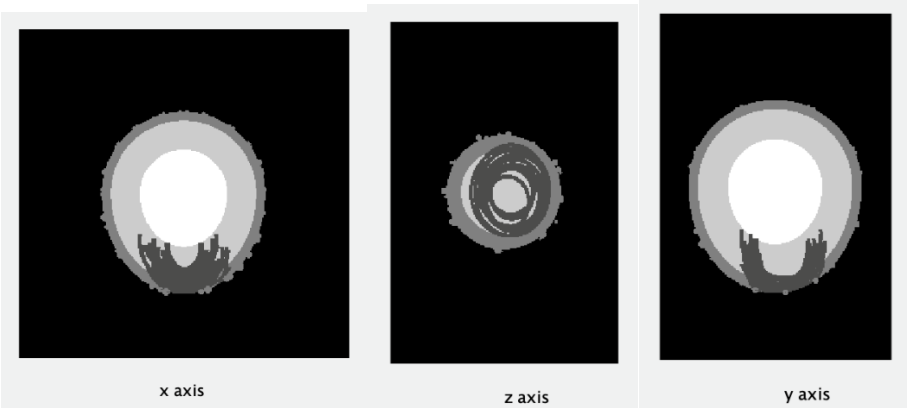



Figure 5.25 Bacon suffering from Stem-end rot at stage 5

Through the observation of the pictures, it was established that, as the value of the degree of decay increases, the decayed area of avocado also increases significantly. Anthracnose is the more obvious disease, probably because the rotted area appears cloud-like. This is more intuitively ‘rotten’ looking to the user than Stem-end rot. In the latter disease, the decay manifests in strips. When the value of decay is increased, the strip graph appears denser, but the visual experience for users is not as strong as that brought by Anthracnose. Overall, the desired ability to exhibit different degrees of decay by selecting different numbers is provided.

### 5.1.5 Randomness of rotten part

In order to test the randomness of the decay effect, the program was run multiple times. Since the variety and decay level of avocados have little effect on the random generation algorithm, it is only shown that when the Bacon avocado suffers from two diseases, the following different images are obtained under the same degree of decay.

- Bacon suffering from Anthracnose at stage 3

Test 1

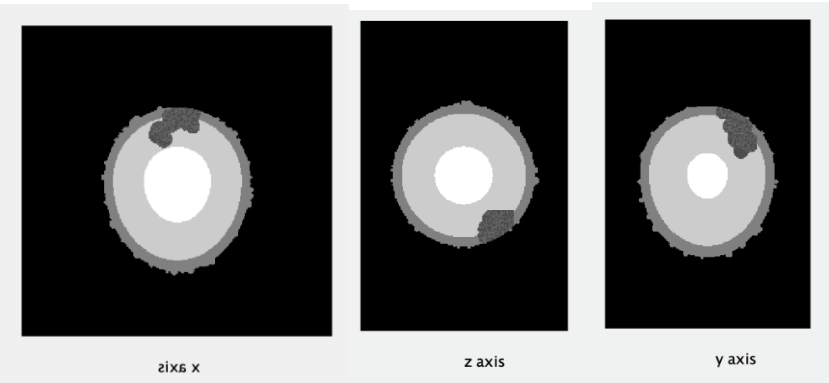



Figure 5.26 Bacon suffering from Anthracnose at stage 3(a)

Test 2

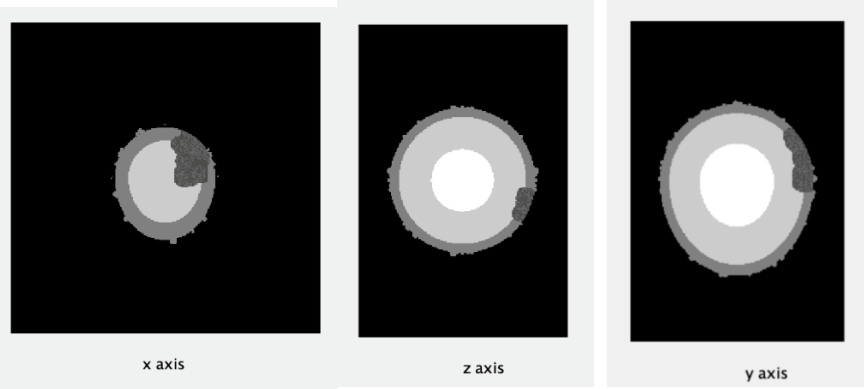



Figure 5.27 Bacon suffering from Anthracnose at stage 3(b)

Test 3

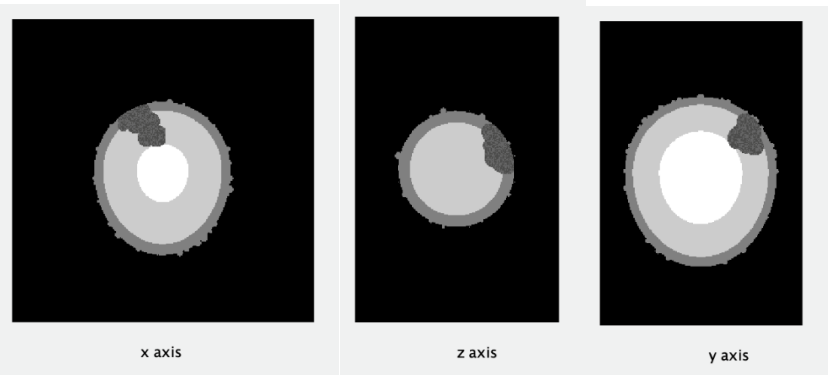



Figure 5.28 Bacon suffering from Anthracnose at stage 3(c)

- Bacon suffering from Stem-end rot at stage 3

Test 1

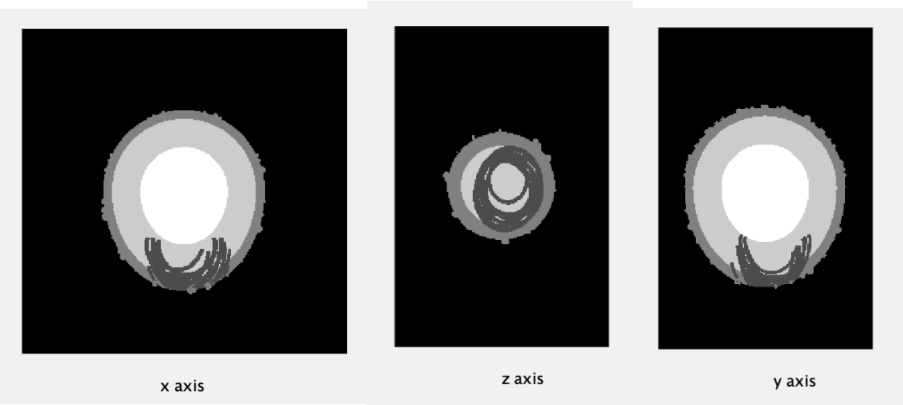



Figure 5.29 Bacon suffering from Stem-end rot at stage 3(a)

Test 2


Figure 5.30 Bacon suffering from Stem-end rot at stage 3(b)

Test 3

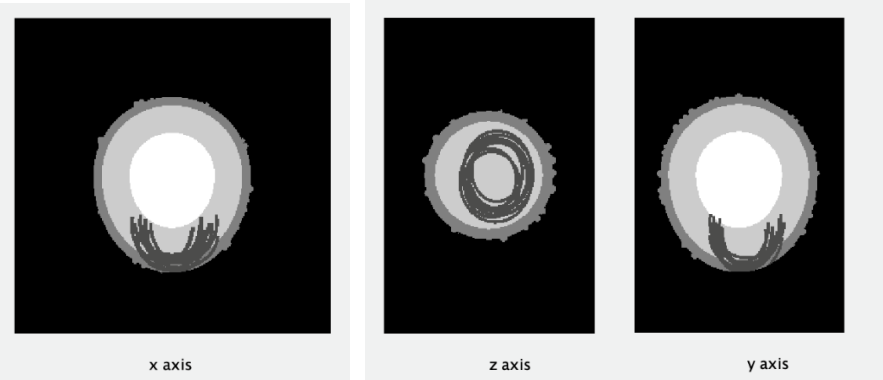



Figure 5.31 Bacon suffering from Stem-end rot at stage 3(c)

Because the rotted area of Anthracnose occurs across the whole fruit and the rotted area is larger, the effect of this disease is the more obvious of the two. As illustrated in the image, the area of decay looks very similar each time. Even if the same variety of avocado and the same degree of decay are selected, however, the decayed areas will not be exactly the same in any two models. This result shows that the randomness of Anthracnose's decay effect is achieved very well. For Stem-end rot, the decayed area is fixed on the head of the avocado, so the random effect is not as good as Anthracnose. If the results of the experiment are compared with great care, though, the differences in the generated decay areas can be observed. This shows that the randomness of the decay effect in this project has also been realized.

## 5.2 Performance Testing

Performance testing is essential to ensure the quality of the program. It tests both the stability and efficiency of the system under specific workloads. Performance testing is generally carried out to obtain performance metrics. These metrics include response time, Transactions Per Second (TPS), throughput and the resource utilization of the program. If a program is very inefficient it is unlikely to be accepted by the public, and it then becomes a meaningless product. In order to test the efficiency of the project, the probe function that comes with MATLAB was employed. This was based on the results returned by the profile function and was used to view the time and memory occupied by the program.

The following table shows the results displayed by the detector interface after making different selections on the gui interface.

Table 5.1 Anthracnose, 1 ZUTANO, Stage 1 (a)

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 14.950 s | 8.986 s | 478944.00Kb | 288224.00Kb | 140376.00Kb |
| factor | 3 | 5.964 s | 5.964 s | 337864.00Kb | 287520.00Kb | 50344.00Kb |
| gui_mainfcn | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |

Table 5.2 Anthracnose, 1 ZUTANO, Stage 1 (b)

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 9.303 s | 8.924 s | 422700.00Kb | 283848.00Kb | 138852.00Kb |
| factor | 3 | 0.379 s | 0.379 s | 281800.00Kb | 281800.00Kb | 0.00Kb |
| gui_mainfcn | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |

Table 5.3 Anthracnose, 3 BACON, Stage 1

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 12.219 s | 8.774 s | 424896.00Kb | 281932.00Kb | 140904.00Kb |
| factor | 3 | 3.445 s | 3.445 s | 283944.00Kb | 281884.00Kb | 2060.00Kb |
| gui_mainfcn | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |

Table 5.4 Anthracnose, 5 REED, Stage 1

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 15.470 s | 8.711 s | 423612.00Kb | 282400.00Kb | 141212.00Kb |
| factor | 3 | 6.759 s | 6.759 s | 282340.00Kb | 282340.00Kb | 0.00Kb |
| gui_mainfcn | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |

Table 5.5 Anthracnose, 1 ZUTANO, Stage 3

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 14.920 s | 8.766 s | 1001260.00Kb | 855672.00Kb | 140900.00Kb |
| factor | 3 | 6.153 s | 6.153 s | 860360.00Kb | 855672.00Kb | 4688.00Kb |
| gui_mainfcn | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |

Table 5.6 Anthracnose, 1 ZUTANO, Stage 5

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 31.050 s | 8.916 s | 1671708.00Kb | 1422964.00Kb | 140868.00Kb |
| factor | 7 | 22.133 s | 22.133 s | 1530808.00Kb | 1422932.00Kb | 107876.00Kb |
| gui_mainfcn | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |

Table 5.7 Stem-end rot, 1 ZUTANO, Stage 1 (a)

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 41.788 s | 41.788 s | 140968.00Kb | 108.00 Kb | 140860.00Kb |
| factor | 1 | 0.000 s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui_mainfcn | 1 | 0.000 s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000 s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000 s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |

Table 5.8 Stem-end rot, 1 ZUTANO, Stage 1 (b)

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 45.472s | 45.472s | 142024.00Kb | 4792.00Kb | 137232.00Kb |
| factor | 1 | 0.000 s | 0.000 s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui_mainfcn | 1 | 0.000 s | 0.000 s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000 s | 0.000 s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000 s | 0.000 s | 0.00Kb | 0.00Kb | 0.00Kb |

Table 5.9 Stem-end rot, 3BACON, Stage 1

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 40.950s | 40.950s | 141292.00Kb | 392.00Kb | 140900.00Kb |
| factor | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui_mainfcn | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |

Table 5.10 Stem-end rot, 5 REED, Stage 1

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 42.040s | 42.040s | 147176.00Kb | 6952.00Kb | 140224.00Kb |
| factor | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui_mainfcn | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |

Table 5.11 Stem-end rot, 5 REED, Stage 3

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 41.765 s | 41.765 s | 146572.00Kb | 6936.00Kb | 139636.00Kb |
| factor | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui_mainfcn | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |

Table 5.12 Stem-end rot, 5 REED, Stage 5

| Function name | calls | total time | Self-time | Allocated memory | Freed memory | Self-memory |
|---|---|---|---|---|---|---|
| gui>pushbutton1_Callback | 1 | 110.248s | 110.248s | 146512.00Kb | 3056.00Kb | 143456.00Kb |
| factor | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui_mainfcn | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| gui | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |
| ...',hObject,eventdata,guidata(hObject)) | 1 | 0.000s | 0.000s | 0.00Kb | 0.00Kb | 0.00Kb |

Of course, this project has not only been tested a few times, but these are typical test data. After the data was obtained, the following analysis is performed on the program.

### 5.2.1 Memory consumption

For Anthracnose, since the disease is generated not only using the gui>pushbutton1_Callback function, but also calling the factor function from the factor.m file, both of which take up a certain amount of memory. The model of healthy disease is still implemented by the pushbutton1_Callback function, so this function takes up more memory than the other.

By comparing Table 5.1 and Table 5.2, it can be found that after the user makes the same choice (disease, type, stage), the data obtained is similar but different. The reason for this could be that the random generation algorithm is used in the program, and the selection of the value is not fixed, so the memory occupied is different.

By comparing Table 5.1, Table 5.2, Table 5.3, Table 5.4, it can be found that the memory occupied by the function does not change much. Obviously, after the different selection of the cultivars in the program, only the parameters of the function were changed. there is no effect on the running of the program, and the occupied memory has not changed.

Comparing Table 5.1, Table 5.2 with Table 5.5, Table 5.6, it can be found that as the degree of decay increases, the memory occupied by the function also gradually increases. It can also be seen from the number of factory function calls shown in the table. As the degree of decay increases, the number of calls also increases, and the program also needs more memory to run.

For Stem-end rot, only the gui.m file is used.

By comparing Table 5.7 and Table 5.8, it can be analyzed that when the program runs by the same selection twice, the obtained data is different, but

similar, which illustrates that the random generation algorithm was successfully used in the procedure of this disease.

Comparing Table 5.7, Table 5.8, Table 5.9 and Table 5.10, it can be found that the choice of the avocado disease has little effect on the size of the occupied memory.

After analyzing Table 5.7, Table 5.8, Table 5.11, and Table 5.12, it is unexpectedly discovered that as the degree of decay increased, the size of the occupied memory was not affected. This result shows that in the pushbutton1_Callback function, the memory used to generate the avocado rot portion is insignificant compared to the memory required for the entire program to run.

The calculations show that the average memory occupied by the system is 737,186.667Kb when running the program that produces Anthracnose's avocado. When the program generates avocados that suffer from Stem-end rot, the average memory used is 144090.667Kb.

### 5.2.2 Processing speed

After analyzing the experimental results, it can be seen that the regular of the time required to generate the two diseases separately is similar.

By comparing Tables 5.1 and 5.2, Table 5.7 and Table 5.8, the two sets of data can show that after the user has done the same selection on the gui for two times, the time may have a gap of 4 seconds to 5 seconds, which indicates that the generation algorithm has a greater impact on the running speed of this program.

Comparing Table 5.1, Table 5.2, Table 5.3, Table 5.4, and Table 5.7, Table 5.8, Table 5.9, Table 5.10, respectively, it can be concluded that the selection of different varieties of avocado has no significant effect on the running time of the program.

Tables 5.1, 5.2, 5.5, and 5.6 are grouped, and Table 5.7, Table 5.8, Table 5.11, and Table 5.12 are grouped. After comparison between two groups, it can be found that the difference in stage selection has a significant impact on program run time. Generating a larger decay area takes longer to run the program.

After calculation, it can be concluded that it takes 16.304 seconds averagely to generate the model of avocado suffering from Anthracnose, while the related program needs about 53.711 seconds to produces the avocado with Stem-end rot, which is much longer than the time required to run the program for another disease.

The time and memory required to run this program are normal and within the user's acceptable range.

# 6.Conclusions and Future work

## 6.1 Conclusions

Recently, the avocado has become a very popular fruit, both due to its flavor and very rich nutritional content. It has become an indispensable food in many countries, those that grow avocados and those that must import them. Additionally, the fruit is used in the cosmetics industry, medical industry and other industries other than the food industry. As such, demand for avocado has also increased rapidly. However, the majority of avocados grow in the tropical and subtropical regions, which means that transportation is important to the sale and availability of avocados. Due to their bio-uniqueness and the latent of pathogens after illness, however, avocados are prone to decay during this transportation. This sometimes leads to massive economic losses. To study and control the diseases of avocados, the research team is trying to develop a device to detect avocado disease. This, of course, requires virtual data on avocados that do not currently exist. This project created a simulation of the diseased avocado generated a three-dimensional model and provided data. Before the establishment of the model, a large number of literature surveys were conducted to study the different varieties of avocados, different disease types, health, and disease patterns, and also to study some theoretical knowledge and technical background. All of this hugely helped with the implementation of the practical part of this project. Next, the mathematical modeling of avocado in various states was carried out and an algorithm was designed to realize the three-dimensional model of different varieties of avocado after suffering from different diseases. MATLAB language was used in MATLAB software to draw the model using the generated algorithm. Finally, to facilitate the user in viewing the generated model, a human-computer interaction interface was designed. At the interface, the user can select the diseases, varieties of avocado and decay stages for the fruit. They can then

view the avocado from three directions. After the program was completed, it was functional testing and performance testing, and the results were analyzed in detail. All the requirements in the early stage of the project were realized to a very acceptable degree.

## 6.2 Future work

Analysis and comparison of the results reveal that there are still some defects and considerable room for improvement. If the existing problems can be resolved, the project will send electromagnetic waves to an avocado. The device provides a more accurate experimental data and experimental basis.

The results also show that if the result of the project is more accurate and the obtained avocado profile is closer to the real thing, the first step is to improve the algorithm. For the first type of disease, Anthracnose, it should try to distinguish the color of the different degrees of decay of the fruit and consider that when the decay is severe, the outside of the whole fruit will be deformed, and the score will also suffer from decay. The second is the second disease type Stem-end rot, which is linked to the need for further improvement. In this project, the decayed shape only simulates the strips, but sometimes there is blocky decay in the actual decay process. Simultaneous simulation of strip rot can also seek to employ other algorithms to make the decaying trend more natural and realistic. Furthermore, due to a large number of looping statements and random generation algorithms, runtime and memory footprint can also be enhanced by improved algorithms.

In addition to the algorithm, the number of avocado varieties is also worth increasing. Due to the time constraints, only five cultivars were implemented in this project, and a three-dimensional model for a wider variety of avocados could therefore also be considered for future research.

# References


[1] Morton JF (1987). "Avocado; In: Fruits of Warm Climates". Center for New Crops and Plant Products, Department of Horticulture and Landscape Architecture, Purdue University, West Lafayette, IN. pp. 91–102. ISBN 978-0-9610184-1-2.

[2] "What's in a name?". University of California. Retrieved March 27, 2016.

[3] Chen, H; Morrell, PL; Ashworth, V; de la Cruz, M; Clegg, MT (2008). "Tracing the Geographic Origins of Major Avocado Cultivars". Journal of Heredity. 100 (1): 56–65.

[4] "2001-2013 World avocado production, 2000-2011 World Avocado Exports,2002-2013 French avocado imports". Novagrim.com. Archived from the original on 2014-02-23. Retrieved 2014-02-01.

[5] Schroeder, C. A. 1953. Abnormal Fruit Types in the Avocado. California Avocado Society 1953-54 Yearbook. 38:121-124. http://www.avocadosource.com/cas_yearbooks/cas_38_195354/cas_1953-54_pg_121- 124.pdf

[6] Kader A.A. (2002). Postharvest biology and technology: an overview. In: Kader A.A. (ed.), Postharvest Technology of Horticultural Crops. Oakland, CA: University of California, Division of Agriculture and Natural Resources, publication 3311, pp. 39-47.

[7] Lee, SK & Coggins, CW, Jr, 1982. Feasibility of Marketing Soft Avocado Fruit. Calif Avocado Soc Yrb, 66, 57-62.

[8] Tokar, V., 2007. The history of commercial avocado ripening. Calif. Avocado Soc. Yearb. 90, 77–85.

[9] Dreher ML, Davenport AJ (2013). "Hass avocado composition and potential health effects". Crit Rev Food Sci Nutr. 53 (7): 738–50. doi:10.1080/10408398.2011.556759.

[10] "Avocado; In: Fruits of Warm Climates". Center for New Crops and Plant Products, Department of Horticulture and Landscape Architecture, Purdue University, West Lafayette, IN. pp. 91–102. ISBN 978-0-9610184-1-2

[11] Morton JF (1987). "Avocado; In: Fruits of Warm Climates". Center for New Crops and Plant Products, Department of Horticulture and Landscape Architecture, Purdue University, West Lafayette, IN. pp. 91–102. ISBN 978-0-9610184-1-2.

[12] Storey, W. B. (1973). "What kind of fruit is the avocado?". California Avocado Society 1973–74 Yearbook. 57: 70–71

[13] Yahia, E. (2011). Postharvest Biology and Technology of Tropical and Subtropical Fruits. Sawston: Woodhead Publishing Limited

[14] Araújo, R., Rodriguez-Jasso, R., Ruiz, H., Pintado, M. and Aguilar, C. (2018). Avocado by-products: Nutritional and functional properties. Trends in Food Science & Technology, 80, pp.51-60

[15] Stradley, Linda (2004). All About Avocados: History of the Hass Avocado. What'sCookingAmerica.net. Newberg, OR: self-published. Retrieved 2008-05-13

[16] "The Hass Mother Tree: 1926–2002". Avocado.org. Irvine, CA: California Avocado Commission. 2008. pp. "About Avocados: History" section. Archivedfrom the original on 13 September 2008. Retrieved 2008-09-27

[17] En.wikipedia.org. (2019). Avocado. [online] Available at: https://en.wikipedia.org/wiki/Avocado#A_cultivars [Accessed 12 Aug. 2019]

[18] Ohr, H. D., Coffer, M. D., McMillan, R. T. (2003). Diseases of Avocado (Persea americana Miller). The American Phytopathological Society.

[19] Common Names of Diseases, The American Phytopathological Society. https://www.apsnet.org/edcenter/resources/commonnames/Pages/Avocado.aspx

[20] Nelson, C. Scot (2008) Mango Anthracnose (Colletotrichum gloeosporioides). University of Hawaii at Manoa cooperative extension service.

[21] Nelson, C. Scot (2008) Mango Anthracnose (Colletotrichum gloeosporioides). University of Hawaii at Manoa cooperative extension service.

[22] G.M. Sanders, L. Korsten (1997) Market survey of stem-end rot and anthracnose on Fuerte avocados and comparison of Colletotrichum gloeosporioidesisolates from different avocado producing areas. S. Afr. Avocado Growers' Assoc. Yearb., 20, pp. 101-105

[23] J. Smith, L. Korsten (1996) Microbes associated with the avocado flower and fruit: the good, the bad and the ugly. S. Afr. Avocado Growers' Assoc. Yearb., 19, pp. 39-40

[24] Smith and Korsten (1996) J. Smith, L. KorstenMicrobes associated with the avocado flower and fruit: the good, the bad and the ugly S. Afr. Avocado Growers' Assoc. Yearb., 19, pp. 39-40

[25] Demoz, B. and Korsten, L. (2006). Bacillus subtilis attachment, colonization, and survival on avocado flowers and its mode of action on stem-end rot pathogens. Biological Control, 37(1), pp.68-74.

[26] Korsten, L., 1993. Biological control of avocado fruit diseases. Ph.D. thesis, University of Pretoria, Pretoria.

[27] Dev, P. (2019). Lifecycle of avocado anthracnose. [online] Far Eastern Agriculture. Available at: http://www.fareasternagriculture.com/crops/agriculture/lifecycle-of-avocado-anthracnose [Accessed 11 Aug. 2019].

[28] Bill, M., Sivakumar, D., Korsten, L. and Thompson, A. (2014). The efficacy of combined application of edible coatings and thyme oil in inducing resistance components in avocado (Persea americana Mill.) against anthracnose during post-harvest storage. Crop Protection, 64, pp.159-167.

[29] G.M. Sanders, L. Korsten (1997) Market survey of stem-end rot and anthracnose on Fuerte avocados and comparison of Colletotrichum gloeosporioidesisolates from different avocado producing areas S. Afr. Avocado Growers' Assoc. Yearb., 20, pp. 101-105

[30] G.M. Sanders, L. Korsten (2003) Comparison of cross inoculation potential of South African

[31] Image, P. (2019). PGM File Extension - What is a .pgm file and how do I open it? [online] Fileinfo.com. Available at: https://fileinfo.com/extension/pgm [Accessed 12 Aug. 2019].

[32] Personalpages.manchester.ac.uk. (2019). [online] Available at: https://personalpages.manchester.ac.uk/staff/fumie.costen/pastwork/DATA/readingmaterial/lectureEECEMLab.pdf [Accessed 12 Aug. 2019]

[33] En.wikipedia.org. (2019). Netpbm. [online] Available at: https://en.wikipedia.org/wiki/Netpbm [Accessed 25 Aug. 2019].

[34] Netpbm.sourceforge.net. (2019). PGM Format Specification. [online] Available at: http://netpbm.sourceforge.net/doc/pgm.html [Accessed 12 Aug. 2019].

[35] En.wikipedia.org. (2019). Netpbm format. [online] Available at: https://en.wikipedia.org/wiki/Netpbm_format [Accessed 12 Aug. 2019]

[36] Amnh.org. (2019). OLOGY. [online] Available at: https://www.amnh.org/ology/features/stufftodo_einstein/threed_dimension.php [Accessed 21 Aug. 2019].

[37] Marion, G. and Lawson, D, 2008. An introduction to Mathematical Modelling, University of Bristol, Bioinformatics and Statistics Scotland. [online] Available at: https://people.maths.bris.ac.uk/~madjl/course_text.pdf [Accessed: 24 Aug.2019].

[38] En.wikipedia.org. (2019). Mathematical model. [online] Available at: https://en.wikipedia.org/wiki/Mathematical_model#Elements_of_a_mathematical_model [Accessed 17 Aug. 2019].

[39] Mat.univie.ac.at. (2019). Mathematical Modeling. [online] Available at: https://www.mat.univie.ac.at/~neum/model.html [Accessed 20 Aug. 2019].

[40] Sfu.ca. (2019). [online] Available at: https://www.sfu.ca/~vdabbagh/Chap1-modeling.pdf [Accessed 25 Aug. 2019].

[41] σφαῖρα, Henry George Liddell, Robert Scott, A Greek-English Lexicon, on Perseus.

[42] En.wikipedia.org. (2019). Sphere. [online] Available at: https://en.wikipedia.org/wiki/Sphere#cite_note-1 [Accessed 21 Aug. 2019].

[43] Putnam, A. and Olmsted, J. (1949). Solid Analytic Geometry. The American Mathematical Monthly, 56(2), p.120.

[44] En.wikipedia.org. (2019). Ellipsoid. [online] Available at: https://en.wikipedia.org/wiki/Ellipsoid [Accessed 24 Aug. 2019].

[45] Mathinsight.org. (2019). Quadric surfaces - Math Insight. [online] Available at: https://mathinsight.org/quadric_surfaces [Accessed 24 Aug. 2019].

[46] Tutorial.math.lamar.edu. (2019). Calculus III - Quadric Surfaces. [online] Available at: http://tutorial.math.lamar.edu/Classes/CalcIII/QuadricSurfaces.aspx [Accessed 24 Aug. 2019].

[47] Encyclopedia Britannica. (2019). Ellipsoid | geometry. [online] Available at: https://www.britannica.com/science/ellipsoid [Accessed 24 Aug. 2019].

[48] WhatIs.com. (2019). What is random numbers? - Definition from WhatIs.com. [online] Available at: https://whatis.techtarget.com/definition/random-numbers [Accessed 26 Aug. 2019].

[49] Barker, Elaine; Barker, William; Burr, William; Polk, William; Smid, Miles (July 2012). "Recommendation for Key Management" (PDF). NIST Special Publication 800-57. NIST. Retrieved 19 August 2013.

[50] GeeksforGeeks. (2019). Pseudo Random Number Generator (PRNG) - GeeksforGeeks. [online] Available at: https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/ [Accessed 26 Aug. 2019].

[51] "Pseudorandom number generators". Khan Academy. Retrieved 2016-01-11.

[52] Definitions, P. and Hope, C. (2019). What is Pseudo-random? [online] Computerhope.com. Available at: https://www.computerhope.com/jargon/p/pseudo-random.htm [Accessed 26 Aug. 2019].

[53] Von Neumann, John (1951). "Various techniques used in connection with random digits" (PDF). National Bureau of Standards Applied Mathematics Series. 12: 36–38.

[54] EDUCBA. (2019). Introduction to Matlab | Component | Advantage & Disadvantage. [online] Available at: https://www.educba.com/introduction-to-matlab/ [Accessed 26 Aug. 2019].

[55] Cimss.ssec.wisc.edu. (2019). What is Matlab. [online] Available at: https://cimss.ssec.wisc.edu/wxwise/class/aos340/spr00/whatismatlab.htm [Accessed 26 Aug. 2019].

[56] MDN Web Docs. (2019). Callback function. [online] Available at: https://developer.mozilla.org/en-US/docs/Glossary/Callback_function [Accessed 28 Aug. 2019].

[57] Staff (1995–2012). "2. What Is Evaluation?". International Center for Alcohol Policies - Analysis. Balance. Partnership. International Center for Alcohol Policies. Archived from the original on 2012-05-04. Retrieved 13 May 2012.

[58] Sarah del Tufo (13 March 2002). "WHAT is evaluation?". Evaluation Trust. The Evaluation Trust. Retrieved 13 May2012.

[59] Prasad, Dr. K.V.K.K. (2008) ISTQB Certification Study Guide, Wiley, ISBN 978-81-7722-711-6, p. vi

[60] ISO/IEC/IEEE International Standard - Systems and software engineering. ISO/IEC/IEEE 24765:2010(E). 2010. pp. vol., no., pp.1–418, 15 Dec. 2010.

Figure 2.1 A cross-sectional view of avocado

http://ucavo.ucr.edu/General/FruitBerry.html

Figure 2.2 Hass Avocado

https://www.tomorrowsharvest.com/store/hass-avocado.html

Figure 2.3 Bacon Avocado

https://www.fourwindsgrowers.com/products/bacon-avocado

Figure 2.4 Fuerte Avocado

https://homeguides.sfgate.com/fuerte-avocado-tree-size-61271.html

Figure 2.5 Gwen Avocado

https://www.californiaavocado.com/avocado101/avocado-varieties

Figure 2.6 Lam Hass Avocado

https://www.ebay.com/itm/Lamb-Hass-Avocado-Grafted-Tree-3-Feet-Tall-/162874186761

Figure 2.7 Pinkerton Avocado

https://www.fourwindsgrowers.com/products/pinkerton-avocado

Figure 2.8 Reed Avocado

https://www.buyfruit.com.au/reed-avocado

Figure 2.9 Zutano Avocado

https://www.louiesnursery.com/plants/avocado-trees/zutano-avocado/

Figure 2.10(a) Early stage of SER

http://barmac.com.au/problem/stem-end-rot-in-avocado/

Figure 2.10(b) Early stage of SER

https://www.pinterest.com/pin/442126888413394840/

Figure 2.11(a) Mid stage of SER

http://postharvest.ucdavis.edu/Commodity_Resources/Fact_Sheets/Datastores/Fruit_English/?uid=8&ds=798

Figure 2.11(b) Mid stage of SER

https://www.pinterest.com/pin/442126888413394854/

Figure 2.12 Late stage of SER

https://www.dreamstime.com/detail-picture-ugly-rotten-avocado-interesting-inside-seed-can-be-seen-image108254816

Figure 2.13 Schematic diagram of radial cracking

https://anywood.com/news/detail/149164.html

Figure 2.14(a) Avocado suffering from Anthracnose

https://www.ctahr.hawaii.edu/oc/freepubs/pdf/pd-58.pdf

Figure 2.14(b) Avocado suffering from Anthracnose

https://www.ctahr.hawaii.edu/oc/freepubs/pdf/pd-58.pdf

Figure 2.15 Example of a PGM-format image

https://en.wikipedia.org/wiki/Netpbm_format#/media/File:Feep_netbpm_p2_pgm_example.png

Figure 2.16 A file in PGM format

https://personalpages.manchester.ac.uk/staff/fumie.costen/pastwork/DATA/readingmaterial/lectureEECEMLab.pdf

Figure 2.17 Three-dimensional coordinate system

https://www.intmath.com/vectors/6-3-dimensional-space.php

Figure 3.1 Depiction of the scientific mathematical model

https://www.sfu.ca/~vdabbagh/Chap1-modeling.pdf

Figure 3.2 Modelling diagram

https://www.mat.univie.ac.at/~neum/model.html

Figure 3.3 Five varieties of avocado

https://www.californiaavocado.com/avocado101/avocado-varieties

Figure 3.4 Three-dimensional graphic of a sphere

https://en.wikipedia.org/wiki/Sphere#/media/File:Sphere_and_Ball.png

Figure 3.5 Ellipsoid with cross-sections

https://blog.csdn.net/leemboy/article/details/79182490

Figure 3.6 Tri-axial ellipsoid

https://en.wikipedia.org/wiki/Ellipsoid#/media/File:Ellipsoide.svg

Figure 3.7 Spheroid

https://en.wikipedia.org/wiki/Ellipsoid#/media/File:Ellipsoide.svg

Figure 3.8 Sphere

https://en.wikipedia.org/wiki/Ellipsoid#/media/File:Ellipsoide.svg

Figure 3.12 Avocado skin

https://www.uooyoo.com/a/160630/12668.html

Figure 5.12 An avocado suffers from Anthracnose

http://edgeindiaagrotech.com/anthracnose

Figure 5.14 An avocado suffers from Stem-end rot

https://www.pinterest.com/pin/442126888413394854/

# Appendix

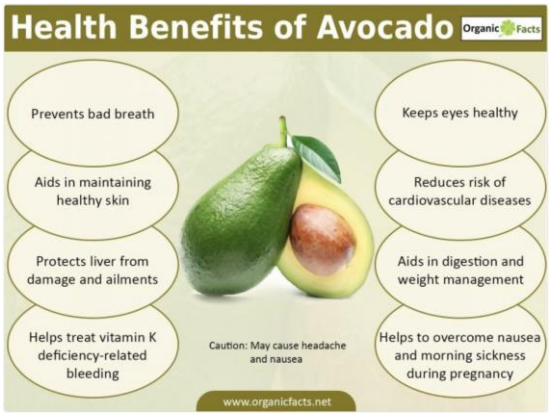
## Nutritional value of avocado

Avocado is a fruit in the medium-energy-density category. Most of the avocados contain 72% water and 6.8% dietary fiber. Its nutrient content is very similar to those of low-fat fruits and vegetables, so avocado can be used in weight control. Avocado contains a variety of fats. For typical avocados, fat provides about 75% of the total avocado mass, and 67% of all fat is essentially monounsaturated, such as oleic acid. Other major fats are palmitic acid and linoleic acid. According to data from adults on NHANES (National Health and Nutrition Examination Survey) 2001-2006, consumers who regularly eat avocados generally have higher HDL cholesterol than non-avocado consumers, and tend to exhibit smaller waist circumference, lighter weight and more At. They also have a lower risk of metabolic syndrome and, generally, exhibit better physical fitness.

Putting an avocado in a watery matrix, will result in the formation of an oil rich in monounsaturated fatty acids (MUFA). Such oil can add nutrients to the human body, while increasing the availability of phytochemicals, greatly reducing dietary fiber in food. Consumers will not feel the presence of plant fibers in avocados when they eat it. Although avocado oil is expensive to produce, it is rich in nutrients and has many unexpected uses in cooking and in cosmetic industries.

Avocados contain large amounts of potassium, phosphorus, magnesium, calcium and sodium, as well as other minerals, including iron and zinc. Foods that contain more potassium and less sodium are of great help for those who want to prevent cardiovascular disease. Also, avocados contain a variety of vitamins, such as beta-carotene, vitamin E, retinol, and ascorbic acid, which

are essential to maintain a healthy body. Overall avocado considerably benefits human health.



## The code in factor.m

```
function g=factor(o_x, o_y, o_z, o_r, x, y, z, r, graphics, h, w, d, n, e)
    nx = zeros(1, e);
    ny = zeros(1, e);
    nz = zeros(1, e);

    while(true)
        %disp(o_r);
        floor_o_r = floor(o_r) + 1;
        %n_x=2 * randi(floor_o_r) - floor_o_r + o_x;
        %n_y=2 * randi(floor_o_r) - floor_o_r + o_y;
        n_x = 2 * randi(floor_o_r - 1) + 1 - floor_o_r + o_x;
        n_r_1 = floor(sqrt(floor_o_r^2 - (n_x - o_x)^2));
        n_y = 2 * randi(n_r_1) - n_r_1 + o_y;
        n_z=sqrt(o_r^2 - (n_x - o_x)^2 - (n_y - o_y)^2) * sign(randi(1)-0.5) +
o_z ;
        if (n_x - x)^2 + (n_y - y)^2 + (n_z - z)^2 <= (r + 1)^2
            break;
        end
        disp('looping');
    end
```

```
for i = 1:e
    while(true)
        floor_o_r = floor(o_r);
        n_xx=2 * randi(floor_o_r) -floor_o_r + o_x;
        n_yy=2 * randi(floor_o_r) - floor_o_r + o_y;
        n_zz=2 * randi(floor_o_r) - floor_o_r + o_z;
        if (n_xx - x)^2 + (n_yy - y)^2 + (n_zz - z)^2 <= r^2
            break;
        end
        disp('looping');
    end
    nx(i) = n_xx;
    ny(i) = n_yy;
    nz(i) = n_zz;
end


n_r = o_r;
for ii=floor(o_x - 2 * o_r):floor(o_x + 2 * o_r)
    for jj=floor(o_y - 2 * o_r):floor(o_y + 2 * o_r)
        for kk=floor(o_z - 2 * o_r):floor(o_z + 2 * o_r)
```

```
                ant_distance2circle=(ii-n_x)^2+(jj-n_y)^2+(kk-n_z)^2;
                distance2circle=(ii-x)^2+(jj-y)^2+(kk-z)^2;
                if ant_distance2circle < n_r^2 && distance2circle<r^2
                    graphics(ii,jj,kk)=0.3;
                end
            end
        end
    end

    for ii=floor(o_x - 2 * o_r):floor(o_x + 2 * o_r)
        for jj=floor(o_y - 2 * o_r):floor(o_y + 2 * o_r)
            for kk=floor(o_z - 2 * o_r):floor(o_z + 2 * o_r)
                for ll=1: e
                    ant_distance2circle=(ii-nx(ll))^2+(jj-ny(ll))^2+(kk-
nz(ll))^2;
                    distance2circle=(ii-x)^2+(jj-y)^2+(kk-z)^2;
                    if ant_distance2circle < n_r^2 && distance2circle<r^2
                        graphics(ii,jj,kk)=(randi(2) + 1) / 10 + 0.1;
                    end
                end
            end
        end
```

```
        end


        if n == 0
            g=graphics;
            return;
        end
        g=factor(n_x, n_y, n_z, n_r, x, y, z, r, graphics, h, w, d, n - 1, e);


end
```

## The code in gui.m

```
function varargout = gui(varargin)


gui_Singleton = 1;
gui_State = struct('gui_Name',            mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn', @gui_OpeningFcn, ...
                   'gui_OutputFcn',    @gui_OutputFcn, ...
                   'gui_LayoutFcn',    [] , ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end


if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end


function gui_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui (see VARARGIN)

handles.size = 10;
handles.count = 10;
handles.cnt = 40;
handles.len = 25;
handles.type = 30;
% Choose default command line output for gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
```

```
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject       handle to pushbutton1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)
%generate_graphics;

graphics=zeros(300,300,200);
[height,width,deepth]=size(graphics);



%% shape_1

circle_x=height/2;
```

```
circle_y=width/2;

circle_z=deepth/2;

r_circle=height/4;


ellipse_x=height/2;

ellipse_y=width/2;

ellipse_z=deepth/2;


a=r_circle + handles.type;

b=r_circle;

c=r_circle;


peel=5;

ci=5;


value1=0.5;

value2=0.8;

value3=1;



ant_r_circle=8; %floor(r_circle / 20);
```

```
% Ᵽ

ant_peel_r_circle=ant_r_circle;


% x

ant_x=-randi(r_circle) + circle_x;

% y

%ant_y=2 * randi(r_circle) - r_circle + circle_y;

n_r_1 = floor(sqrt(r_circle^2 - (ant_x - circle_x)^2));

ant_y = 2 * randi(n_r_1) - n_r_1 + circle_y;


% z

ant_z=sqrt(r_circle^2 - (ant_x - circle_x)^2 - (ant_y - circle_y)^2) *
sign(randi(1)-0.5) + circle_z ;



for ii=1:height

    for jj=1:width

        for kk=1:deepth


            distance2circle=(ii-circle_x)^2+(jj-circle_y)^2+(kk-circle_z)^2;
```

```
distance2ellipse=(ii-ellipse_x)^2/(a^2)+(jj-ellipse_y)^2/(b^2)+(kk-ellipse_z)^2/(c^2);


if abs(distance2circle-r_circle^2)<peel && distance2circle>r_circle^2 && ii<height/2 && (ii - ant_x)^2 + (jj - ant_y)^2 + (kk - ant_z)^2 > (ant_r_circle + 10)^2
    r_circle_mini=peel;
    if mod(randi([1,10]),3)==0
        r_circle_mini=rand*r_circle/ci;
    end
    for iii=-10:10
        for jjj=-10:10
            for kkk=-10:10
                distance2circle_t=iii^2+jjj^2+kkk^2;
                if distance2circle_t<r_circle_mini
                    graphics(ii+iii,jj+jjj,kk+kkk)=value1;
                end
            end
        end
    end
end
```

```
                if abs(distance2ellipse-1)<0.001 && distance2ellipse<1 &&
ii>=height/2
                    if mod(randi([1,10]),3)==0
                        r_circle_mini=rand*r_circle/ci;
                        for iii=-10:10
                            for jjj=-10:10
                                for kkk=-10:10
                                    distance2circle_t=iii^2+jjj^2+kkk^2;
                                    if distance2circle_t<r_circle_mini
                                        graphics(ii+iii,jj+jjj,kk+kkk)=value1;
                                    end
                                end
                            end
                        end
                    end
                end

                if distance2circle<r_circle^2 && ii<height/2
                    graphics(ii,jj,kk)=value1;
                end
                if distance2ellipse<1 && ii>=height/2
                    graphics(ii,jj,kk)=value1;
```

```
        end


        if distance2circle<r_circle^2*0.8 && ii<height/2
            graphics(ii,jj,kk)=value2;
        end
        if distance2ellipse<0.8 && ii>=height/2
            graphics(ii,jj,kk)=value2;
        end


        if distance2circle<r_circle^2*0.3 && ii<height/2
            graphics(ii,jj,kk)=value3;
        end
        if distance2ellipse<0.3 && ii>=height/2
            graphics(ii,jj,kk)=value3;
        end


        ant_distance2circle=(ii-ant_x)^2+(jj-ant_y)^2+(kk-ant_z)^2;
        %if ant_distance2circle<ant_r_circle^2 &&
distance2circle<r_circle^2
        %       graphics(ii,jj,kk)=value1;
        %end
```

```
                %if ant_distance2circle<ant_peel_r_circle^2 &&
distance2circle<r_circle^2
                %       graphics(ii,jj,kk)=0.3;
                %end
            end
        end
end

graphics = factor(ant_x, ant_y, ant_z, ant_r_circle, circle_x, circle_y, circle_z,
r_circle, graphics, height, width, deepth, handles.count, handles.size);



% graphics=imgaussfilt3(graphics,1);

for ii=1:width
    if ii < deepth
        axes(handles.axes1);
        image=graphics(:,:,ii);
        imshow(image)
    end
    axes(handles.axes2);
```

```
        image=graphics(ii,:,:);
        image=reshape(image,[width,deepth]);
        imshow(image)

        axes(handles.axes3);
        image=graphics(:,ii,:);
        image=reshape(image,[height,deepth]);
        imshow(image)
end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%generate_graphics;

graphics=zeros(300,300,200);
[height,width,deepth]=size(graphics);
```

```
%% shape_1

circle_x=height/2;
circle_y=width/2;
circle_z=deepth/2;
r_circle=height/4;


ellipse_x=height/2;
ellipse_y=width/2;
ellipse_z=deepth/2;

a=r_circle + handles.type;
b=r_circle;
c=r_circle;

peel=5;
ci=5;

value1=0.5;
value2=0.8;
```

```
value3=1;



ant_r_circle=10; %floor(r_circle / 20);


% Ƥ

ant_peel_r_circle=ant_r_circle - peel;


% x

%ant_x=-randi(r_circle) + circle_x;

% y

%ant_y=2 * randi(r_circle) - r_circle + circle_y;

% z

%ant_z=sqrt(r_circle^2 - (ant_x - circle_x)^2 - (ant_y - circle_y)^2) *
sign(randi(1)-0.5) + circle_z ;


ant_x = circle_x + 50;

ant_y = circle_y;

ant_z = circle_z;


f = handles.cnt;
```

```
e = handles.len;


disp(f)

disp(e)


n_r = 10;


%nx = zeros(1, e * f);

%ny = zeros(1, e * f);

%nz = zeros(1, e * f);


nx = zeros(1, e);

ny = zeros(1, e);

nz = zeros(1, e);

ndd = zeros(1, e);

ndd1 = zeros(1, e);

ndd2 = zeros(1, e);

disp(nx)


cnt = 1;
```

```
for j = 1: e
    x1 = 2 * randi(n_r - 1) + 1 - n_r + ant_x;
    n_r_1 = floor(sqrt(n_r^2 - (x1 - ant_x)^2));
    y1 = 2 * randi(n_r_1) - n_r_1 + ant_y;
    z1 = sqrt(n_r^2 - (x1 - ant_x)^2 - (y1 - ant_y)^2) * sign(randi(1)-0.5) +
ant_z ;
    nx(j) = x1;
    ny(j) = y1;
    nz(j) = z1;
    ndd(j) = 2 * randi(8) - 8;
    ndd1(j) = 2 * randi(6) - 6;
    ndd2(j) = 2 * randi(10) - 10;
end
%for j = 1: f
%     x1 = 2 * randi(n_r - 1) + 1 - n_r + ant_x;
%     n_r_1 = floor(sqrt(n_r^2 - (x1 - ant_x)^2));
%     y1 = 2 * randi(n_r_1) - n_r_1 + ant_y;
%     z1 = sqrt(n_r^2 - (x1 - ant_x)^2 - (y1 - ant_y)^2) * sign(randi(1)-0.5) +
ant_z ;
%     for i = 1:e
%         nx(cnt) = i * (x1 - ant_x) + x1;
%         ny(cnt) = i * (y1 - ant_y) + y1;
```

```
%             nz(cnt) = i * (z1 - ant_z) + z1;
%             %disp(cnt)
%             cnt = cnt + 1;
%       end
%end



for ii=1:height
      for jj=1:width
            for kk=1:deepth

                  distance2circle=(ii-circle_x)^2+(jj-circle_y)^2+(kk-circle_z)^2;

                  distance2ellipse=(ii-ellipse_x)^2/(a^2)+(jj-ellipse_y)^2/(b^2)+(kk-ellipse_z)^2/(c^2);

                  if abs(distance2circle-r_circle^2)<peel && distance2circle>r_circle^2 && ii<height/2 && (ii - ant_x)^2 + (jj - ant_y)^2 + (kk - ant_z)^2 > (ant_r_circle + 10)^2
                        r_circle_mini=peel;
                        if mod(randi([1,10]),3)==0
```

```
                r_circle_mini=rand*r_circle/ci;
            end
            for iii=-10:10
                for jjj=-10:10
                    for kkk=-10:10
                        distance2circle_t=iii^2+jjj^2+kkk^2;
                        if distance2circle_t<r_circle_mini
                            graphics(ii+iii,jj+jjj,kk+kkk)=value1;
                        end
                    end
                end
            end
        end

        if abs(distance2ellipse-1)<0.001 && distance2ellipse<1 &&
ii>=height/2
            if mod(randi([1,10]),3)==0
                r_circle_mini=rand*r_circle/ci;
                for iii=-10:10
                    for jjj=-10:10
                        for kkk=-10:10
                            distance2circle_t=iii^2+jjj^2+kkk^2;
```

```
                        if distance2circle_t<r_circle_mini
                            graphics(ii+iii,jj+jjj,kk+kkk)=value1;
                        end
                    end
                end
            end
        end
    end

if distance2circle<r_circle^2 && ii<height/2
    graphics(ii,jj,kk)=value1;
end
if distance2ellipse<1 && ii>=height/2
    graphics(ii,jj,kk)=value1;
end

if distance2circle<r_circle^2*0.8 && ii<height/2
    graphics(ii,jj,kk)=value2;
end
if distance2ellipse<0.8 && ii>=height/2
    graphics(ii,jj,kk)=value2;
```

```
        end


        if distance2circle<r_circle^2*0.3 && ii<height/2
            graphics(ii,jj,kk)=value3;
        end
        if distance2ellipse<0.3 && ii>=height/2
            graphics(ii,jj,kk)=value3;
        end


        ant_distance2circle=(ii-ant_x)^2+(jj-ant_y)^2+(kk-ant_z)^2;
        %if ant_distance2circle<ant_r_circle^2 &&
distance2circle<r_circle^2
        %       graphics(ii,jj,kk)=value1;
        %end


        %if ant_distance2circle<ant_peel_r_circle^2 %&&
distance2circle<r_circle^2
        %       graphics(ii,jj,kk)=0.3;
        %end


        for i = 1:e
            dd = ndd(i);
```

```
        dd1 = ndd1(i);
    %for i = 1: 3
        %distance = (ii-nx(i))^2+(jj-ny(i))^2+(kk-nz(i))^2;
        %if distance<(n_r)^2 && distance2ellipse<1
        %       graphics(ii,jj,kk)=0.3;
        %end
        %disp(i)
      %%

            aa = 40 + dd;
            bb = 35 + dd1;
            cc = 30 + dd1;

            aa1 = 37 + dd;
            bb1 = 32 + dd1;
            cc1 = 27 + dd1;

            distance2ellipse1=(ii-nx(i))^2/(aa^2)+(jj-
ny(i))^2/(bb^2)+(kk-nz(i))^2/(cc^2);
            distance2ellipse2=(ii-nx(i))^2/(aa1^2)+(jj-
ny(i))^2/(bb1^2)+(kk-nz(i))^2/(cc1^2);
```

```
                    if distance2ellipse1<1 && distance2ellipse2 > 1 && ii >
nx(i) + ndd2(i) && distance2ellipse < 1
                        graphics(ii,jj,kk)=0.3;
                    end
                %%

            end

        end
    end
end
for ii=1:width
    if ii < deepth
        axes(handles.axes1);
        image=graphics(:,:,ii);
        imshow(image)
    end
    axes(handles.axes2);
    image=graphics(ii,:,:);
    image=reshape(image,[width,deepth]);
    imshow(image)
```

```
        axes(handles.axes3);
        image=graphics(:,ii,:);
        image=reshape(image,[height,deepth]);
        imshow(image)
end

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
str = get(hObject, 'String');
val = get(hObject,'Value');
% Set current data to the selected data set.
switch str{val};
case '1 ZUTANO'
    handles.type = 30;
```

```
case '2 HASS'
    handles.type = 24;
case '3 BACON'
    handles.type = 18;
case '4 GWEN'
    handles.type = 12;
case '5 REED'
    handles.type = 6;
end
guidata(hObject,handles)
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
end
% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


str = get(hObject, 'String');
val = get(hObject,'Value');
% Set current data to the selected data set.
switch str{val};
case '5'
    handles.size = 10;
    handles.count = 10;
    handles.cnt = 40;
    handles.len = 25;
case '4'
    handles.size = 8;
    handles.count = 8;
    handles.cnt = 32;
    handles.len = 16;
case '3'
    handles.size = 6;
    handles.count = 6;
```

```
    handles.cnt = 24;
    handles.len = 12;
case '2'
    handles.size = 4;
    handles.count = 4;
    handles.cnt = 16;
    handles.len = 8;
case '1'
    handles.size = 1;
    handles.count = 2;
    handles.cnt = 8;
    handles.len = 4;
end
guidata(hObject,handles)


% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```