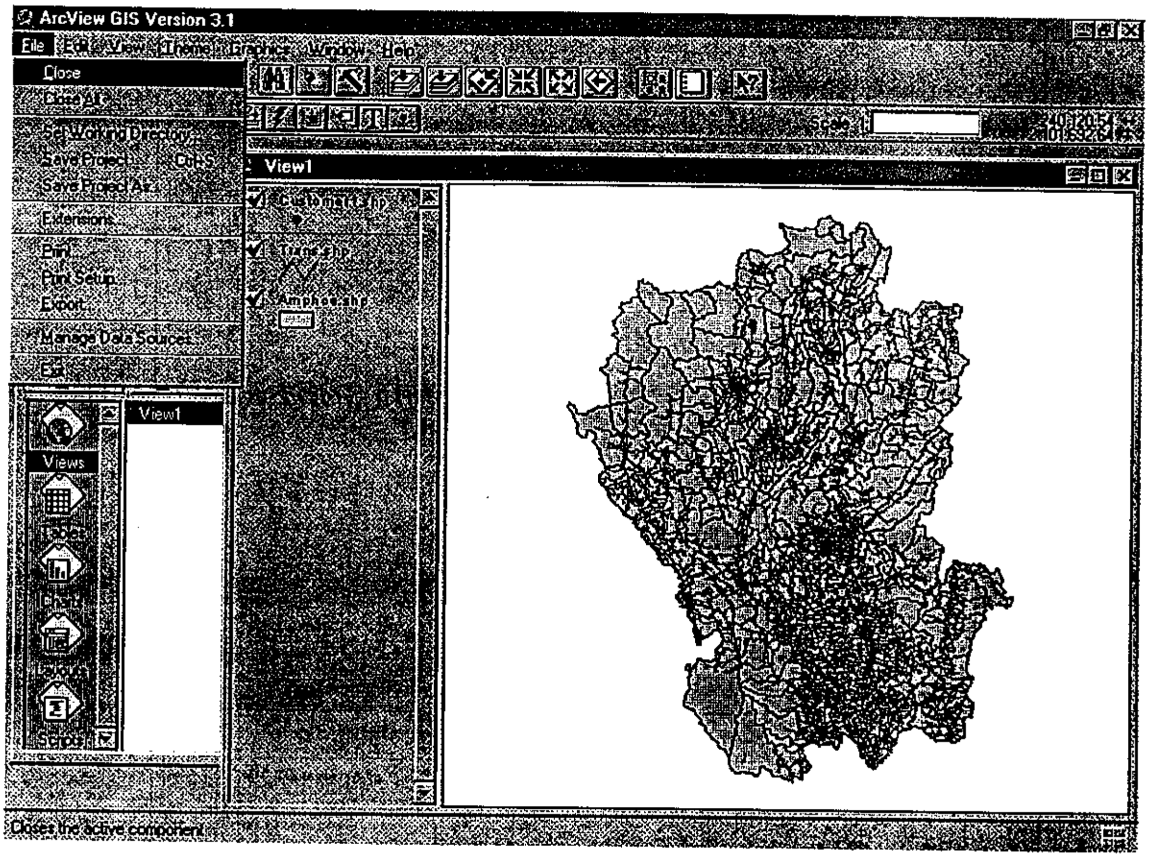
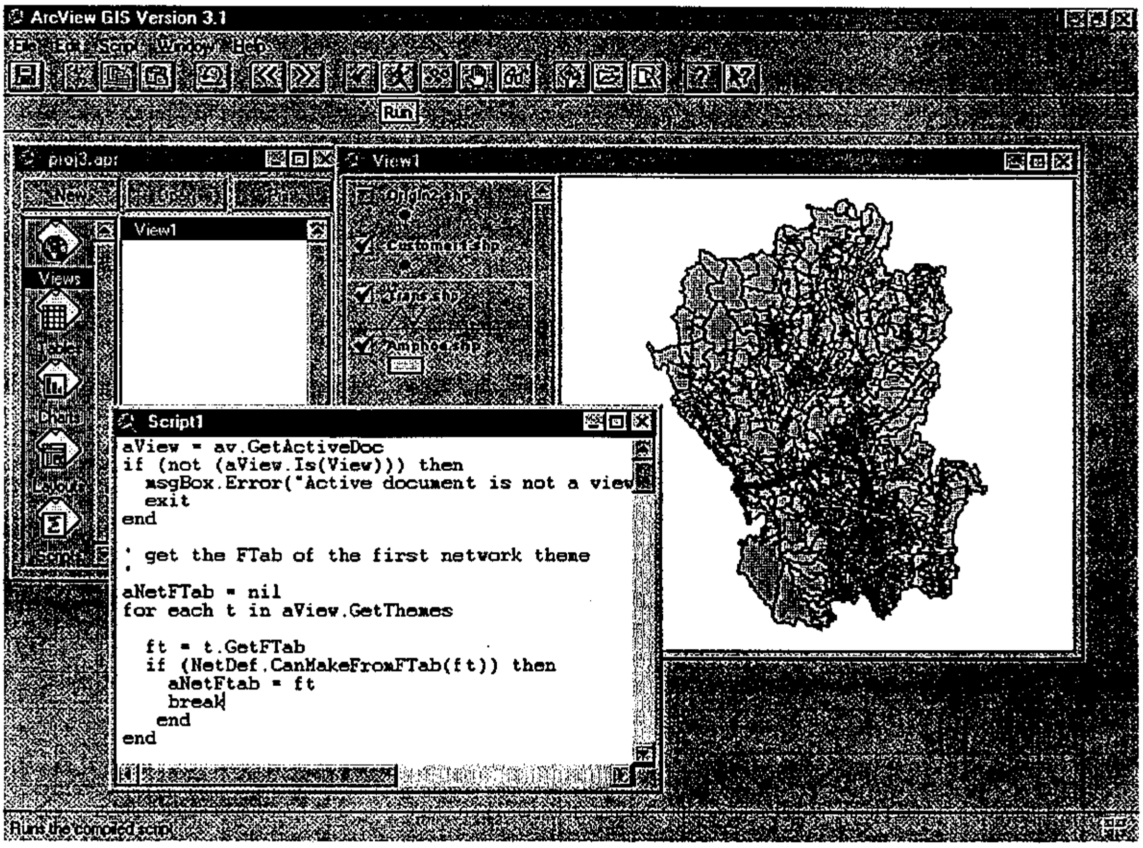
ภาคผนวก ก

ตัวอย่างโปรแกรม Arcview

ตัวอย่างโปรแกรม Arcview ที่ใช้ในการวิจัยครั้งนี้



รูปแบบโปรแกรมทั้งหมด

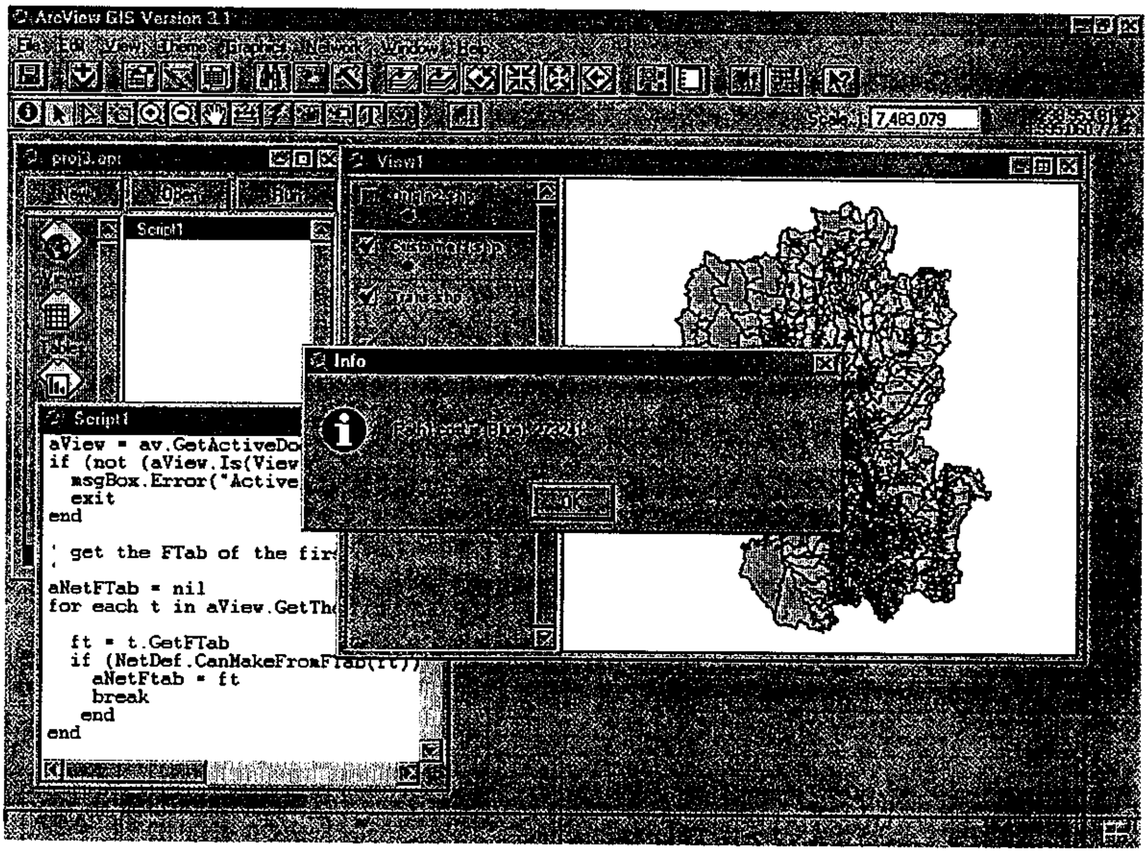


```
aView = av.GetActiveDoc
if (not (aView.Is(View))) then
  msgBox.Error("Active document is not a view
  exit
end

' get the FTab of the first network theme
'
aNetFTab = nil
for each t in aView.GetThemes

  ft = t.GetFTab
  if (NetDef.CanMakeFromFTab(ft)) then
    aNetFtab = ft
    break
   end
end
```

รูปก่อนจะทำการ Run โปรแกรม



```
aView = av.GetActiveDoc
if (not (aView.Is(View
  msgBox.Error("Active
  exit
end

' get the FTab of the fir
'
aNetFTab = nil
for each t in aView.GetTh

  ft = t.GetFTab
  if (NetDef.CanMakeFromFTab(ft))
    aNetFtab = ft
    break
   end
end
```

รูปขณะทำการ Run โปรแกรม

ภาคผนวก ข

ตัวอย่างการเขียนโปรแกรมคำสั่ง FindClosestFac

ตัวอย่างการเขียนโปแกรมโดยใช้ภาษา Avenue คำสั่ง FindClosestFac

```
 aView = av.GetActiveDoc
if (not (aView.Is(View))) then
 msgBox.Error("Active document is not a view.","")
 exit
end
' get the FTab of the first network theme
'
aNetFTab = nil
for each t in aView.GetThemes
 ft = t.GetFTab
 if (NetDef.CanMakeFromFTab(ft)) then
  aNetFtab = ft
  break
 end
end
' did we find a networkable FTab?
'
if (aNetFTab = nil) then
 msgBox.Error("Network theme not found.","")
 exit
end
' make the NetDef and check it for error
'
aNetDef = NetDef.Make(aNetFTab)
if (aNetDef.HasError) then
 msgBox.Error("NetDef has error.","")
 exit
```

```
end

' make the Network object

'

aNetwork = Network.Make(aNetDef)

' make a list of point themes for the user to choose from

'

aPointThemeList = {}

for each t in aView.GetThemes

  if (t.GetFTab.GetSrcName.GetSubName = "Point") then

    aPointThemeList.Add(t)

  end

end

' did we find any point themes?

'

if (aPointThemeList.Count = 0) then

  msgBox.Error("No point themes found.","")

  exit

end

' prompt for the origin theme

'

origTheme = msgBox.Choice(aPointThemeList, "Select the origin point theme:","Origin selection")

' prompt for the destination theme

'

destTheme = msgBox.Choice(aPointThemeList,  "Select the destination (facility) point theme:",

                                    "Destination (facility) selection")


origFTab = origTheme.GetFTab

destFTab = destTheme.GetFTab
```

```
origShapeField = origFTab.FindField("Shape")
destShapeField = destFTab.FindField("Shape")

origLabelField = origTheme.GetLabelField
destLabelField = destTheme.GetLabelField
' Make point lists from the origin and destination point themes,
' validate points, and set the names of the points
'
origPointList = {}
for each rec in origFTab
  p = origFTab.ReturnValue(origShapeField, rec)
  if (aNetwork.IsPointOnNetwork(p)) then
    if (origLabelField <> nil) then
      p.SetName(origFTab.ReturnValueString(origLabelField, rec))
    else
      p.SetName("Origin" + (origPointList.Count + 1).AsString)
    end
    origPointList.Add(p)
  end
end

destPointList = {}
for each rec in destFTab
  p = destFTab.ReturnValue(destShapeField, rec)

  if (aNetwork.IsPointOnNetwork(p)) then
    if (origLabelField <> nil) then
      p.SetName(destFTab.ReturnValueString(destLabelField, rec))
    else
      p.SetName("Destination" + (destPointList.Count + 1).AsString)
    end
    destPointList.Add(p)
  end
end
' set the parameters for FindClosestFac
'
numToFind = destPointList.Count       ' find all destinations
cuttOff     = 77762                   ' set cut off distance
toFrom      = True                    ' travel to destination

' Note: No cost field is selected here, so <line length> will be
'       used by default.
' solve the problem
'
numFoundList = aNetwork.FindClosestFac(origPointList, destPointLis,  numToFind,
                                        cuttOff, toFrom)
```

ภาคผนวก ค

ตัวอย่างการเขียนโปรแกรมคำสั่ง FindPath

ตัวอย่างการเขียนโปรแกรมโดยใช้ภาษา Avenue คำสั่ง FindPath

```
aView = av.GetActiveDoc
if (not (aView.Is(View))) then
  msgBox.Error("Active document is not a view.","")
  exit
end
' get the FTab of the first network theme
'
aNetFTab = nil
for each t in aView.GetThemes

  ft = t.GetFTab
  if (NetDef.CanMakeFromFTab(ft)) then
    aNetFtab = ft
    break
  end
end
' did we find a networkable FTab?
'
if (aNetFTab = nil) then
  msgBox.Error("Network theme not found.","")
  exit
end
' make the NetDef and check it for error
'
aNetDef = NetDef.Make(aNetFTab)
if (aNetDef.HasError) then
  msgBox.Error("NetDef has error.","")
```

```
 exit
end
' make the Network object
'
aNetwork = Network.Make(aNetDef)
' make a list of point themes for the user to choose from
'
aPointThemeList = {}
for each t in aView.GetThemes
  if (t.GetFTab.GetSrcName.GetSubName = "Point") then
    aPointThemeList.Add(t)
  end
end
' did we find any point themes?
'
if (aPointThemeList.Count = 0) then
  msgBox.Error("No point themes found.","")
  exit
end
' prompt for the origin theme
'
origTheme = msgBox.Choice(aPointThemeList,
                    "Select the origin point theme:",
                    "Origin selection")
' prompt for the destination theme
'
destTheme = msgBox.Choice(aPointThemeList,
                    "Select the destination (facility) point theme:",
                "Destination (facility) selection")

origFTab = origTheme.GetFTab
destFTab = destTheme.GetFTab

origShapeField = origFTab.FindField("Shape")
destShapeField = destFTab.FindField("Shape")

origLabelField = origTheme.GetLabelField
destLabelField = destTheme.GetLabelField

' Make point lists from the origin and destination point themes,
' validate points, and set the names of the points
'
origPointList = {}
for each rec in origFTab
  p = origFTab.ReturnValue(origShapeField, rec)
  if (aNetwork.IsPointOnNetwork(p)) then
    if (origLabelField <> nil) then
      p.SetName(origFTab.ReturnValueString(origLabelField, rec))
    else
      p.SetName("Origin" + (origPointList.Count + 1).AsString)
    end
    origPointList.Add(p)
  end
end

destPointList = {}
```

```
for each rec in destFTab
  p = destFTab.ReturnValue(destShapeField, rec)

  if (aNetwork.IsPointOnNetwork(p)) then
    if (origLabelField <> nil) then
      p.SetName(destFTab.ReturnValueString(destLabelField, rec))
    else
      p.SetName("Destination" + (destPointList.Count + 1).AsString)
    end
    destPointList.Add(p)
  end
end
' The user will be prompted to choose one of the origin and destination
' pairs ("assignments"). A graphic for this path will then be added to
' the view.
' Build a string list with the results.
' Note: Writing out the complete "od matrix" would be very similar.
'
resultStringList = {}
ijList = {}
for each i in 0..(origPointList.Count - 1)
  for each j in 1..numFoundList.Get(i)

    ' Get the index into the original destination list for
    ' the current (jthClosest) facility.
    destNum = aNetwork.GetClosestFacIndex(i,j)
    str = "From:" ++ origPointList.Get(i).GetName ++
        "To:"   ++ destPointList.Get(destNum).GetName ++
        "Cost:" ++ aNetwork.GetClosestFacPathCost(i,j).AsString

    resultStringList.Add(str)
    ijList.Add({i,j})
  end
end

' prompt for a specific path
'
choiceNum = msgBox.ListAsString(resultStringList, "Display a path:", "Display Path")
'Set for the pointList

aProject=av.GetProject
aView=aProject.FindDoc("View1")
aThemeC=aView.FindTheme("Customer1.Shp")
aThemeCFTab=aThemeC.GetFTab
aThemeCShapeField=aThemeCFTab.FindField("Shape")

point1=aThemeCFTab.ReturnValue(aThemeCShapeField,0)
point2=aThemeCFTab.ReturnValue(aThemeCShapeField,1)
point3=aThemeCFTab.ReturnValue(aThemeCShapeField,2)
point4=aThemeCFTab.ReturnValue(aThemeCShapeField,3)
point6=aThemeCFTab.ReturnValue(aThemeCShapeField,5)
point16=aThemeCFTab.ReturnValue(aThemeCShapeField,15)
point17=aThemeCFTab.ReturnValue(aThemeCShapeField,16)
point18=aThemeCFTab.ReturnValue(aThemeCShapeField,17)
```

```
point1List=List.Make
point1List.Add(point1)
point1List.Add(point2)
point1List.Add(point3)
point1List.Add(point4)
point1List.Add(point6)
point1List.Add(point16)
point1List.Add(point17)
point1List.Add(point18)

findBestOrder =True
returnToOrigin=True

'Calculate the path
path1Cost=aNetwork.FindPath(point1List, findBestOrder, returnToOrigin)

'Make sure the FindPath succeeded
if((not(aNetwork.HasPathResult))or(path1Cost=0))then
 MsgBox.Error("Path not found.","")
 exit
end

'Display the cost
MsgBox.Info("Path1 cost is (Blue):"++path1Cost.AsString,"")

'Create a shape for the path
aPathShape1=aNetwork.ReturnPathShape
```

```
aGraphicShape1=GraphicShape.Make(aPathShape1)

'Make a nice symbol

aSymbol=Symbol.Make(#SYMBOL_PEN)
aSymbol.SetSize(3)
aSymbol.SetColor(Color.GetBlue)
aGraphicShape1.SetSymbol(aSymbol)

'Add the graphic to the view
aView.GetGraphics.Add(aGraphicShape1)
```