Pergamon

Comput. & Graphics, Vol. 22, No. 5, pp. 621–628, 1998

PII: S0097-8493(98)00069-7
Printed in Great Britain
0097-8493/98/$ - see front matter

Technical Section

EFFICIENT INTEGER ALGORITHMS FOR THE GENERATION OF CONIC SECTIONS

A. AGATHOS, T. THEOHARIS† and A. BOEHM*

Department of Informatics, University of Athens, Panepistimioupolis, TYPA Buildings, 157 71 Athens, Greece

Abstract—Efficient integer 8-connected algorithms for the fast generation of Conic Sections whose axes are aligned to the coordinate axes are described based on a Bresenham-like methodology. Performance results show that in the case of the ellipse, the algorithm is at least as fast as other known integer algorithms but requires lower integer range and always performs correct region transitions. Antialiasing is easily incorporated. © 1998 Elsevier Science Ltd. All rights reserved

1. INTRODUCTION

Conic sections (ellipse, hyperbola and parabola) are important geometric primitives and, after the straight line and circle, have received much attention from the computer graphics community. A number of algorithms have appeared in the literature for the generation of these primitives [1–7].

We have derived efficient *integer* 8-connected algorithms for the generation of ellipses hyperbolas and parabolas whose axes are aligned with the axes of the plane, using a Bresenham-like methodology [2] simulating, in effect, the midpoint technique [1]. Our algorithms use integer arithmetic in a straightforward manner without any scaling and do not lack in performance with regard to any previous algorithm. Spacewise, they require a small constant number of integer variables. They are also *symmetric* as to the number of arithmetic operations per pixel generated in each octant. Thus they are highly suitable for teaching. *Erroneous pixels* are *not* generated at region boundaries due to a better region transition criterion. In the case of the ellipse our algorithm requires *lower integer ravage* than Kappel's integer ellipse drawing algorithm. Our algorithms are very suitable for hardware implementation especially in view of increasing display resolutions and the availability of high-resolution plotters. Our parabola algorithm in particular, is suitable for very high resolutions due to the elimination of the calculation of a square factor. We are also able to exploit the value of the decision variable for antialiasing.

The rest of this paper is organised as follows: Section 2 presents a modified Bresenham circle algorithm. Section 3 describes the derivation of the ellipse generating algorithm. Section 4 briefly describes the parabola and hyperbola algorithm derivations. Appendix A, Appendix B and Appendix C give the ellipse, hyperbola and parabola Pascal procedures.

†Corresponding author. Tel.: +0030-1-7275106; Fax: +0030-1-7231569; E-mail: theotheo@di.uoa.gr.

*We would like to dedicate this paper to the memory of our dear friend Alexandros Boehm who died on May 1st 1998.

2. REFORMATTING THE BRESENHAM CIRCLE ALGORITHM

We develop a small variation to Bresenham's circle generating algorithm which concerns the criterion for next pixel selection and octant change detection. We shall later use this small change in a generalisation of the algorithm to the more complex conic sections; the integer algorithms derived in this way are correct and efficient.

Consider the second octant of a circle of integer Radius R (Fig. 1). As in Bresenham's algorithm, having chosen pixel A, we define:

$$d1 = R1^2 - R^2 = (y_i^2 + (x_i + 1)^2) - (y^2 + (x_i + 1)^2)$$

$$= y_i^2 - y^2,$$

$$d2 = R^2 - R2^2$$

$$= (y^2 + (x_i + 1)^2) - ((y_i - 1)^2 + (x_i + 1)^2)$$

$$= y^2 - (y_i - 1)^2$$

in a manner similar to Fellner [8]. We then take:

$$d = d1 - d2 = (y_i^2 - y^2) - (y^2 - (y_i - 1)^2),$$

where d is the decision variable for the selection of the next pixel between the two candidates B and D.

At this point we take a diversion from Bresenham's algorithm by setting $\varepsilon = y_i - y$ to get:

$$d(\varepsilon) = -2\varepsilon^2 + 4y_i\varepsilon + 1 - 2y_i. \qquad (1)$$

13. Kumar, Vinay, Muttoo, S.K.,Bansal Abhishek 'Hiding Information in Vector Layer while Extracting Layer from Raster Data 'Information Security Journal: A Global Prespective' Vol. 21, Issue 6, 2012.

14. Kumar, V. and Muttoo, S.K.,, 'A Graph Theoretic Approach to Sustainable Steganography', Int. J. of Tiawan , Vol.17, No.1. 2011.

15. Sushil Kumar and S.K. Muttoo " Steganography based on Counterlet Transform", Bharti Vidyapeeth International Journal of Information Technology. Vol. 9 No. 6, June 2011.

16. S.K. Muttoo and Sushil Kumar,"A multilayered secure, robust and high capacity image steganographic algorithm" World Computer science and IT Jr. Vol. No.,pp 2011.

17. Muttoo, S.K. and Kumar, V., 'Hamiltonian Graph Approach to Steganography'. International Journal of Electronic Security and Digital Forensic, Inderscience, Vol.3, No.4, pp 311–332 2010

18. Muttoo, S. K. and Kumar, V., 'Hiding Message in Map along pre Hamiltonian Path'. Int.J. of information Security and Privacy, Idea Group USA, Vol.4, No.4,pp.21-34. (2010).

19. Kumar, V. and Muttoo, S.K.,(2010), 'Graph Theoretic Approach to Steganography to Secure Message Digest'. Information Security Journal: A global perspective, Taylor & Francis, Vol.19, No.6, pp.328-335.

Budget

Research Grant Required: Rs. 1,50,000 under the following items

1.	Minor Equipment:(Android based equipment)	30,000
2.	Desktop Computer :	70,000
3.	Software	30,000
4.	Consumables	20,000

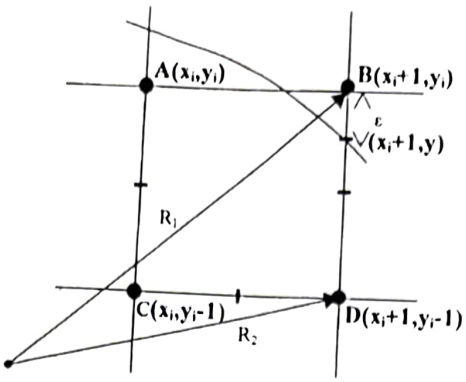


Fig. 1. Candidate Pixels B or D

The above expression is monotonically increasing in the interval $\varepsilon \in (-\infty, y_i]$. We can therefore use the value of $d(\varepsilon)$ at $\varepsilon = 1/2$ i.e. $d(1/2) = 1/2$ as the decision value:

if $d \leq 1/2$ then pixel B is chosen
else pixel D is chosen.

Note that since d is the integer, we can replace the 1/2 by 0, without affecting the semantics. Note that what we really accomplish here is to simulate a midpoint-type technique [1].

Due to the 8-way symmetry of the circle, we need not consider another octant; in the case of the ellipse, which has 4-way symmetry, we need to consider 2 regions which make up one-quarter of the ellipse.

3. DERIVATION OF THE ELLIPSE GENERATING ALGORITHM

Consider an ellipse centered at the origin of the 2D cartesian space defined by:

$$x^2/a^2 + y^2/b^2 = 1$$

The ellipse has a 4-way symmetry and it is therefore only necessary to generate its arc in the first quadrant. Here we distinguish two regions separated by the point on the ellipse where $dy/dx = -1$. In the first region the axis of major movement is X and in the second Y (Fig. 2).

We must derive an incremental expression for the decision variables in Regions 1 and 2, the initial values of the decision variables and a condition to detect the transition from Region 1 to Region 2.

3.1. *Decision variable for Region 1*

Let us begin by considering the 1st region of Fig. 2, where the X-axis is the major axis of movement. Assume that the ellipse is generated in a clockwise manner starting from the point $(0,b)$. At each step in the generation the X value is therefore always incremented. It must be determined whether the Y value should be decremented or not. This region corresponds to the 2nd octant of the circle (Fig. 1) and we define:

$$d1 = y_i^2 - y^2$$

$$d2 = y^2 - (y_i - 1)^2,$$

as in the case of the circle. Taking as decision variable:

$$d = a^2(d1 - d2),$$

the following will hold (Fig. 1):

if $d \leq a^2/2$ then pixel $B(x_i + 1, y_i)$ is chosen
else pixel $D(x_i + 1, y_i - 1)$ is chosen. (2)

The only reason for multiplying by a^2 is to facilitate its incremental derivation (see below). Of course the division of the integer value a^2 by 2 can be replaced by one Right Shift of a^2; since d is an integer the result is semantically equivalent.

We next derive the incremental computation of the decision variable; its value for the ith step of the algorithm in Region 1 is:

$$d_{1,i} = a^2(d1 - d2)$$

$$= a^2 y_i^2 + a^2(y_i - 1)^2 - 2a^2 y^2$$

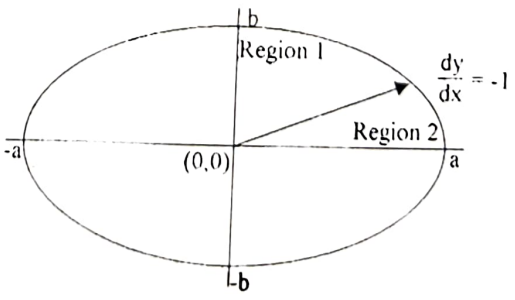


Fig. 2. Ellipse

LIST OF RESEARCH PAPERS(2010-2015)

1. M. P. S. Bhatia, Sunil Kumar Muttoo, Manjot Bhatia "An Image Steganography Method Using Spread Spectrum Technique" Proceedings of Fourth International Conference on Soft Computing for Problem Solving, December, 2014.

2. Sunil Kumar Muttoo, Kumar Vinay. Bansal Abhishek" Steganography using solution of 8 queen's problem" International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.7, No.5 , pp.47-58, 2014.

3. Bhavya Ahuja, S.K. Muttoo, Deepika Aggarwal,"Analysis of effect of varying quantization level on the image communication scheme based on combination of compression, cryptography and steganography", International Journal of Enhanced Research in Management & Computer Applications,Vol. 3 Issue 2, pp: (30-36), 2014.

4. S K Muttoo , Sushil Kumar " Self-synchronising image steganography algorithms based on error correcting codes" I Jr Electronic Security and Digital Forensics,Vol 5, Nos 3 / 4 ,2013.

5. MK Bhatia, SK Muttoo, MPS Bhatia "Secure Requirement Prioritized Grid Scheduling Model" International Journal of Network Security 15 (6), 478-483 2013.

6. S Kumar, SK Muttoo "Image Steganogaraphy Based on Complex Double Dual Tree Wavelet Transform." International Journal of Computer & Electrical Engineering 5 (2) 2013.

7. S Kumar, SK Muttoo " A comparative study of image steganography in wavelet domain International" Journal of Computer Science and Mobile Computing, IJCSMC Issue 2 2013.

8. M Bhatia, SK Muttoo, MPS Bhatia "Secure Group Communication with Hidden Group Key" Information Security Journal: A Global Perspective 22 (1), 21-34 2013.

9. SK Muttoo" Data Security " Mass Communicator: International Journal of Communication Studies 7 (1), 39-40 2013.

10. S Kumar, SK Muttoo "Image Steganography Based on Wavelet Families "International Journal of Electronic Security and Digital Forensics 5 2013.

11. V Gaur, A Soni, SK Muttoo, N Jain "Comparative analysis of Mamdani and Sugeno inference systems for evaluating inter- agent dependency requirements" J Comput Eng Inf Technol 2, 2013.

12. SKMuttoo Abdul Satar A NEW STEGO-SYSTEM BASED ON LFSR GENERATOR International Journal of Engineering and Innovative Technology (IJEIT), Vol. 1 Issue 1. 2012.

Given that $a^2y^2 = a^2b^2 - b^2(x_i + 1)^2$ [equation of ellipse],

$$d_{1,i} = -2a^2b^2 + 2b^2(x_i + 1)^2 + a^2y_i^2 + a^2(y_i - 1)^2. \quad (3)$$

We shall now define $d_{1,i+1}$ in terms of $d_{1,i}$:

$$d_{1,i+1} = -2a^2b^2 + 2b^2(x_{i+1} + 1)^2 + a^2y_{i+1}^2 + a^2(y_{i+1} - 1)^2$$

$$= -2a^2b^2 + 2b^2((x_i + 1) + 1)^2 + a^2y_{i+1}^2 + a^2(y_{i+1} - 1)^2\{(x_{i+1} - x_i) + 1\}$$

$$= -2a^2b^2 + 2b^2(x_i + 1)^2 + 2b^2 + 4b^2(x_i + 1) + a^2y_{i+1}^2 + a^2(y_{i+1} - 1)^2 \quad (4)$$

But $-2a^2b^2 + 2b^2(x_i + 1)^2 = d_{1,i} - a^2y_i^2 - a^2(y_i - 1)^2$, therefore

$$d_{1,i+1} = d_{1,i} + a^2y_{i+1}^2 + a^2(y_{i+1} - 1)^2 - a^2y_i^2 - a^2(y_i - 1)^2 + 2b^2 + 4b^2(x_i + 1)$$

If $d_{1,i} > a^2/2$ then $y_{i+1} = y_i - 1$ by Equation (2), thus

$$d_{1,i+1} = d_{1,i} + 2b^2 + 4b^2(x_i + 1) - 4a^2(y_i - 1).$$

if $d_{1,i} \le a^2/2$ then $y_{i+1} = y_i$ by Equation (2), thus

$$d_{1,i+1} = d_{1,i} + 2b^2 + 4b^2(x_i + 1).$$

The initial value $d_{1,0}$ is determined by substituting the coordinates of the first pixel of Region 1 $(0,b)$ for (x_i, y_i) in the expression for $d_{1,i}$:

$$d_{1,0} = 2b^2 + a^2(1 - 2b)$$

3.2. *Transition from Region 1 to Region 2*

Van Aken [3] proposed a transition criterion based on midpoints which gives correct results but is computationally expensive. Kappel's method [4] is efficient but there exist cases where an erroneous pixel can arise at region boundaries. This is because:

i. Region change is detected by taking the tangent on integer coordinates rather than the true ellipse as acknowledged in Kappel's paper [4] and

ii. A drastic change of curvature can take place within a single pixel.

We propose a transition criterion which combines the advantages of the above two methods (see Fig. 3).

Our integer algorithm uses a correct criterion which is based on the value of the error function in the next column of pixels. In particular, the value of the error function d at the point $(x_i + 1, y_i - 3/2)$ is considered; note that if the ellipse 'passes under' this point then an octant transition is required.

We can express the error function d given in Equation (3) in terms of $\varepsilon (= y_i - y)$ in a manner similar to the circle (1):

$$d(\varepsilon) = -2a^2\varepsilon^2 + 4a^2y_i\varepsilon + a^2 - 2a^2y_i$$

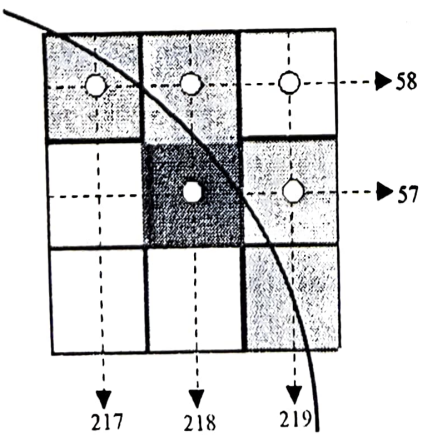
setting $\varepsilon = 3/2$ we get:

$$d(3/2) = -2a^2(3/2)^2 + 4a^2y_i(3/2) + a^2 - 2a^2y_i$$

$$= 4a^2(y_i - 1) + a^2/2.$$

The transition criterion is as follows:

if $d \le 4a^2(y_i - 1) + a^2/2$ then we remain in the same region



a=245 b=126

Already Estimated Grid Point = *(218,58)*

Approximate Slope = -0.9941

True Slope = -1.0027

Next Grid Point :

	Y – Coordinate	= 57
Kappel	X-Coordinate	= 219
Our Algorithm	X-Coordinate	= 218
Van Aken	X-Coordinate	= 218
True	X-Coordinate	= 218.4971

Fig. 3. Transition from Region 1 to Region 2

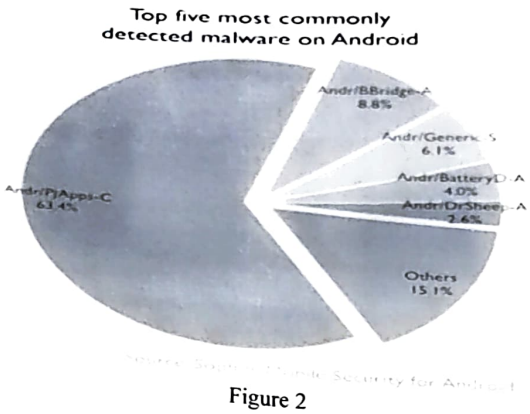


Figure 2

Advertisements on the device.

- **Andr/Generic-S** The "generic" category included apps that use privilege escalation exploits and aggressive adware (such as Android Plankton).
- **Andr/DrSheep-A** Dr.Sheep is the Android equivalent to Firesheep, the Firefox plug-in that allows people to hijack Twitter, Facebook and Linkedin sessions in a wireless network environment.

REFERENCES

1. Statista, http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/
2. Readwrite , "http://readwrite.com/2013/10/28/android-kitkat-security" \l "awesm=~onh8fragpNriAO"
3. Security watch,http://securitywatch.pcmag.com/android/299152-top-five-android-malware-types
4. Developer, http://developer.android.com/reference/java/lang/ThreadGroup.html
5. ANDROID SECURITY-Attacks And Defenses- Abhishek Dubey, Anmol Mishra

PROPOSAL:

As Android emerges as the leading platform for mobile devices, security issues associated with the Android platform becomes a growing concern for personal and enterprise customers. With the proliferation of smart phone users, Android malware variants is increasing in terms of numbers and amount of new victim android apps. This research will focus on creating android malware by introducing hindrance in the normal working of the android applications and detecting the malwares in the android applications.

else we change region.

As we have already mentioned above the division $a^2/2$ can be replaced by a semantically equivalent Right Shift. The above criterion is optimised and used in the code fragment in Appendix A. We have to note that the above criterion works correctly for $y_i \geq 1$. Square corners, as mentioned by McIlroy [6], can be predicted easily in our algorithm using one copy of the pixel last printed in the first region. Other degenerate cases, such as McIlroy's long thin ellipses, have been tested. It is heuristically believed that no degenerate cases will cause the algorithm to fail.

The initial value of the decision variable $d_{2,i}$ for Region 2 (see Equation (5) below) can be calculated by adding to the final value of $d_{1,i}$ the difference $d_{2,i} - d_{1,i}$. From Equation (4) and Equation (5):

$$d_{2,i} = d_{1,i} - a^2(2y_i - 1) - b^2(2x_i + 1).$$

$$d_{2,i} = -2a^2b^2 + b^2x_i^2 + 2a^2(y_i - 1)^2 + b^2(x_i + 1)^2 \quad (5)$$

An incremental expression for $d_{2,i}$ can be derived in a similar manner to $d_{1,i}$ to be:

$$d_{2,i+1} = d_{2,i} + b^2x_{i+1}^2 + b^2(x_{i+1} + 1)^2 - b^2x_i^2 - b^2(x_i + 1)^2 + 2a^2 - 4a^2(y_i - 1)$$

which can be simplified, depending on the value of x_{i+1}, as follows:

if $d_{2,i} > b^2/2$ then $x_{i+1} = x_i$, thus

$$d_{2,i+1} = d_{2,i} + 2a^2 - 4a^2(y_i - 1),$$

if $d_{2,i} < b^2/2$ then $x_{i+1} = x_i + 1$ thus

$$d_{2,i+1} = d_{2,i} + 2a^2 + 4b^2(x_i + 1) - 4a^2(y_i - 1).$$

3.3. *Decision variable for Region 2*

In Region 2 the expressions for $d1$ and $d2$, as can be seen from Fig. 4, are:

$$d_1 = (x_i + 1)^2 - x^2$$

$$d_2 = x^2 - x_i^2$$

Having chosen pixel $A(x_i, y_i)$, the difference $d = d1 - d2$ determines which of the 2 pixels in the next row of pixels ($y = y_i - 1$) is closer to the real ellipse:

if $d \leq b^2/2$ then pixel $D(x_i + 1, y_i - 1)$ is selected

else pixel $C(x_i, y_i - 1)$ is selected.

The decision variable (scaling again by b^2) for Region 2 step i is defined to be:

$$d_{2,i} = b^2(d1 - d2) = b^2(x_i + 1)^2 + b^2x_i^2 - 2b^2x^2 \quad (4)$$

Given that $b^2x^2 = a^2b^2 - a^2(y_i - 1)^2$ [equation of ellipse],

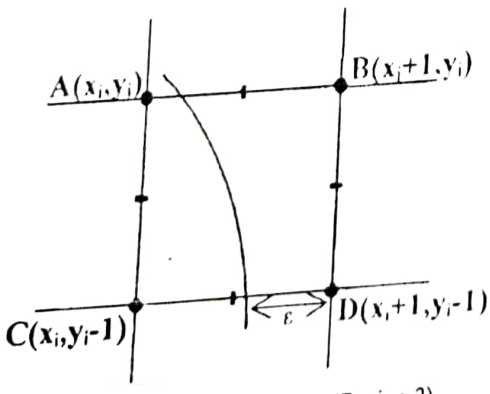


Fig. 4. Ellipse Construction (Region 2)

3.4. *Antialiasing and results*

In the past a linear antialiasing function for conic sections has been proposed [5]. A similar function could be applied to our algorithm (specifically the function $(d(\varepsilon) - d(0))/(d(1) - d(0))$), but unfortunately such linear approximation only works correctly for very few bits of colour ([9], page 971).

We have incorporated a very fast version of the box-filtering antialising technique [10] in the ellipse drawing algorithm, achieving satisfactory results, see Fig. 5.

Since it is computationally expensive to compute the reverse of $d(\varepsilon)$ function, we perform a binary search of the given value of d in the space of values $d(t/n, y_i)$, $t = 0 \ldots n$ where n is the number of grey levels available to determine the required grey level (in the second region we would use $d(t/n, x_i)$). The

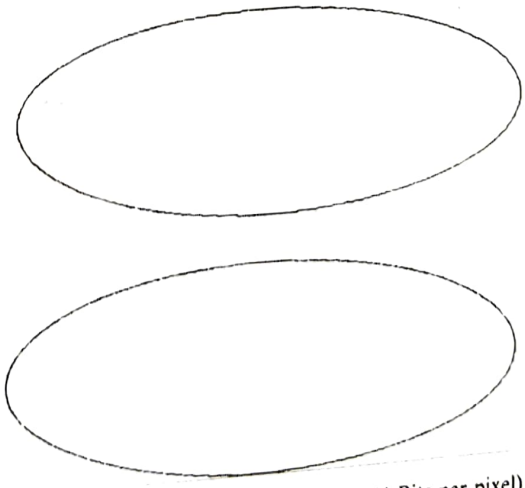


Fig. 5. Midpoint and Antialiased ellipse (4 Bits per pixel)

that must be addressed when a device is rooted because of the fact that malicious applications gain more control they are able to take over a rooted device.

Applications

The Google Play store is an application that comes preinstalled on Android in general. Users use this application to search for and download other apps onto their device. Developers publish their completed applications to this store where they become available for the public to download them. It is the equivalent of the Apple sore for iOS. There are other third party ways to download them. It is equivalent of the Apple store for iOS. There are other third party ways to download applications as well that are more widely used in other countries. In order to have a better chance of downloading a trusted applications, users should generally stick the Google Play Store although it is not guaranteed that there are no malicious applications on this more trusted platform.

Android users have to be careful when installing new applications, as there are malicious applications that pretend to be useful or to try to trick people into downloading them. By tricking users into granting them permissions, they can do harmful things like steal user information, destroy personal data, and even make calls. This is why it is very important to look over the permissions of a specific application and checking that the developer is trusted source before downloading it. These malicious applications are typically discovered and removed by Google when they are found but they are still a real threat to the uninformed users.

4. Developer's Concerns

This section will cover a wide range of precautions that developers should take while building Android applications[5].

Permissions

Permissions are the rights that a specific application has that allow it to perform certain actions on a device. Examples of these actions include taking pictures, using the GPS, reading contacts, or making phone calls. All applications have their permissions available for users to check; you should always make sure you are only installing an application with permissions that you want to give it access to.

Intents

Intents are a mechanism that allow data to be sent between Android processes. They are essentially messages that can cross system security policy themselves. Intents can be sent to the other Android application components like Activities, Services and Broadcast Receivers. It is possible to set up IntentFilters to pick certain types of Intents and read them with a given priority.

Activities

Activities are single functional components of an application. They can also be used to allow applications to call eac h other, and reuse each others features. Intents are used to send data to an Activity to initialize or set it up. It is impor tant to not put any vital information that would interest an attacker into an Intent that is going to be used to start an activity. This is because a malicious program could register an IntentFilter be used to start an Activity. This is because a malicious program could register an Intent Filter that could pick up on the Intents sent in your application and read the sensitive data inside.

Broadcasts

Broadcasts provide a way for applications and system components to communicate securely. They send messages as intents, and the system handles the details for delivering them correctly. Broadcasts can explicitly define their target receiver, so there is not the same problem as with sending Intents to Activities, although developers still need to make sure this is done. There is a special type of broadcast known as a sticky broadcast that has some different properties. They stay around after they have been sent, and any applications with the BROADCAST_STICKY privilege can read any stickybroadcast that is left lying around. So since they can be read malicious applications that may have permission, there should not be any sensitive information sent in these type of broadcasts. They should typically only be used to inform other processes about system state.

Services

Services are essentially background processes that can

values of the function $d(1/n, y_i)$ can be precomputed. Note that in the case of antialiasing we do not follow the midpoint philosophy as it is always necessary to keep the 2 pixels above and below the true y-intercept for Region 1. The y value of these 2 pixels is decremented when $d \geq d(1,y_i)$ and an octant change occurs when $d \geq d(2,y_i)$.

Our 8-connected algorithm produces equally good antialiased results as 4-connected algorithms [5], but an 8-connected algorithm is advantageous in the case of a single bit per pixel since it provides regions of constant thickness.

The time performance of the new algorithm was compared against the algorithm described by Kappel [4] as well as an integer version of Kappel's algorithm which we derived by suitably scaling by 4 its variables in order to achieve the best possible performance. The integer Kappel algorithm exhibits similar performance to our ellipse algorithm; this should be expected because the integer version of Kappel we derived is very similar in structure to our algorithm. However, the integer Kappel produces arithmetic overflow quicker than ours. It also requires a greater integer range as can be seen in Scheme 1 which compares the two algorithms in terms of the maximum integer value required, as ellipse size increases. The maximum integer arises in the calculation of y_slope in both of the algorithms.

It must be restated here that Kappel's algorithm can give rise to erroneous pixels at the 4 region boundaries as pointed out by Kappel (see Fig. 3). Our integer algorithm does not exhibit this problem.

4. THE HYPERBOLA AND PARABOLA ALGORITHMS

In a similar manner to the ellipse, one can derive incremental error expressions for the construction of our hyperbola and parabola generating algorithms.

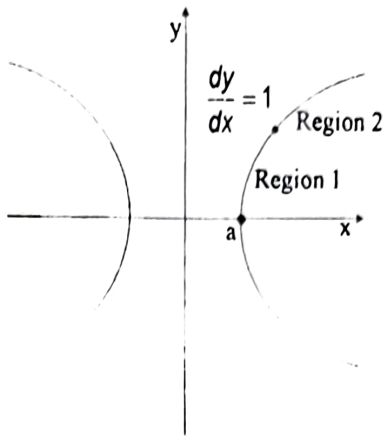


Fig. 6. Hyperbola.

4.1. *Hyperbola*

Figure 6 shows a hyperbola centered at (0,0), symmetric about the X and Y-axes, defined by the equation

$$x^2/a^2 - y^2/b^2 = 1$$

We consider here only the case $a > b$ in which the hyperbola has 2 regions, one in which the major axis of movement is Y (Region 1) and another in which the major axis of movement is X (Region 2). If $a <= b$ there is no Region 2. The two regions are separated by the point where the tangent to the hyperbola has slope $dy/dx = 1$.

In Region 1 (see Fig. 7), the expressions for a measure of the distance of the true hyperbola to the 2 nearest pixels are:

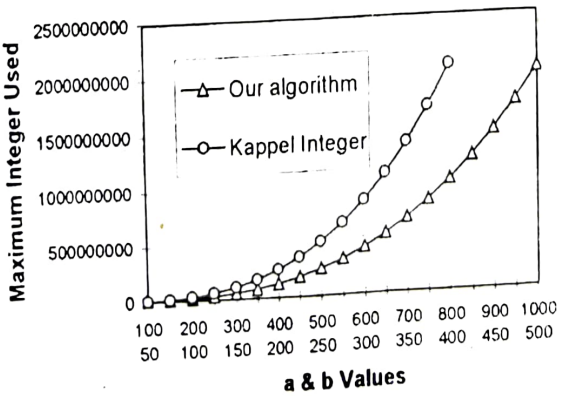
$$d1 = b^2x^2 - b^2x_i^2$$

$$d2 = b^2(x_i + 1)^2 - b^2x^2$$

Setting $\varepsilon = x - x_i$ we get:

$$d(\varepsilon) = d1 - d2$$

$$= 2b^2\varepsilon^2 + 4b^2x_i\varepsilon - b^2 - 2b^2x_i \qquad (5)$$



Scheme 1. Maximum integer graph.

Potentially run for a long period of time. They are used for things like running of a game server of playing mu
Certain applications require a call to a running Service which could be potentially insecure. Developers mu
validate the Service they are connecting to and not an unknown program, especially if they are sending it
information such as passwords or emails.

SQL Injection SQL injection is a fairly common exploit that is present in all forms of application with the database calls. For the android platform, SQL injection can be easily avoided through the use of parameterized queries which explicitly distinguish between the data being sent and the query logic. One caveat would be
that if string concatenation is being used in the application
(and then these strings are passed along), there could still be the possibility of SQL injection if the data is not being passed directly into a parameterized query.

File Permissions
The Android file system is very similar to any Linux computer and has similar permissions. Developers should take similar precautions as with any other sort of programs and only create files and grant permissions for these files as they intend. Make sure that the permissions are explicitly defined: i.e. log files, and temporary files are fine for writing, but they should probably not be given global read permissions.

B. Android Malwares

The woes continue for Android users when it comes to security. Read write[2] that apps listed on the Play Store are more vulnerable than their counterparts on the App Store, and because of the lacking review policy in the former, anyone who installs an app from Google Play is at a greater risk of malware device infections.
And in the third quarter, the malware that makes the Android system vulnerable accounted for 97% of the threats, with the rest being accounted for by the Symbian OS. As a comparison, there was no malware associated with Windows Phone, iOS and Blackberry OS.

1. Top Five Android Malware Types
While there are plenty of apps eavesdropping on SMS messages and transmitting sensitive data back to the command and control server, it appears there most common are cracked apps. See the list for the most common types of Android infections detected by the Sophos antivirus tool[3].

- **Andr/PJApps-C** This category refers to apps that have been cracked using a publicly available tool. The most common example is of a paid version of the app that is now available for free. They aren't always malicious, but are usually illegal.
- **Andr/BBridge-A** BaseBridge uses a privilege escalation exploit to elevate its privileges so that it can download and install additional apps onto the device. BaseBridge also uses HTTP to communicate with a central server and transmit potentially identifiable personal information. BaseBridge can also send and read SMS messages, as well.
- **Andr/BatteryD-A** This type of app promises to extend your device's battery life. Instead, "Battery Doctor" sends potentially identifiable information to a server using HTTP and aggressively displays

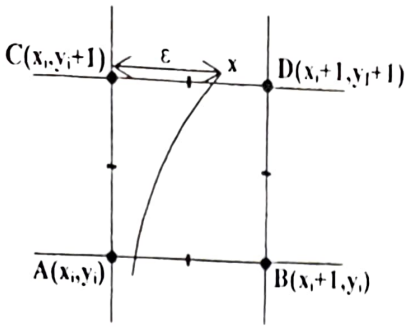


Fig. 7. Hyperbola construction (Region 1).

The above expression is monotonically increasing in the interval $\varepsilon \in [-x_i, +\infty)$. Thus by noting that $d(1/2) = -b^2/2$, the following will hold (Fig. 7):

if $d \geq -b^2/2$ then pixel $D(x_i + 1, y_i + 1)$ is chosen
else pixel $C(x_i, y_i + 1)$ is chosen. (6)

We next derive the incremental computation of the decision variable whose value for the ith step of the algorithm in Region 1 is:

$$d_{1,i} = d1 - d2$$

$$= 2a^2b^2 + 2a^2(x_i + 1)^2 - b^2y_i^2 - b^2(x_i + 1)^2$$

which can be incrementally derived to be:

if $d_{1,i} \geq -b^2/2$ then $x_{i+1} = x_i + 1$ by Equation (6), thus

$$d_{1,i+1} = d_{1,i} + 2a^2 + 4a^2(y_i + 1) - 4b^2(x_i + 1),$$

if $d_{1,i} < -b^2/2$ then $x_{i+1} = x_i$ by Equation (6), thus

$$d_{1,i+1} = d_{1,i} + 2a^2 + 4a^2(y_i + 1).$$

The initial value $d_{1,0}$ is determined by substituting the coordinates of the first pixel of Region 1 $(a,0)$ for (x_i, y_i) in the expression for $d_{1,i}$ in Equation (7):

$$d_{1,0} = 2a^2 - b^2(1 + 2a).$$

In Region 2, expressions for a measure of the distance of the true parabola to the 2 nearest pixel centers are:

$$d1 = a^2(y_i + 1)^2 - a^2y^2$$

$$d2 = a^2y^2 - a^2y_i^2$$

The error term is:

$$d_{2,i} = d1 - d2$$

$$= 2a^2b^2 - 2b^2(x_i + 1)^2 + a^2(y_i + 1)^2 + a^2y_i^2$$

which can be incrementally derived to be:

if $d_{2,i} \leq a^2/2$ then $y_{i+1} = y_i + 1$, thus

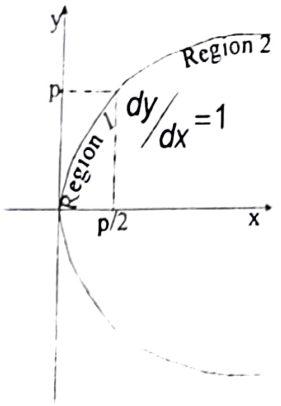


Fig. 8. Parabola.

$$d_{2,i+1} = d_{2,i} - 2b^2 - 4b^2(x_i + 1) + 4a^2(y_i + 1),$$

if $d_{2,i} > a^2/2$ then $y_{i+1} = y_i$, thus

$$d_{2,i+1} = d_{2,i} - 2b^2 - 4b^2(x_i + 1)$$

In a manner similar to the ellipse, the transition criterion from Region 1 to Region 2 is as follows:

if $d < 4b^2(x_i + 1) - b^2/2$ then we remain in the same region else we change region.

The expression for the initial value of the error term in Region 2 can then be derived:

$$d_{2,i} = d_{1,i} - b^2(1 + 2x_i) - a^2(1 + 2y_i)$$

4.2. *Parabola*

Figure 8 shows a parabola centered at (0,0) symmetric about the X-axis defined by the equation

$$y^2 = 2px$$

In Region 1 the axis of major movement is Y while in Region 2 it is X. The two regions meet at $x = p/2$, $y = p$ where the tangent to the parabola has slope $dy/dx = 1$.

In Region 1 the expressions for a measure of the distance of the true parabola to the 2 nearest pixels are:

$$d1 = px - px_i$$

$$d2 = p(x_i + 1) - px$$

The error term is:

protect system resources, and provide application isolation. To achieve these goals the following security features are provided (security overview):

- Robust security at the OS level through the Linux Kernel
- Mandatory application sandbox for all applications
- Secure inter process communication
- Application signing
- Application defined and user granted permissions

The different components and considerations of the Android Software stack are shown below in Figure 1.

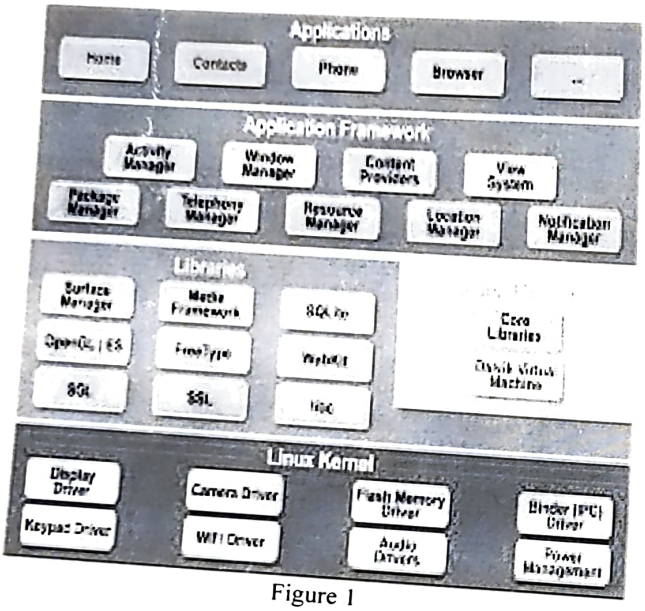


Figure 1

The core of the Android security model is the Linux kernel.
Linux itself has been around for a very long time and is a very robust kernel now after being constantly improved. It is used in the industry and trusted by many professionals. This kernel provides the Android OS with a user-based permissions model, process isolation, a mechanism for secure IPC, and the ability to remove parts of the kernel.

3. User Concerns

This is the detailed description of the user concerns[5].

Versions

There are many different versions of the Android OS, and not all devices use the latest version. Android devices are not updated automatically; individual phone manufacturers have the responsibility to push out updates. This means that if there is a threat that is exploitable in an early version, it can still exist for some users of that old version even if it is fixed in a newer version. Users should make sure that their Android version stays up to date especially if there are security exploits that are found that they are not protected from.

Rooting

Normally, a user does not have full permissions on their Android device, but there is a process called rooting where a user can give himself root privileges on his device. The reasons for doing this include full customization, improved performance, etc., but there are security risks associated with it as well. Since your phone now has root access, the security restrictions on your device are bypassed. A rooted device is susceptible to worms, viruses, spyware, and Trojans that can take control of devices without the user's knowledge. Additionally, there are many more concerns

$$d_{1,i} = d1 - d2$$

$$= (y_i + 1)^2 - px_i - p(x_i + 1)$$

which can be incrementally derived to be:

if $d_{1,i} \geq 0$ then $x_{i+1} = x_i + 1$, thus

$$d_{1,i+1} = d_{1,i} + 2(y_i + 1) + 1 - 2p.$$

if $d_{1,i} < 0$ then $x_{i+1} = x_i$, thus

$$d_{1,i+1} = d_{1,i} + 2(y_i + 1) + 1.$$

$d_{1,0} = (d_1 - d_2) + (0,0)$
$= 1 - p$

In Region 2, expressions for a measure of the distance of the true parabola to the 2 nearest pixel centers are:

$$d1 = (y_i + 1)^2 - y^2$$

$$d2 = y^2 - y_i^2$$

The error term is:

$$d_{2,i} = d1 - d2$$

$$= (y_i + 1)^2 + y_i^2 - 4p(x_i + 1)$$

which can be incrementally derived to be:

if $d_{2,i} < 0$ then $y_{i+1} = y_i + 1$, thus

$$d_{2,i+1} = d_{2,i} + 4(y_i + 1) - 4p.$$

if $d_{2,i} > 0$ then $y_{i+1} = y_i$, thus

$$d_{2,i+1} = d_{2,i} - 4p.$$

The expression for the error, when making the transition from Region 1 to Region 2 can be derived to be:

$$d_{2,0} = d_{1,0} + y_0^2 - p(2y_0 + 3).$$

The square in the calculation of $d_{2,i}$ gives rise to large integers and is unsuitable for hardware implementation. We have proved and verified experimentally that the final value of $d_{1,i}$ will be 1 or $p+1$ and:

if $d_{1,i} = 1$ then $d_{2,i} = -4p + 1$,
if $d_{1,i} = p + 1$ then $d_{2,i} = -2p + 1$,

5. CONCLUSIONS

Despite years of research into basic graphics algorithms, new algorithms still emerge. The integer algorithms for conic sections described in this paper have straightforward Bresenham-like symmetric derivations, are at least as fast as previous integer algorithms, require lower integer arithmetic precision and do not set erroneous pixels at region boundaries, thus incorporating the advantages of well-known previous algorithms. They are very suitable for high performance applications and teaching. Fast antialiasing can also be incorporated.

REFERENCES

1. Pitteway, M. L. V., Algorithms for drawing ellipses or hyperbolae with a digital plotter. *Computer J.*, 1967, **10**(3), 282–289.
2. Bresenham, J. E., A linear algorithm for incremental digital display of circular arcs. *CACM*, 1977, **20**(2), 100–106.
3. Van Aken, J. R., An efficient ellipse-drawing algorithm. *CG&A*, 1984, **4**(9), 24–35.
4. Kappel M. R., An Ellipse-Drawing Algorithm for Raster Displays. In Earnshaw R. (ed) *Fundamental Algorithms for Computer Graphics*, NATO ASI Series, Springer-Verlag, Berlin, 1985, pp. 257–280.
5. Pitteway M. L. V. and Ebadollah Banissi, Soft Edging Fonts. Computer Graphics Technology and Systems. In *Proceedings of the conference held at Computer Graphics '87*, London, October 1987.
6. McIlroy, M. D., Getting Raster ellipses right. *ACM TOG*, 1992, **11**(3), 259–275.
7. Da Silva D., Raster Algorithms for 2D Primitives. Master's Thesis, Computer Science Department, Brown University, Providence, R.I., 1989.
8. Fellner W. D., Computer Grafik. Bibliografisches Institut, Zuerich, 1992.
9. Foley J. D. *et al.*, Computer Graphics, Principles and Practice, 2nd Edn. Addison-Wesley, 1990.
10. Wu, X., An efficient antialiasing technique. *Computer Graphics*, 1991, **25**(4), 143–152.

APPENDIX A

Ellipse Pascal Code

```
Procedure Ellipse(a,b:longint);
var a_sqr,b_sqr,a22,b22,a42,b42,x_slope,y_slope:longint;
    d,mida,midb:longint;
    x,y:integer;

begin
    x:=0;
    y:=b;
    a_sqr:=sqr(a);
    b_sqr:=sqr(b);
    a22:=a_sqr + a_sqr;
    b22:=b_sqr + b_sqr;
    a42:=a22 + a22;
    b42:=b22 + b22;
    x_slope:=b42;        {x_slope = (4*b^^2)*(x + 1) always}
    y_slope:=a42*(y-1); {y_slope = (4*a^^2)*(y - 1) always}
    mida:=a_sqr SHR 1; {a^^2 div 2}
    midb:=b_sqr SHR 1; {b^^2 div 2}
    d:=b22 - a_sqr - y_slope SHR 1 - mida;
    {subtract a^^2 div 2 to optimise}

    {Region 1}
    while d <= y_slope do
    begin
        Draw(x,y);
        if d > 0 then
        begin
            d:=d - y_slope;
            y:=y - 1;
            y_slope:=y_slope - a42;
        end;
        d:=d + b22 + x_slope;
        x:=x + 1;
        x_slope:=x_slope + b42;
    end;

    d:=d-(x_slope+y_slope) SHR 1 +(b_sqr-a_sqr)+(mida-midb)
    {Optimised region change using x_slope , y_slope}
```

PROJECT PROPOSAL ON ANDROID MELWARE DETECTION

Android is a modern mobile platform that was designed to be truly open. Android applications make use of advanced hardware and software, as well as local and served data, exposed through the platform to bring innovation and value to consumers. It is one of the most popular mobile platforms.
According to Statista [1], The Statistics Portal, the number of applications available for download in leading app stores in July 2014. As of that month, android users were able to choose between 1.3 million apps.
With the explosive growth of smart mobile devices market and usage, there are an increasing number of malicious mobile applications that are developed to target these devices and platforms. These malicious applications are called mobile malware. Nowadays, mobile malware have reached a new level of maturity. Threats targeting smart phones and tablets are beginning to pose meaningful challenges to users, enterprises, and service providers alike. The number of instances of just one family of malware can be in the thousands. The largest proportion of malware is targeting on the Android mainly due to the dominant market share of the Android platform and its open market policy. Securing an open platform requires a robust security architecture and rigorous security programs. Android was designed with multi-layered security that provides the flexibility required for an open platform, while providing protection for all users of the platform.
This research paper focuses on the type of android malwares and define changes in the android application applications by introducing threads and asynchronous tasks in it, in order to make them malicious.

A. ANDROID

1. Overview

Created by Google, Android is a most popular cell phone operating system for mobile devices including smartphones and tablets. It is available to all kinds of developers with various expertise levels, ranging from rookie to professional. Based on the Linux kernel Android can provide a middleware implementing subsystems such as telephony, window management, and management of communication with and between applications, managing application lifecycle, and so on. On top of the kernel, Android provides all types of apps for users.
Android applications are programmed primarily in Java through the programmers. Native programming are allowed via Java native interface. Different from Java compilation process, instead of Java bytecode, Android outputs and runs Dalvik bytecode. In comparison to Java, all the compiled classes are generated and packed together into a single .dex file in Dalvik.
An Android application is composed of four types of components, namely activities, services, broadcast receivers, and content providers. All these four components are defined as classes in the library. They are further declared in the AndroidManifest (a.xml file used for the web browser). The Android end user apps will directly or indirectly use these components and interact with the kernel through them. AndroidManifest is a manifest file defined as binary XML file, which declares the application package name. A package id is defined as a string and needs to be unique to an application. It also declares other things (such as application permissions) which are not so relevant to the present work.
Android SDK (Software Development Kit) includes a virtual mobile device emulator that allows android apps to run on the computer. The Android emulator mimics all of the hardware and software features except it cannot place actual phone call. The current Android version is 5.0.

2. Android Security Model

The Android security model[5] was designed with multiple layers that provide flexibility as well as sufficient protection for all of the consumers of the platform. The flexibility of the platform allows developers of all experience levels to easily work with the SDK to build secure applications. Visibility to the users is also very stressed with the Android security model. Users are given information on how applications work and what permissions the applications have on their device.

Security Architecture
The Android operating system's goal is to protect user data,

```
        {Region 2}
        while y >= 0 do
        begin
            Draw(x,y);
            if d <= 0 then
            begin
                d:=d + x_slope;
                x:=x + 1;
                x_slope := x_slope + b42;
            end;
            d:=d + a22 - y_slope;
            y:=y-1;
            y_slope:=y_slope - a42;
        end;
    end;
```

APPENDIX B

Hyperbola Pascal Code

```
Procedure Hyperbola(a,b:longint;bound:integer);
{bound limits the hyperbola in y}
var x,y,d,mida,midb:longint;
    a22,b22,a_sqr,b_sqr:longint;
    a42,b42:longint;
    x_slope,y_slope:longint;
begin
    x:=a;
    y:=0;
    a_sqr:=sqr(a);
    b_sqr:=sqr(b);
    a22:=a_sqr+a_sqr;
    b22:=b_sqr+b_sqr;
    a42:=a22+a22;
    b42:=b22+b22;
    x_slope:=b42*(x+1); {x_slope = (4*b^^2) * (x + 1) always }
    y_slope:=a42;       {y_slope = (4*a^^2) * (y + 1) always }
    mida:=a_sqr shr 1;{a^^2 div 2}
    midb:=b_sqr shr 1;{b^^2 div 2}
    d:=a22 - b_sqr * (1+2*a) + midb; {add b^^2 div 2 to optimize}

    {Region 1}
    while (d < x_slope) and (y<=bound) do
    begin
        Draw(x,y);
        if d >= 0 then
        begin
            d:=d - x_slope;
            x:=x + 1;
            x_slope:=x_slope + b42;
        end;
        d := d + a22 + y_slope;
        y := y+1;
        y_slope := y_slope + a42;
    end;

    d:=d - (x_slope + y_slope) shr 1 + (a_sqr+b_sqr) - midb - mida;
    {optimised region change using x_slope , y_slope}

    {Region 2}
    if a>b then
    while y <= bound do
    begin
        Draw(x,y);
        if d<=0 then
        begin
            d:=d+y_slope;
            y:=y+1;
            y_slope:=y_slope + a42;
        end;
        d:=d - b22 - x_slope;
        x:=x + 1;
        x_slope:=x_slope + b42;
    end;
end;
```

APPENDIX C

Parabola Pascal Code

```
Procedure Parabola(p,bound:integer);
{bound limits the parabola in x}
var x,y,d:integer;
    p2,p4:integer;
begin
    p2 := 2*p;
    p4 := 2*p2;
    x  := 0;
    y  := 0;
    d := 1 - p ;

    {Region 1}
    while (y < p) and (x <= bound) do
    begin
        Draw(x,y);
        if d >= 0 then
        begin
            x := x + 1;
            d := d - p2;
        end;
        y := y + 1;
        d := d + 2*y + 1;
    end;
    if d = 1 then d := 1 - p4
    else d := 1 - p2;

    {Region 2}
    while x <= bound do
    begin
        Draw(x,y);
        if d <=0 then
        begin
            y := y + 1;
            d := d + 4*y;
        end;
        x := x + 1;
        d := d - p4;
    end;
end;
```

The Chairperson,
Research Council,
University of Delhi,

Delhi-110007.

Madam,

Please find enclosed the research proposal for research and development grant for the year 2016.

It is to request you to sanction a grant of Rs. 1.5 Lacs to support the research in the proposed area as per the attached requirements

With regards,

Yours faithfully,

S K Muttoo

Enclosures: Research proposal along with Budgetary Requirements