BibTeX:

@book{cgkp,

author = "John F. Hughes",

Title = {Computer Graphics: Principles & Practice },

publisher = "Addison-Wesley",

YEAR = 2014,

}

Nicholas-Sweeney-Week-1-2020-9-8-1: Chapter 1 CS Graphics Book

The first chapter to the book written by John F. Hughes talks about what Computer Graphics stands for at the core. He starts off by saying that Computer Graphics usually ties into the communication that goes from computer-to-human and that the way the human is able to respond is through input devices such as keyboards and controllers. Since the main thing that needs to be straight forward in these interactions is the perceived image the human gets from the computer this is what computer graphics ends up revolving around. With computer graphics the typical goal that someone has when developing a game is to reach the point where they are at the same level as a human’s perception of graphics. This means that there is no point in aiming to have graphics that are better than what the human eye can see because at that point it will just end up looking the same regardless of any improvements that are made. Obviously some applications have a simplistic look for graphics but this is just proof that presentation and the orientation you have using computer graphics is just as important as high-definition graphics. To summarize this chapter was written simply to get users with little knowledge of the usage for computer graphics and paint them a simple picture to give them a basic understanding of the concept.

Nicholas-Sweeney-Week-1-2020-9-8-2: Chapter 2 CS Graphics Book

The second chapter of Computer Graphics: Principles and Practice by Hughes starts to talk about different applications that can be used to create 2D computer graphics. One application they start the chapter off talking about is WPF. It has a variety of utility for a user to be able to create 2D as well as 3D graphics which is incredibly useful in terms of someone who is looking to learn and needs to start off with something that’s simple in 2D and slowly progress to the point where they have a large enough understanding of graphics to move into 3D applications. The application itself is based on the same graphics programming style as HTML which makes it very easy for someone with a lot of experience in HTML to move over to WPF without losing all of their previous knowledge that they had before. On top of this it allows for very rapid testing of your application which is king for any programmer because you will always have plenty of bugs when you first start out coding and the only way to fix them is to continuously tweak your code and rebuild it. This chapter summarized a variety of applications such as this one and all of them highlighted the positives and negatives that come with each different computer graphics programming application.

Nicholas-Sweeney-Week-2-2020-9-15-1: Chapter 3 CS Graphics Book

As Hughes continues to write about different aspects of computer graphics he begins to dive into the intricacies that go along with relational databases. He highlights a few flaws that the current relation schemas used in databases have. These flaws are described as anomalies and one of the ones Hughe talks about specifically is functional dependencies. These functional dependencies basically can be summarized as anytime there are R tuples that have the same value then all of these tuples must have the same value for a separate attribute. This leads to major discrepancies due to the fact that this puts a lot of faith in the database to always have logical data but sometimes that won’t be the case. An example of this being an anomaly would be in a hospital database where two babies ended up being born at the same time with the same first name which logically would make the method of functional dependency assume that this is the same record but then the two babies could have different last names and this would end up ruining the current method of functional dependency.

Nicholas-Sweeney-Week-2-2020-9-15-2: Chapter 4 CS Graphics Book

The book Computer Graphics: Principles and Practice by John F. Hughes begins to move into more complex topics as you move towards the middle of the book. The fourth chapter specifically talks about creating a 2D “test bed”. As this chapter continues Hughes talks directly about how after reading the previous chapter on WPF that the reader knows a basic understanding how programming works in computer graphics. Breaking down computer graphics you typically use two languages of code to develop your program. Personally, I use HTML to develop the environment and JavaScript to develop the functionality but in the case of this chapter Hughes uses XAML and C# respectively. The 2D test bed that this chapter references basically is a guide for a user to get comfortable with the syntax of code for computer graphics as well as the different things they can very easily produce on the screen such as polylines. From here the user is able to copy code into their own source files to create shapes on their own accord by altering this code and are able to see those changes take place every time they save their code.

Nicholas-Sweeney-Week-3-2020-9-22-1: Chapter 5 CS Graphics Book

The fifth chapter of Hughes’ book on Computer Graphics talks directly about the subject of human perception. If humans were unable to see computer graphics would be rendered completely useless which is why the subjects have always been so closely tied together. The end goal for computer graphics in terms of advancements is to get to the point where graphics are better than anything the human eye can perceive because once that point is reached there is no need to continue advancements until humans evolve and increase the strength of their sight. Hughes talks about at a basic level computer graphics is a simply a method of transmitting light to our eyes in a certain “pattern”. This pattern is how the human eye perceives the things they see on a screen and based on this the visual cortex is able to deduct what it sees to the rest of the human brain. The rest of the chapter continues to go into more detail about exactly how the intricacies of the human eye work and exactly why computer graphics end up being such a complex thing to develop even at the base level.

Nicholas-Sweeney-Week-3-2020-9-22-2: Chapter 6 CS Graphics Book

Chapter 6 of Computer Graphics: Principles and Practice begins to move into the world of 3D graphics. The book continues to work on using the features available in the WPF application and makes direct reference to the different things you can use in a 3D scene in order to amplify your graphics project. The application coordinate system that can be used in a 2D scene for WPF is relatively the same but with a 3rd z-axis added on the represent the change in dimensions. This is called the world coordinate system and the application user is able to directly determine what unit of measurement they would like to use for their program whether real or imaginary. From here Hughes goes into detail on how someone should go about setting up their scene and touches upon meshes and lighting. Test code for the application is then provided so the reader is able to go into the program and see exactly what is shown in the pictures found by the code in the book. Each section of this chapter goes into depth about a different feature of WPF and overall gives the user a very large variety of knowledge about various things they will have to take into account while working on computer graphics.

Nicholas-Sweeney-Week-4-2020-9-29-1: Chapter 7 CS Graphics Book

Hughes’ 7th chapter of his book on computer graphics talks directly about the different mathematics and geometry practices that a computer graphics developer needs to know in order to properly code any sort of 2D or 3D scene. Each section of this chapter goes over a different function that is used for calculating objects within the Cartesian plane that the scene is created on. The first couple sections touch upon notation used in these equations referencing the variable used for sets of real numbers known as R as well as the variable C which is used to reference a set of complex numbers. Once the baseline for these equations is set up for the reader Hughes goes on to start talking about function notation which is used for a programmer to define the curve of a line that they will be placing on their scene. This is used by using one or more of the variables referenced at the beginning of the chapter combined with the transition of the equation across the line, one example of this would be a function of a set of real numbers going towards a set of real positive numbers via the function conversion from just x to x^2.

Nicholas-Sweeney-Week-4-2020-9-29-2: Chapter 8 CS Graphics Book

Computer Graphics: Knowledge and Practice continues on with its 8th chapter by diving into simple ways of representing 2D as well as 3D shapes with a scene in computer graphics. Many basic shapes can simply be defined using two tables: one which references the vertices of the shape in question and one which references the edges of the same shape. The vertex tables directly define where the points of the shape will be drawn, these won’t necessarily define the lines you see but set up the structure for the edges table. The second table is where the magic happens when it comes to actually drawing a shape on the screen. This table has different edges which are basically just stating what two vertices need a line between them. In the book the example of this shows vertices 1 through 6 being defined in the first table and then the first point of the edge table simply shows (1, 2). What this ends meaning is that the first two points on the vertices table then have a line drawn directly between them. More complicated tables can use functions defined in the previous chapter in order to implement curved lines because without this shapes such as semi-circles and other various shapes that may or may not end up being fully closed because there isn’t any rule in computer graphics that states that the shape has to be closed because this allows the programmer more creative power when developing their scene.

Nicholas-Sweeney-Week-5-2020-10-6-1: Chapter 9 CS Graphics Book

Hughes book Computer Graphics: Knowledge and Practice goes over a variety of in depth topics involving different aspects of computer graphics and chapter 9 specifically dives into the complexity of functions being used on meshes. In the previous two chapters Hughes talks about functions in general as well as drawing lines and how you can use functions to get lines that end up using these functions, creating curves. From here the aspect of functions is interwoven with meshes which is where things truly start to get hard to follow. Towards the beginning of the chapter a variety of new terms are introduced to the reader in order to give them a better understanding of the mathematics taking place in this chapter one of these main terms is called a simplex. A Simplex can be either a vertice, edge, or a face defined as either a 0-simplex, 1-simplex, or a 2-simplex respectively. These terms allow functions to directly reference these features on a scene in a similar format to how functions were placed into tables in order to produce curved lines in the previous chapter. A point is then selected across a list of affected simplices and then a function is then used to determine the overall curve or “bumps” on the given distribution between the selected point and every simplex within this list in order to convert the mesh from a flat plane to something more 3D within the scene.

Nicholas-Sweeney-Week-5-2020-10-6-2: Chapter 10 CS Graphics Book

When it comes to computer graphics performing even a simple operation such as moving a shape across the screen in the appropriate direction because a much more complex task than one would imagine it to be. The 10th chapter of Hughes book directly touches upon this subject with transformations in a 2D scene. As referenced in previous chapters functions will still be used to influence an object in the scene’s position but for the 2D images it will mainly be used to influence different coordinates within an object's matrix. A function is combined with an objects matrix which lists all of its individual shape point’s coordinates producing a matrix with the overall change for each of these coordinates. Based on the function provided this can have a variety of effects on the shape but typically, as seen in the examples provided on page 222 and 223 either end up moving the shape, changing it’s overall size, or both. This combined with a loop in your scripting language of choice provides the programmer to create moving objects across the screen and is the entry point for developing complex scenes with moving parts in any computer graphics programming application, such as WPF.

Nicholas-Sweeney-Week-6-2020-10-13-1: Chapter 11 CS Graphics Book

Once the reader is able to learn about transformations in a 2D scene Hughes goes on to explain the obvious next step in teaching someone the basics of computer graphics and that is transformations on a 3D scene. The beginning of the chapter shows off a lot of the similarities that can be found between transformations of both the 2D and 3D variants. Both types use matrices as a way of listing off the coordinates of each individual point of the object in question; the 3D matrix just so happens to have an extra column so the user is able to influence that 3rd dimension as well as the x and y coordinates. From here different transformations are listed and the formulas to perform each is discussed. One of these formulas is rotations which mainly use functions such as cosine, sine, and tangent in order to rotate each individual point on the object. The different movements of each axis is depicted with a picture of a plane and it gives the reader a general idea of what will happen to their object based on whatever axis and function they end up using. If the user wants to roll their object all they have to do is influence their x coordinates via their function of choice and then set it up to end up rolling to the exact angle that they end up wanting the object at.

Nicholas-Sweeney-Week-6-2020-10-13-2: Chapter 12 CS Graphics Book

Computer Graphics: Knowledge and Practice goes about describing different basic aspects of different computer graphics throughout each chapter and then follows it up with a direct example that can be found within a computer graphics application. The main application discussed throughout the book is WPF and in chapter 12 Hughes goes through a specific library you can import in WPF to use for when you need to make 2D and 3D transformations. The chapter introduces predefined classes within the library that are used for points and vectors and then moves into the different transformation classes the chapter will be discussing: MatrixTransformation1 & 2, LinearTransformation2, AfflineTransformation2, and ProjectiveTransformation2. LinearTransformation2 is the class that allows the user to determine the changes they want to implement on any given vector class. Once the 2D transformation classes are described Hughes follows it up by saying that any of the 3D transformation classes are very similar as well and after making it through the past few chapters the reader has a decent grasp of the transition from 2D to 3D transformations so converting this into coding knowledge ends up being relatively easy thanks to the formatting of the book up to this point.

Nicholas-Sweeney-Week-7-2020-10-20-1: Chapter 13 CS Graphics Book

One of the main things that makes or breaks how your scene is set up and how it looks is the camera. The overall positioning of your camera and the way it’s set-up in terms of its view completely control how the user interprets the scene you have created. Without the right positioning you could program something that makes total sense but since your camera setup doesnt work well with the scene the user will have a very hard time interpreting the graphics leading to a subpar user experience which for most applications is one of the top priorities that the programmer needs to worry about. Hughes makes sure to touch upon transformations in this chapter as well because they are also a major factor when it comes to the camera positioning because these transformations can lead to an image moving on or off screen in a way that the user wasn’t expecting because their camera wasn’t moving along with the object that was being transformed on the screen. 3D scenes are especially difficult for this aspect because not only does the user have to worry about the object in question moving off screen on the x and y planes but if they have a background on their scene then the object can actually be transformed and end up going behind this background on the z axis.

Nicholas-Sweeney-Week-7-2020-10-20-2: Chapter 14 CS Graphics Book

The end goal of computer graphics in terms of overall improvements has always been getting to the point where they are better than anything a human eye can ever perceive. One approach for this is to get as close to real life in terms of overall graphics quality. Obviously this approach ends up being less than stellar as there are numerous non-graphics related things regarding the physics of the scene that play an influence on exactly how every object moves throughout the scene. Hughes starts chapter 14 off by listing a variety of factors that a programmer in computer graphics should analyze in order to judge the overall efficiency of the scene that they are developing such as physical accuracy, perceived accuracy, design goals, space efficiency, time efficiency, implementation complexity, and associated cost of content creation. The one that many developers end up trying to make the most accurate is the physical accuracy because if the scene doesn’t look like something that makes sense to them in relation to what they experience on a day to day basis in real life then they typically will see that scene as more clunky and less polished than many of the other applications that they have seen in the past.

Nicholas-Sweeney-Week-8-2020-10-27-1: Chapter 15 CS Graphics Book

Obviously the programming aspect of computer graphics in terms of how the applications functions is huge but there is also the detail of how the colors of each pixel are determined that still needs to be discussed. In previous chapters Hughes goes over exactly how camera angles can truly change how your scene is viewed by the user which can end up making or breaking your game. At a more primitive level than this is how the color of each pixel on the screen is displayed. The 15th chapter goes into detail about two of the major aspects of this process which are ray casting and rasterization. Ray casting allows the user to create a ray for each individual picture on the screen and from this ray is then projected across every surface within the screen. With this method the user is able to process parallel pixels because of the way light works to increase our overall application performance. The negative to this function is that we also have to keep the entire scene in memory which, for some larger more high resolution applications, can end up taking up a large chunk of memory. Due to this underlying necessity for storing the entire scene in memory many programmers aim to reduce the amount of “entities” or moving objects in their scene to decrease the amount of times the scene needs to be resaved because of movement within the scene altering its save state.

Nicholas-Sweeney-Week-8-2020-10-27-2: Chapter 16 CS Graphics Book

Hughes 16th chapter of Computer Graphics: Knowledge & Practice taps into the information around real-time 3D graphics platforms. One main subject is the overall improvements in graphics from the mid 1900s onwards. Computer graphics initially started out using fixed-function pipelines. These pipelines at first only allowed the user to edit the parameters provided in a given function or algorithm and didn’t give the user any access to customize that actual algorithm being used within their program. The next step for graphics was the development of shaders which gave the programmer the ability to not only define the colors of the surfaces they had within a scene but also access to a variety of different aspects in the rendering pipeline that previously were completely unavailable. Shaders became easily accessible in the early 2000s when high-level languages began to include access to this shader functionality which was much easier to learn via these languages in comparison to an assembly language which is much less customizable. From this point forward the baseline for computer graphics programming finally moved away from the fixed-function pipeline to something more complex known as the hybrid pipeline which made extensive use of shaders as they continued to become more complex and customizable over the years to come.

Nicholas-Sweeney-Week-9-2020-11-3-1: Chapter 17 CS Graphics Book

One main issue when it comes to taking pictures is that there are a variety of different aspects that a camera captures that aren’t necessarily carried over when those pictures are uploaded to the computer. Hughes discusses this in Computer Graphics: Knowledge & Practice and highlights the different values that are stored whenever a camera takes a photo and how those values end up being transferred over onto a computer. Most people with computer graphics experience are very familiar with the fact that pixels are typically defined by their red, green, and blue values also known as their RGB values. Hughes explains that with many digital cameras there are actually a few other values that are defined for the pictures taken by these cameras that help give the pictures a more realistic feeling. The two extra values that most cameras have nowadays for each pixel are a depth value which is determined based on how far away from the virtual camera the pixel is as well as the alpha value which basically determines the transparency of the pixel in question. This 5 value format can end up leading to images that start taking up a significant amount of space when uploaded to a computer in bulk and one solution discussed in the chapter is lossy compression. To summarize this method removes every other pixel from a picture’s file and when opened it takes an average of all of the values stored on the adjacent pixels in order to generate the color for each of the missing pixels.

Nicholas-Sweeney-Week-9-2020-11-3-2: Chapter 18 CS Graphics Book

Another huge question that comes from porting pictures in from digital cameras onto computers is what happens when various operations are performed on these images in a 2D or 3D scene using computer graphics. Hughes describes it to be very similar for most of the functions you would use to alter regular shapes because you are basically already doing these transformations when you perform them on a shape that has a mesh on it that changes the colors projected onto it. Using ray tracing transformations can be taken as a function defined on the real line with one ray per pixel that is then traced at the center of that given pixel. By doing this for each individual pixel you can very easily move the entire picture from one position to another given the reader an easy explanation of how the can go from moving a simple square across the screen to moving something more complex like a picture they just took of a sunset in the full resolution that it was originally taken in.

Nicholas-Sweeney-Week-10-2020-11-10-1: Chapter 19 CS Graphics Book

With the transformations of images comes another major question for many computer graphics developers which is what happens when an image’s size is altered? This becomes a relatively large issue when up or down-scaling an image because you can end up losing a large amount of the initial quality the base image had by going in either direction if your functions for resizing aren’t properly designed. One obvious solution that Hughes states is actually very suboptimal is removing alternate pixels when down-sizing an image as he confirms that by doing this you can end up having very bad outcomes for the resulting image if you go down this route. The solution to this is to essentially set up all of the pixels as integer values in an array that then either shrinks or stretches based on the desired size the user wants. If the image is 300 x 300 pixels for example, Hughes states that the pixel values then become integers that are put into an array from 0 to 299. These values are then put into a function that converts them into an array that is the exact amount of pixels either greater or lower based on what the user is aiming to do. If the user is looking to make the image 100 pixels bigger on either axis then the user simply needs a function that takes the original array values and multiplies them by 0.75 to get the new values for the larger image. The same method can be used as well for shrinking an image.

Nicholas-Sweeney-Week-10-2020-11-10-2: Chapter 20 CS Graphics Book

Hughes makes a good segway from the previous chapter by going straight from manipulating images to mapping textures. Mapping textures is similar to the last chapter but takes a slightly different approach. Textures are basically an image that a programmer would use on a shape to give it a different appearance. One example of this would be when you make a house in a 3D scene it would end up being like a blank canvas and then you would map a texture to it with details so the house then had all the regular features a real house would like windows, doors, siding, and roofing. The book uses a slightly more basic example by describing how to map a soda can label to a 3D cylinder. The formula used to map out each polygon of the cylinder takes into account a variety of different factors depending on the shape. In the case of mapping to a cylinder the formula has variables r for radius, h for height, as well as an i and j variable for determining what part of the cylinder the user will be mapping. Some more intricate shapes use much more complex formulas due to their complex state but overall these formulas are the direct route for a reader to learn how to draw a simple shape into creating something that they would see in many triple AAA projects today.

Nicholas-Sweeney-Week-11-2020-11-17-1: Chapter 21 CS Graphics Book

The next chapter written by Hughes breaks down a variety of different interaction techniques that computer graphics has a heavy presence in, one of which being the GUI which is a necessity for almost every application regardless of its use. GUI stands for graphical user interface that relates to just about any menu or any interactable object that a user sees on the screen. Without the knowledge of how to develop a GUI a programmer’s ability to successfully create any application in computer graphics is basically non-existent. Everything from the way all of the features on the screen look and are organized to how each of these features effects the program when interacted with by the user require manually programming through computer graphics. Back in the early days the first modern GUI was developed on a CAD-like system that used a pen as well as physical buttons for the majority of the input which compared to modern systems isn’t that far off. On top of all of the interactable features a user can interact with on the screen through the GUI the user can also use any keys on their keyboard for input as well in some cases and this would be somewhat similar to what the original system known as the Sut63.

Nicholas-Sweeney-Week-11-2020-11-17-2: Chapter 22 CS Graphics Book

Animation directly relates to the movement of objects throughout a scene and in the 22nd chapter of Computer Graphics: Knowledge and Practice, some of the methods used for animation, specifically those that are used in animations involving geometric modeling, are discussed in this chapter. The example Hughes uses to simplify the concepts for the reader is explaining how to animate a car going from some point A to some point B on a curve. The formula at its most basic state takes point A and adds a vector which then adds up to be point B. The most important part of animation though is making this transition as realistic as possible and to do this you need to use the computer graphics formulas known as the Hermite functions. These functions take four variables, three of which are defined above, with the last one being a second vector to influence movement on a second axis. From here the variables are all placed into the formula listed on page 596 of Hughes’ book and when any time t is put into the formula the result is the position of the object based on whatever vectors as well as the starting and end points that were provided. The Hermite functions provide a semi-complex way for the reader to fully grasp how an animation formula operates at the most basic level.

Nicholas-Sweeney-Week-12-2020-11-24-1: Chapter 23 CS Graphics Book

The spline curves that are discussed in the previous chapter by Hughes are elaborated on with the introduction of spline surfaces to the reader. These surfaces are provided through the generalization of the curves and from this state they can be further broken down into small pieces known as a Bezier patch. In the previous chapter Bezier curves were defined as having a sequence of four points so it makes sense that as you move into 2D that the patches would have a sequence of 16 points that are used to define them. These 16 points are organized and defined by a variable Pij where i and j both have a range of 1 to 4 with P23 representing the 7th point. Any of the points that have i equal to 1 are defined as control points and the location of these four points determine the shape of the Bezier patch. The formula to determine the output of the surface takes into account the 16 points discussed previously as well as a combination of just the four control points in order to calculate the exact patch within the 3D space that the programmer originally set out to create.

Nicholas-Sweeney-Week-12-2020-11-24-2: Chapter 24 CS Graphics Book

As the reader is probably well aware at this point, most of the development of shapes, curves, and animations in computer graphics typically requires complex formulas in order for the programmer to have as much control over the output as possible. Hughes’ next chapter of Computer Graphics: Knowledge & Practice focuses on implicit representations of shapes in order to give the user a simplified formula to quickly make very basic shapes. The benefits of having access to create shapes like this is the fact that these shapes are generally as smooth and well-made as a user could possibly make a shape. A lot of the development of shapes can never be made “perfect” without extensive programming experience in computer graphics so this access gives many entry-level programmers a much easier way to get something in there scene to begin actively working on animations and other sorts of manipulations to increase their overall graphics knowledge at a more rapid pace. The main disadvantage that steers more experienced programmers away from using these shapes is that converting them to a representation that is polygon mesh compatible requires a large amount of resources from most renderers that are currently on the market which is why the author opts to not mention the capability to develop these implicit representations until after the user has at least read about how to develop shapes using polygon meshes.