VBVenn—A Visual Basic Program for Calculating and Graphing Quantitative Two-Circle Venn Diagrams

By GREGORY E.GRANATO

A Contribution to the
NATIONAL HIGHWAY RUNOFF DATA AND METHODOLOGY SYNTHESIS

Northborough, Massachusetts 2001

CONTENTS

List of Figures

Note: This program was created, reviewed, approved, and distributed as a component of the CD-ROM accompanying the report:

Granato, G.E., Dionne, S.G., Tana C.K., and King, T.L., 2002, National Highway Runoff Water-Quality Data and Methodology Synthesis, volume II—Project Documentation: Federal Highway Administration Research Report FHWA-EP-02-008, 22p.

Abstract

VBVenn is a Visual Basic program that calculates the size and position of two circles to construct a quantitative Venn diagram. It is a simple one-form program with a fairly intuitive user interface. The program normalizes the size of two populations and their intersection, sizes a circle for each population, calculates the area of the intersection, solves for the relative position of the circles in the Venn diagram by iteration, produces a graphical rendition of the Venn diagram on the screen, and writes an output text file. This text file may be used with graphing software to create a publication-quality Venn diagram. This Report provides a guide for using *VBVenn*, describes the calculations used to create the diagram, and documents the Visual Basic code used to implement the iterative method.

VBVenn was created by the author on personal time and therefore was not produced at cost to the taxpayer. *VBVenn*, however, was partially produced and initially distributed using U.S. Government computers. Therefore *VBVenn* is in the public domain and may be utilized and distributed as freeware. As such, this software may not be copyrighted or sold as a commercial product.

INTRODUCTION

In October, 1996, the Federal Highway Administration and the U.S. Geological Survey began the National Highway Runoff Data and Methodology Synthesis (NDAMS) to catalog and assess the suitability of pertinent and highway-runoff information for regional or National synthesis efforts. State and Federal highway decision makers need valid current and technically supportable information to determine the water-quality of highway runoff and potential ecological effects of highway planning, design, construction, maintenance, and operation efforts. A large number of complex and interrelated factors were considered to evaluate the suitability of information and data for synthesis efforts (Granato, 2002).

It can be difficult to convey relations between different groups without graphs to illustrate the complexities of the different information necessary for various decision objectives. For example, to determine the effects of runoff on aquatic life in a stream a study must define both *A* the runoff quality entering the stream and *B* the effect of runoff on aquatic biota in the stream. In the NDAMS study it was determined that about 81 percent of the studies had water samples (*A*), about 19 percent had evaluated the ecological state of aquatic biota in receiving waters (*B*), but only 11 percent had documented sampling efforts for both *A* and *B*. Quantitative Venn diagrams clearly convey these types of relations between and among populations.

No readily available software packages were designed to produce quantitative Venn diagrams. A number of resources provide definitions and static diagrams for discussion or illustration, but do not provide a quantitative graphing tool. A search of the literature revealed that the Venn diagram is considered useful for logic theory and set notation, but these diagrams are typically used as qualitative representations of the relations between two or three populations with one or more subsets of common elements. A search of the Internet revealed the availability of a number of static and qualitative Venn-diagram forms, but confirmed that there was no readily available quantitative Venn-diagram software.

A Venn diagram is a graphical representation of the relation between two or three sets. Circles are commonly used to represent the relative size of the subject populations and the amount of overlap between circles represents the relative size of the intersections. For example figure 1 is a simple 2-circle Venn diagram, which graphically conveys the information that population *A* is larger than population *B* and that the intersection of *A* and *B* (the subset that has both characteristic *A* and characteristic *B*) is relatively small.

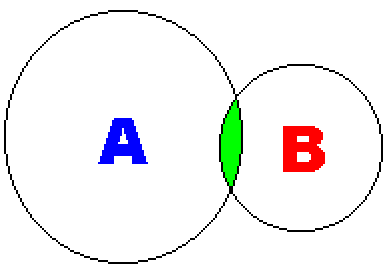

Figure 1. Example of a two-circle Venn diagram showing the relative sizes of populations *A* and *B*, and their intersection (the green shaded area).

The purpose of this report is to provide a guide for using *VBVenn*, to describe the calculations used to create the quantitative Venn diagrams, and to document the Visual Basic code that is used to implement the iterative method.

Acknowledgements

The Author gratefully acknowledges Berwyn Jones, Cameron Tana, and Chester Zenone of the U.S. Geological Survey and Patricia Cazenas of the Federal Highway Administration for reviews and feedback that improved both the program and the documentation. Brian Reece and Steve Tessler of the U.S. Geological Survey were helpful Visual-Basic mentors. Susan Granato patiently surrendered “quality time” with the author many evenings and weekends while this program was written and documented.

THE VBVENN PROGRAM

VBVenn converts information about two populations and their intersection into a Venn diagram that quantitatively represents the relative size of the poulations and their intersection. *VBVenn* provides a graphical form that may be printed to document the results of calculations. *VBVenn* also creates a space-delimited text file with the coordinates of the Venn circles. This text file can be used with available computer programs (such as a spreadsheet, a statistical package, or a graphing program) to create quantitative graphs for use in presentations or publications.

VBVenn creates a two-circle Venn diagram. A two-circle Venn is relatively simple to program because there is one intersection (*AB*) therefore the position of the circles may be located in one dimension along a common axis. An exponential increase in time and effort would be required to program a three circle Venn because there are three individual intersections (*AB*, *AC*, and *BC*), and a fourth composite intersection (*ABC*) to be calculated and positioned in two-dimensional radial space. When a three circle Venn is required, the user may determine each intersection separately and add a third circle to a two-circle graph using the law of cosines.

VBVenn normalizes the populations and the Venn circle sizes so that the size of the circles and the size of the intersection are proportional to the total number of potential members in the sets being considered. For example, consider the case where 100 people are studied. Fifty people have characteristic *A*, 50 people have characteristic *B*, and 25 people have both *A* and *B*. *VBVenn* divides *A, B,* and the intersection by 100 (the total possible number). The program then iteratively calculates the relative position of the circles and creates an on-screen graphic and the output text file with the position of the properly scaled Venn circles.

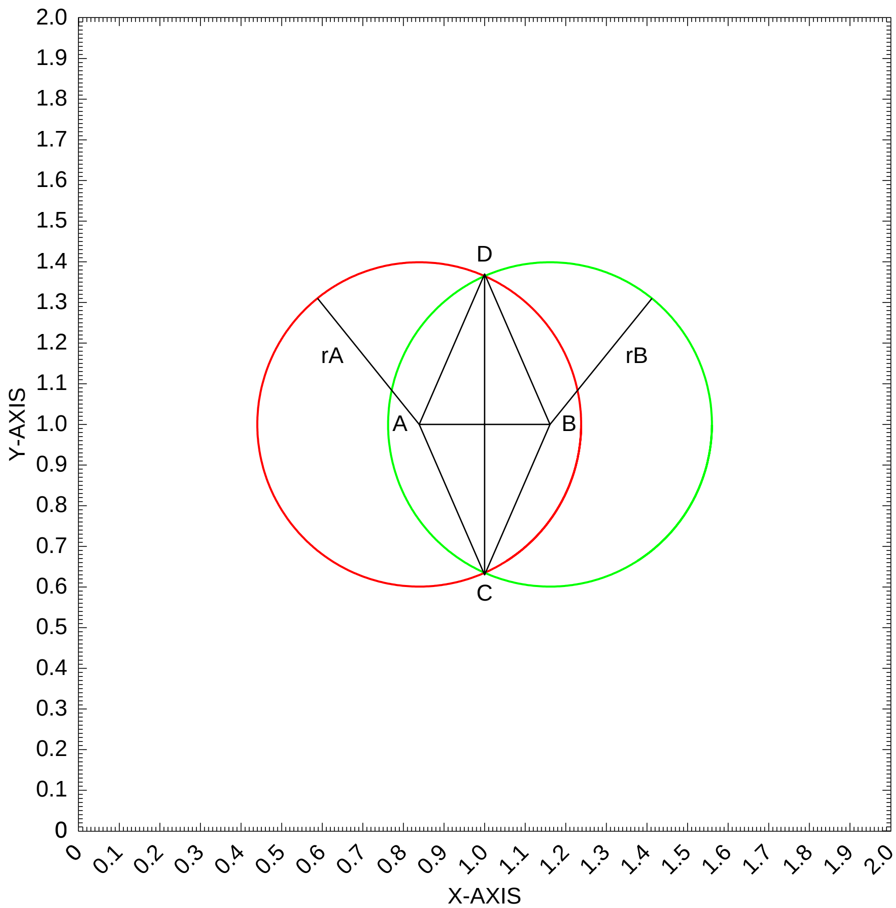

Figure 2. A quantitative Venn diagram with circles A and B, the distance between circle centers (line AB), the circle radii (lines rA and rB), the circle intersection points (D and C), and the intersecting angles (angles CAD and DAC).

Program Theory

VBVenn calculates the position of the two Venn circles by dividing the population of the intersection by the total population to determine the area of the intersection. If the intersection is zero, (there are no members with both the characteristics *A* and *B*) *VBVenn* sizes the Circles for *A* and *B* and placed them so that the circles touch, but do not overlap (to avoid division by zero errors in the distance iteration). If all of population *B* falls within population *A* the circles placed so that the edges of the circles touch and *B* is within *A* (which also avoids division by zero errors). For the remaining cases, the circles are initially set so that there is a small overlap (0.0001 units) and both circles are incrementally moved toward the center in steps of 0.0001 units until the area of intersection (as a function of the distance between centers) equals the area specified in the input (fig 2).

To determine the area of intersection it is necessary to calculate the areas of each circle that are inscribed by the intersecting points (D and C in figure 2). This calculation requires a number of steps as follows. Two circles A and B are placed on a common y axis at locations xA, yA and xB, yB (where y equals 1.0). Because the areas of the respective circles represent population size the radius of the populations are calculated:

Radius= SQRT((MPopulation /Total Popuation/ PI),

Where
MPopulation is the population of A or B,

Total Population is the total popuation of interest, and

PI is the ratio of the circumference of a circle to its diameter (about 3.14159).

The length between circle centers is calculated using the Pythagorean theorem as:

AB = SQRT((xB-xA)^2 + (yB-yA)^2)

Where

xA and **xB** are the x-coordinate of circle centers, and

yA and **yB** are the x-coordinate of circle centers for circles A and B respectively.

The cosine formula is then used to calculate the angle (in radians) between the center lines and the intersections (for example angles CAB and CBA in figure 2).

$$\text{Cos(CAB)} = \frac{\text{RadiusA}^2 + \text{AB}^2 - \text{RadiusB}^2}{2 * \text{RadiusA} * \text{AB}}$$

Angle CAD = 2 * Angle CAB

Similiarly,

$$\text{Cos(CBA)} = \frac{\text{RadiusB}^2 + \text{AB}^2 - \text{RadiusA}^2}{2 * \text{RadiusB} * \text{AB}}$$

and

Angle CBD = 2 * Angle CBA

Finally, to determine the proportion of each circle cut off by chord CD it is necessary to determine the area of each circle included within the angles CAD and CBD and subtract the area of the triangles. When the intersection represents half or more of population (as in figure 2) the negatives cancel and these areas add. The areas are calculated as follows:

Area= (CAB* RadiusA^2) – (0.5* RadiusA^2 * Sin(CAD)) + (CBA* RadiusB^2) – (0.5* RadiusB^2 * Sin(CBD)).

If the calculated area does not equal the specified intersection the program incrementally moves each circle toward the center and recalculates the angles and area of intersection.

INSTALLING VBVENN

The *VBVenn* program is written and compiled in Microsoft Visual Basic. Therefore it may only be used on computers with a Windows operating system. The program files and solution iterative code (applendix 1) are provided for future modification and or adaptation to other programming languages and operating systems. The files included on disk are as follows:

Visual Basic Proprietary files (for programmers):
2circvenn.vbp – the Visual Basic project file,
2circvenn.vbw – the Visual Basic workspace file,
VennForm1.frm – the Visual Basic form file,
VennForm1.frx -- a binary form file, and
MSSCCPRJ.SCC -- a Microsoft source code control file
The Program file:
VBVenn2C.exe
Other files:
VennOut.txt -- an example output file,
VBVenn00.jpg – An image of the interface without data,
VBVenn01.jpg – An image of the interface with input data, and
VBVenn.jpg – An image of the interface with input and output data.

Most users, however, will require only the executable program code (VBVenn2C.exe) to use this program.

USING VBVENN

It is easy to use *VBVenn*. Simply click on the program icon to start the program. The program interface will appear as a blue form on the right hand side of the user's video monitor (fig 3).

The user must enter the population values including the Total Number (the largest number which will be used to normalize the population to a basis of one), the value for Population A (which must be the larger of the two groups considered), the value of population B (which must be less than or equal to population A), and the intersection of populations A and B (which must be less than or equal to population B). An example of population input values is provided in figure 4. The default output filename is VennOut. The user may change this name if more than one output file will be used. *VBVenn* will automatically append the file suffix ".txt" to the text in the output filename field.

Once the values are entered and the output filename has been specified, the user must click on the calculation bar. The calculation bar is labeled "2. Calculate Variables." The program calculates the variables, displays the results (in the "Calculated Results" fields below the calculation bar) and displays the appropriate 2-circle Venn diagram in the graphic window at the bottom of the screen (fig 5).

Once results are calculated the program provides several methods for documenting results. The output file contains coordinates for two intersecting circles of the Venn diagram. The user may either respecify the output filename in on the form, or copy the output file results to a different filename.The print button on the form provides a method to print the Venn diagram and calculated output for future use. The user may also obtain a screen capture of the results by simultaneously pressing the "Alt" and "PrintScrn" keys on the keyboard to capture the image and by pasting the image into a word processor (such as MSWord) or a graphics program (such as PCPaint).

The user may repeat the calculation command or close the program after each calculation. If more than one Venn diagram is desired, the user may simply respecify the input and click the calculation button as many times as is desired. If, however, the output filename is not respecified the program will overwrite the existing file. Once the user is done with the program, they may click on the Close button to exit *VBVenn.*

VBVenn also has features to help the users. The program interface has on-screen instructions. The tool-tip help feature of Visual basic was used in designing and implementing the interface. Therefore, information and instructions will appear when the user pauses the screen cursor over an input field, an output field, or a form control button.

VBVenn is provided as freeware with no expressed or implied warrantee for its use. The program is a simple one-form visual basic application and has no known bugs, but has not been rigorously tested (beyond the accuracy of Venn diagram calculations). This program may be redistributed freely, but may not be copyrighted or sold for profit.

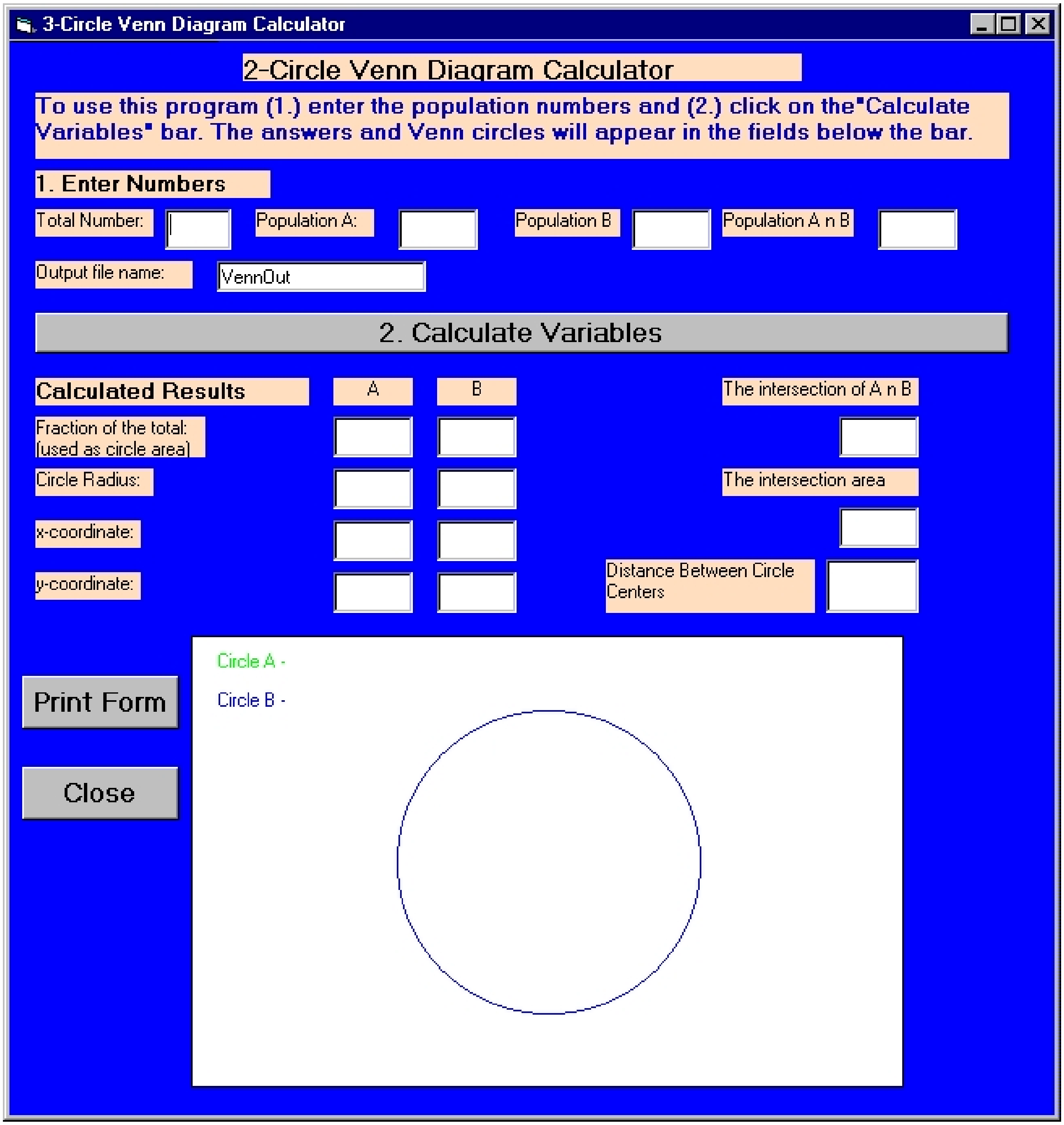

Figure 3. The VBVenn program interface showing the initial condition of the program interface.

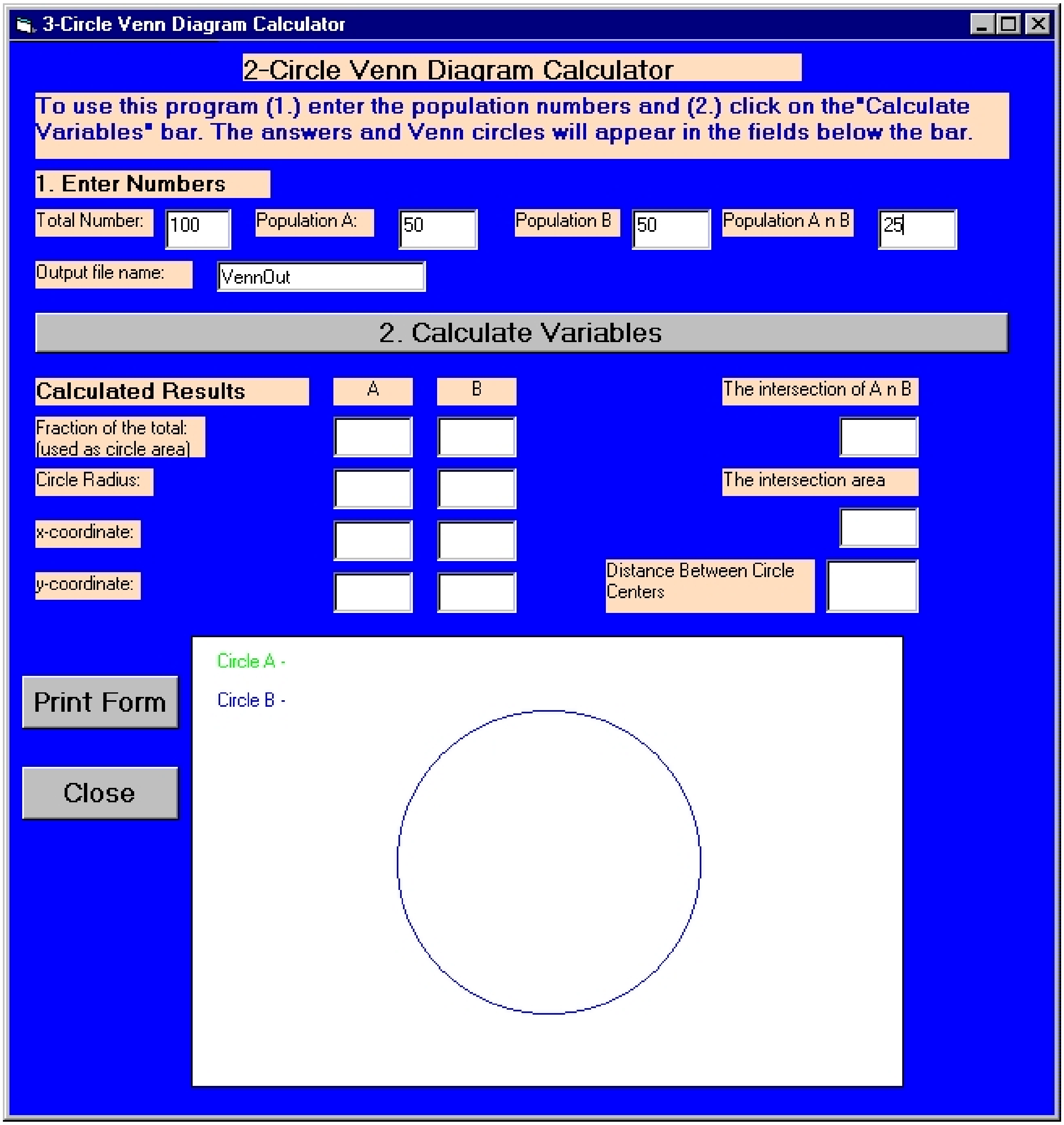

Figure 4. The VBVenn program interface showing the program interface once the input data has been entered, but before the "Calculate Variables" feature has been activated.

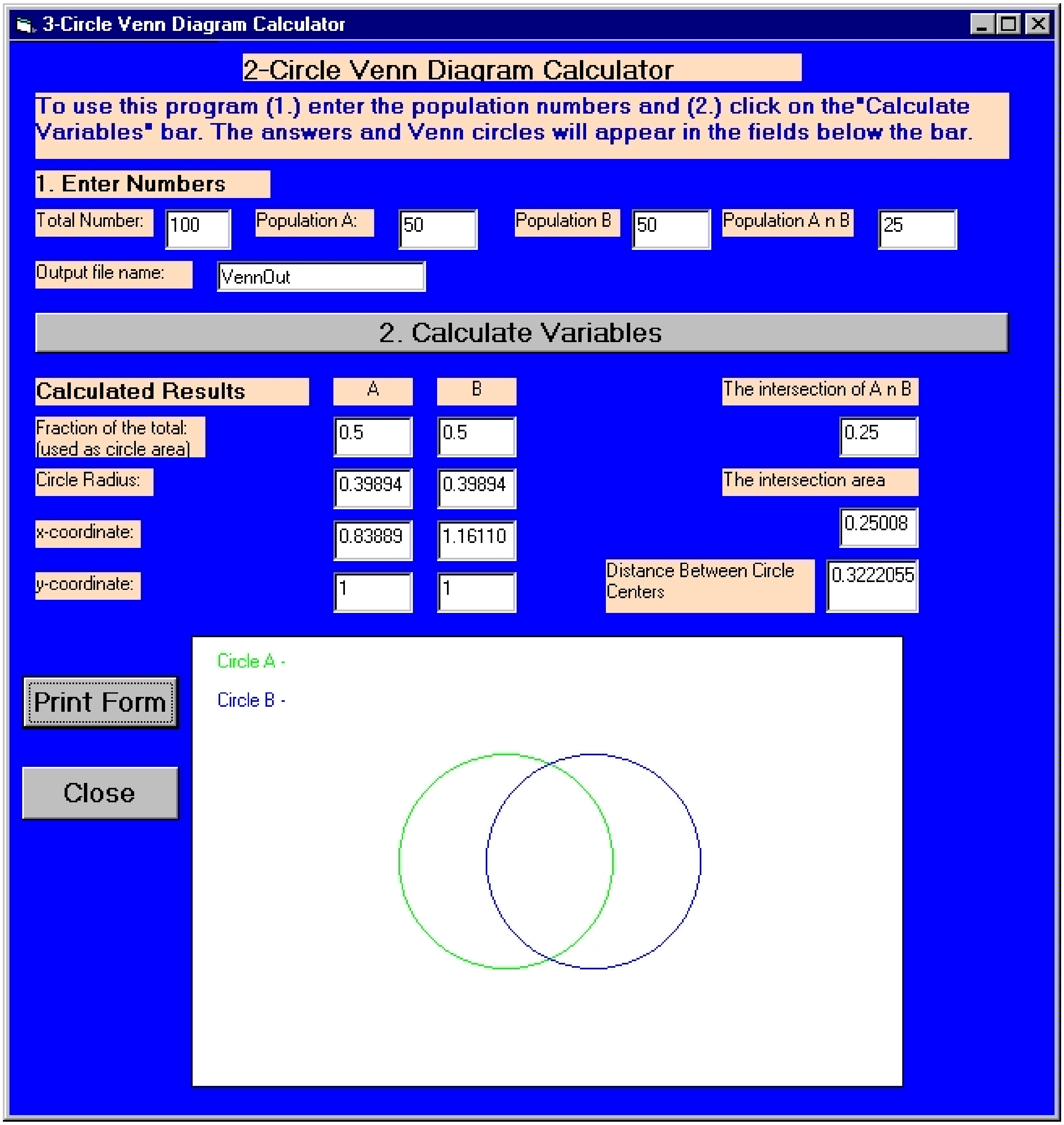

Figure 5. The VBVenn program interface showing the program interface once the input data has been entered, the "Calculate Variables" feature has been activated, and the program results have been displayed.

GRAPHING VBVENN OUTPUT

The output file is a space delimited ASCII file with one header line. The columns are labeled xa and ya as the coordinates for circle *A* and xb and yb as the coordinates for circle **B**. This file contains only the coordinates for the circumference of each circle. Information about the radius, the coordinates of each circle center, and the area of intersection must be retained from the program interface. The circles are provided in Cartesian coordinates and the intersection of the circles is centered on the coordinate x=1, y=1. As described in the theory section, the circles are normalized to a maximum size of one and are moved in the x direction along the axis y=1 to simplify calculations and standardize the graphs.

The output file provides for high-quality graphing with commercial graphing software. The output file contains 729 xy coordinate pairs for each circle. Therefore, there are two linear segments for each degree around the circle (fig 6). This resolution proved sufficient when tested in several graphing packages at several scales.

If a three circle Venn diagram is desired the individual output files can be imported into a spreadsheet and the x-y coordinate values to offset the position of the third circle so its center is the proper distance from each circle in the primary circle-pair.

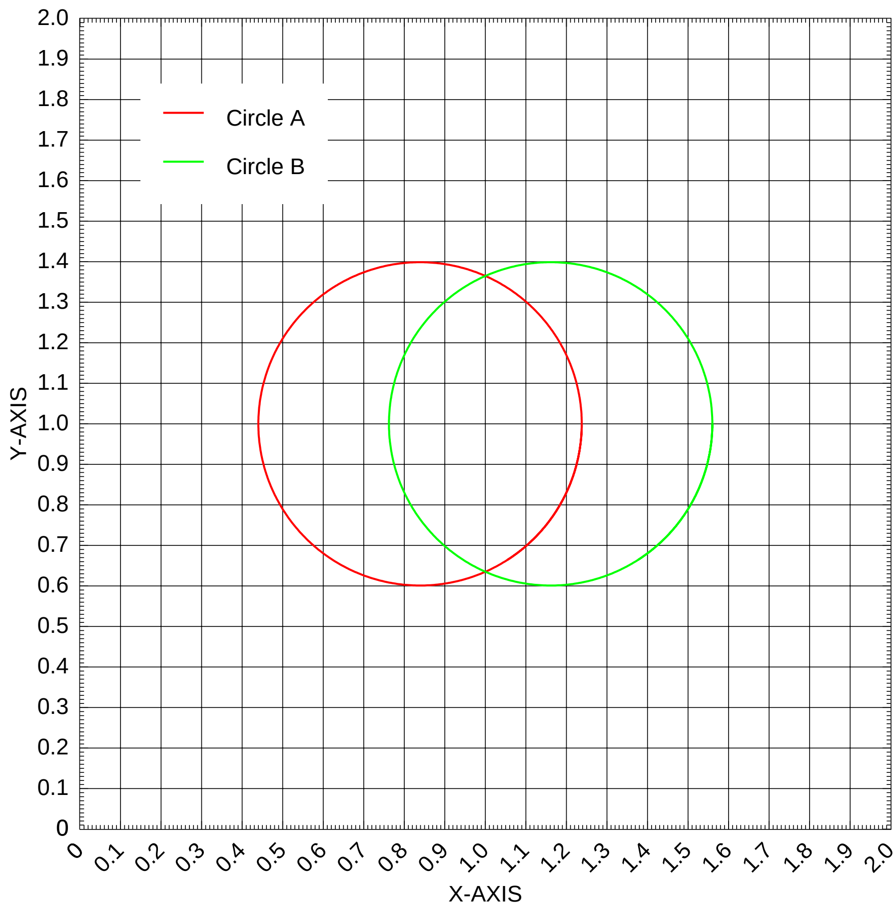

Figure 6. A quantitative Venn diagram with gridlines to show the normalization of coordinates, the position of the circles, and the intersection at x=1 and y=1 in Cartesian coordinate space.

Summary

VBVenn is a visual basic program that runs in the windows environment. To use the program users must click on the program icon for the program Venn2C.exe, enter the input values, click the “Calculate Variables” button and record the program output. *VBVenn* is provided as freeware with no expressed or implied warrantee for its use. The program is a simple one-form visual basic application and has no known bugs, but has not been rigorously tested (beyond the accuracy of Venn diagram calculations). This program may be redistributed freely, but may not be copyrighted or sold for profit.

References

Granato, G.E., 2002, National Highway Runoff Water-Quality Data and Methodology Synthesis, volume III—Availability and Documentation of Published Information for Synthesis of Regional or National Highway-Runoff Quality Data: Federal Highway Administration Research Report FHWA-EP-02-009, 71 p.

Granato, G.E., Dionne, S.G., Tana C.K., and King, T.L., 2002, National Highway Runoff Water-Quality Data and Methodology Synthesis, volume II—Project Documentation: Federal Highway Administration Research Report FHWA-EP-02-008, 22p.

APPENDIX 1: PROGRAM CODE for VBVenn

```
Private Sub Command1_Click()
On Error GoTo Err_Command1_Click
Dim Total As Single, PopA As Single, PopB As Single, PopAUB As Single
Dim fPopA As Single, fPopB As Single, fPopAUB As Single
Dim RPopA As Single, RPopB As Single, PI As Single
Dim xA As Single, yA As Single, xB As Single, yB As Single
Dim LAB As Single, iarea As Single, Angle1 As Single, Angle2 As Single, Angle11 As Single,
Angle22 As Single
Dim cAngle1 As Single, cAngle2 As Single
Dim xcircle As Single, ycircle As Single, rcircle As Single
Dim axc As Single, ayc As Single, bxc As Single, byc As Single, theta As Single
Dim fnm As String, fname As String


' Get values
fnm = Me.fnm
Total = Me.Total
PopA = Me.PopA
PopB = Me.PopB
PopAUB = Me.PopAUB

'parse filename
fname = fnm & ".txt"

'Refresh square
 Me.Shape1.Refresh
'ReSet Circles
 Me.CircleA.Left = 5000 - 1410.474
 Me.CircleA.Top = 7600 - 1410.474
 Me.CircleA.Height = 2 * 1410.474
 Me.CircleA.Width = 2 * 1410.474
 Me.CircleB.Left = 5000 - 1410.474
 Me.CircleB.Top = 7600 - 1410.474
 Me.CircleB.Height = 2 * 1410.474
 Me.CircleB.Width = 2 * 1410.474
```

```
'Error Trap
If (PopA > Total) Then
MsgBox "Population A cannot be larger than the total--Please Fix", vbOKOnly
ElseIf (PopB > Total) Then
MsgBox "Population B cannot be larger than the total--Please Fix", vbOKOnly
ElseIf (PopB > PopA) Then
MsgBox "Population B cannot be larger than Population A--Please Fix", vbOKOnly
ElseIf (PopAUB > Total) Then
MsgBox "Population AUB cannot be larger than the total--Please Fix", vbOKOnly
ElseIf (PopAUB > PopA) Then
MsgBox "Population AUB cannot be larger than Population A--Please Fix", vbOKOnly
ElseIf (PopAUB > PopB) Then
MsgBox "Population AUB cannot be larger than Population B--Please Fix", vbOKOnly
Else: GoTo 5
End If
GoTo 40

' Set constants
5 PI = 3.14159265358979

' Calculate the fraction of each population
fPopA = PopA / Total
fPopB = PopB / Total
fPopAUB = PopAUB / Total


' Calculate the Venn Circle Radius for each population
RPopA = Sqr(fPopA / PI)
RPopB = Sqr(fPopB / PI)
```

```
' Set Initial Conditions
  xA = 1# - (RPopA)
  yA = 1#
  xB = 1# + (RPopB)
  yB = 1#
  iarea = 0.00001
' Calculate the distance between centers AUB
If (PopAUB = 0) Then
' Calculate the distance between centers for zero overlap
  xA = 1# - (RPopA + (RPopA / 50))
  xB = 1# + (RPopB + (RPopB / 50))
  LAB = Sqr(((xA - xB) ^ 2) + ((yA - yB) ^ 2))
  GoTo 40
 ElseIf (PopAUB = PopB) Then
' Calculate the distance between centers for 100% overlap
  xA = 1#
  xB = 1# + (RPopA - RPopB)
  LAB = Sqr(((xA - xB) ^ 2) + ((yA - yB) ^ 2))
 GoTo 40
End If
20 If (fPopAUB > iarea) Then
    xA = xA + 0.0001
    xB = xB - 0.0001
    LAB = Sqr(((xA - xB) ^ 2) + ((yA - yB) ^ 2))
    cAngle1 = ((RPopA ^ 2) + (LAB ^ 2) - (RPopB ^ 2)) / (2 * RPopA * LAB)
    cAngle2 = ((RPopB ^ 2) + (LAB ^ 2) - (RPopA ^ 2)) / (2 * RPopB * LAB)
    Angle1 = Atn(-1 * cAngle1 / Sqr(-1 * cAngle1 * cAngle1 + 1)) + 2 * Atn(1)
    Angle2 = Atn(-1 * cAngle2 / Sqr(-1 * cAngle2 * cAngle2 + 1)) + 2 * Atn(1)
    Angle11 = 2 * Angle1
    Angle22 = 2 * Angle2
    iarea = (Angle1 * (RPopA ^ 2)) - (0.5 * (RPopA ^ 2) * Sin(Angle11)) + (Angle2 * (RPopB ^ 2))
- (0.5 * (RPopB ^ 2) * Sin(Angle22))
    Else
     GoTo 40
    End If
    GoTo 20
```

```
' Serve information to form
40  Me.xA = xA
    Me.yA = yA
    Me.xB = xB
    Me.yB = yB
    Me.RPopA = RPopA
    Me.RPopB = RPopB
    Me.LAB = LAB
    Me.fPopA = fPopA
    Me.fPopB = fPopB
    Me.fPopAUB = fPopAUB
    Me.iarea = iarea
    ' calculate circle coordinates for the screen
    ' to be roughly centered in white box
    xcircle = (2500 + (2500 * xA))
    ycircle = (5100 + (2500 * yA))
    rcircle = (2500 * RPopA)
    Me.CircleA.Left = xcircle - rcircle
    Me.CircleA.Top = ycircle - rcircle
    Me.CircleA.Height = 2 * rcircle
    Me.CircleA.Width = 2 * rcircle
    ' check
    ' VennForm1.Circle (xcircle, ycircle), rcircle
    xcircle = (2500 + (2500 * xB))
    ycircle = (5100 + (2500 * yB))
    rcircle = (2500 * RPopB)
    Me.CircleB.Left = xcircle - rcircle
    Me.CircleB.Top = ycircle - rcircle
    Me.CircleB.Height = 2 * rcircle
    Me.CircleB.Width = 2 * rcircle
    '
```

```
   ' Create an output file for g2
   Open fname For Output As #1
   Print #1, "xa  ya  xb  yb"
   ' Calculate circle circumfrence x y coordinates
   For theta = -1 To (2 * PI) Step 0.01
   axc = xA + (RPopA * Cos(theta))
   ayc = yA + (RPopA * Sin(theta))
   bxc = xB + (RPopB * Cos(theta))
   byc = yB + (RPopB * Sin(theta))
   Print #1, axc & "  " & ayc, "  " & bxc & "  " & byc & "  "
   Next theta
   Close #1

     Me!cmdPrint.SetFocus

Exit_Command1_Click:
   Exit Sub
Err_Command1_Click:
   MsgBox Err.Description
   Resume Exit_Command1_Click
50 End Sub
```