Dynamic Visualizations for the Analysis of Desert Tortoise Telemetry and Habitat in Joshua Tree National Park

David D. Turnbull
University of Redlands

University of Redlands

Dynamic Visualizations for the Analysis of Desert Tortoise Telemetry and Habitat in Joshua Tree National Park

A Major Individual Project submitted in partial satisfaction of the requirements
for the degree of Master of Science in Geographic Information Systems

by

David D. Turnbull

Fang Ren, Ph.D., Committee Chair

Mark Kumler, Ph.D.

August 2010

Dynamic Visualizations for the Analysis of Desert Tortoise Telemetry and Habitat in Joshua Tree National Park

The report of David D. Turnbull is approved.

[illegible]

Mark Kumler, Ph.D.

[illegible]

Fang Ren, Ph.D., Committee Chair

August 2010

Acknowledgements

I would like to thank the University of Redlands and the professors for the opportunity, education, and guidance that enabled me to complete this project. In particular, I give my gratitude to my advisor, Fang Ren, Ph.D., for her guidance and input.

I would like to give much thanks to Joshua Tree National Park GIS Analyst Sean Murphy, whose timely support and enthusiasm with the project allowed me to ensure successful completion.

I would also like to thank Kevin Martin, creator of the *Export to KML Tool* for ArcMap. His assistance and willingness to enhance his tools based upon my request, helped to provide for better animations of the data within Google Earth, ArcGIS Explorer, and ArcMap 10.

The encouragement and support of my supervisor, Jerry Tuttle, made it possible for me to participate in this program.

I would like to dedicate this project to my late grandmother, Marie Zacharias, who always had a house filled with tortoise and turtle figurines.

Of course, I could not have done this without the support of my wife, who suggested that I enroll in the MS GIS program and supported me throughout the project. I also thank my family as a whole for their voting, "Let's go to California!"

Abstract

Dynamic Visualizations for the Analysis of Desert Tortoise Telemetry and Habitat in Joshua Tree National Park

by
David D. Turnbull

Joshua Tree National Park (JOTR) has been monitoring their desert tortoise population's telemetry since 2005 using transmitters attached to as many as 18 tortoises. JOTR needed to visualize tortoise telemetry data and temporal environmental datasets to see what factors may be affecting the movements of the tortoises. JOTR also needed to look at the telemetry data in relation to roads to see if curbing implementations have been effective. The foundation for this project is ArcMap™. This project includes various tools written in Python within ArcMap's toolboxes that allow for automated data manipulation and analysis of the various datasets. These tools prepare the datasets for dynamic visualizations of the data in Google Earth®. Additionally, the preparation of the data will also enable it to be exploited by ArcGIS 10's integrated time-visualization capability. ArcGIS Explorer™ can also be used to perform similar dynamic visualizations as Google Earth. The provided tools and visualizations aim to provide JOTR with a means to help analyze and monitor their tortoise population to improve their survivability.

Table of Contents

Table of Figures

List of Tables

List of Acronyms and Definitions

95PVC	95 Percent Volume Contour
AGX	ArcGIS Explorer
AIS	Aerial Information System
CSV	Comma Separated Values
GIS	Geographic Information System
GPS	Global Positioning System
ISO	International Organization for Standardization
JOTR	Joshua Tree National Park
KML	Keyhole Markup Language
KMZ	Keyhole Markup Language Zipped files
MCP	Minimum Convex Polygon
MXD	ArcMap Project File
NPS	National Park Service
PVC	Percent Volume Contour
USGS	U.S. Geological Survey
UTM	Universal Transverse Mercator

Chapter 1 – **Introduction**

Joshua Tree National Park (JOTR) has been monitoring its desert tortoise population's telemetry since 2005 using radio-transmitters attached to approximately 18 tortoises. The number changes because of tortoises dying or disappearing, as well as new tortoises being equipped with transmitters. The tortoise was placed on both the California and Federal Endangered Species Lists between 1989 and 1990, with a threatened status. This is just one notch below endangered (Davidson, 2009).

JOTR needed to be able to visualize tortoise telemetry, habitat, and environmental datasets in relation to time to determine which factors may be affecting the movements of the tortoises. JOTR also needed to look at the telemetry data in relation to roads to see if new curbing equipped with tortoise trots need to be implemented in other locations to reduce tortoise deaths. A tortoise trot is a gap in the curb that allows the tortoise to easily move off the road without having to climb the curbing. The curbs are put in place to deter vehicles from going off-road.

1.1 Client

The client for this project is the National Park Service (NPS), in particular the JOTR Resource Management Division. Sean Murphy, a GIS specialist with the NPS, was the primary point of contact for the project and was supported by the chief of the JOTR Wildlife Branch, Michael Vamstad.

The client provided the telemetry data gathered from 2005 through 2008 for 18 tortoises, as well as other datasets, including habitat extents, roads, curbing, tortoise trot locations, trails, wilderness areas, and camping areas (Figure 1-1). In addition, it was decided to use the temperature and precipitation data acquired from the data collected from the two NPS air quality stations in JOTR.

Additional work with the client included fieldwork, tracking tortoises, and recording tortoise data to get an understanding of the tracking procedure and witness the tortoises in their environment.

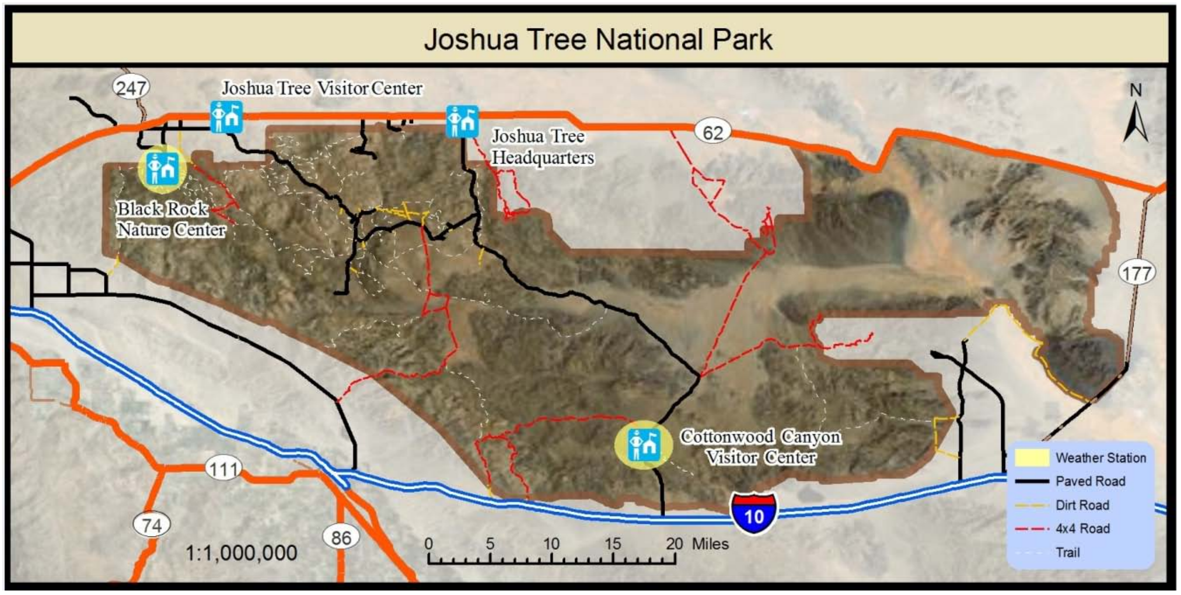


Figure 1-1: Joshua Tree National Park General Layout

1.2 Problem Statement

JOTR currently has over four years of collected telemetry data from 18 of their desert tortoises, and has collected other potentially related data sources. JOTR needed a way to bring together the data to enable analysis through dynamic visualization, as well as to perform analyses using analytical methods. For example, since roads offer the biggest threat to the tortoises, the analyst needed to be able to dynamically visualize and perform analyses on the telemetry data as it relates to the roads, curbing, and tortoise trots. Since many factors may influence the movement of the tortoises, it was possible that other factors were helping to lead the tortoises towards the roads (Figure 1-2). It is difficult to reveal these relationships without an interactive visualization environment. Thus, the lack of a dynamic visualization of the telemetry and related data hinders the analysis of the JOTR tortoise researchers.

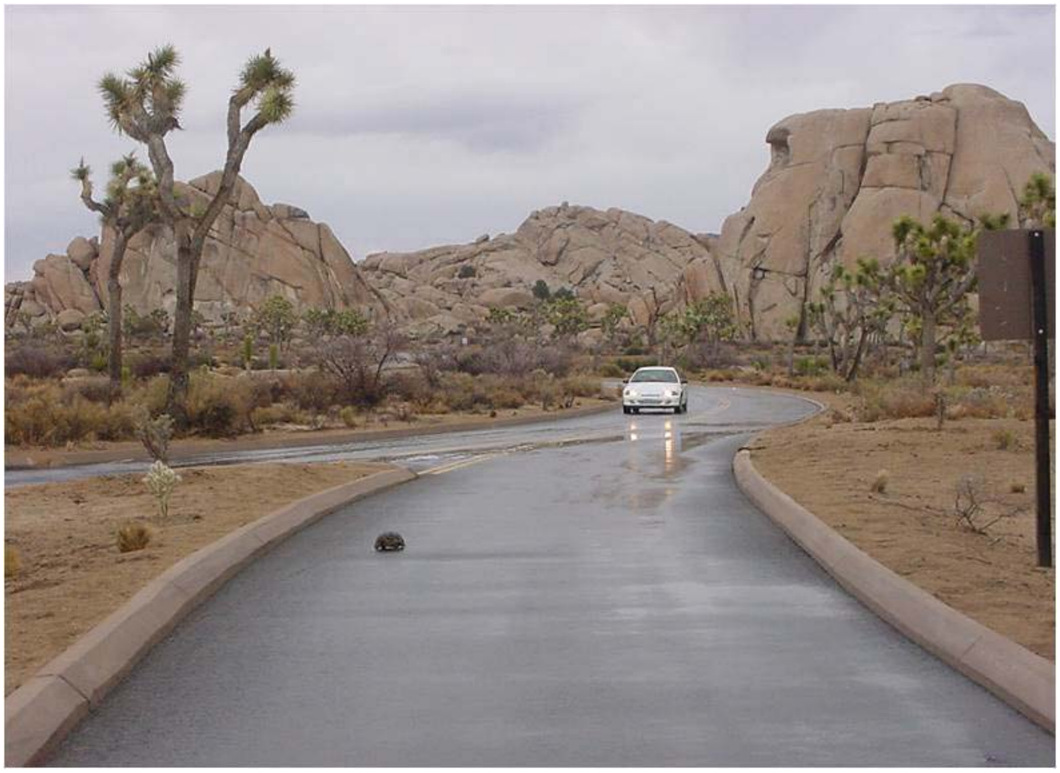


Figure 1-2: Tortoise in Road at Joshua Tree National Park

1.3 Proposed Solution

This project aimed to address JOTR's need to further examine the movement of the desert tortoise in relation to the roads, curbing, and tortoise trots that have been placed along the roads. To reach this goal, ArcMap was used as the foundation for static viewing, data preparation, analysis, and data export. Dynamic viewing was achieved by using the time-attributed and exported Keyhole Markup Language (KML) datasets from ArcMap in Google Earth®. Manipulation of the data was achieved by the creation of several ArcMap tools using Python. These tools prepare existing datasets and create new time attribute fields. The datasets can be exported to KML format through an existing user-built KML creator for ArcMap. By having the two time attributes, the data can be animated and visualized dynamically within Google Earth. Additionally, in ArcGIS 10, these attributes will enable dynamic visualization with ArcMap. The resulting visualization includes tortoise movement, potential tortoise movement, daily high and low temperatures, daily precipitation, implementation of curbing, tortoise trots, roads, and associated buffers. A secondary option for dynamic viewing is ArcGIS Explorer™ (AGX). The produced KML datasets will view in AGX. However, AGX currently lacks the capability to change KML layer attributes, such as color, symbol, and elevation.

1.3.1 Goals and Objectives

The client’s main goal was to help improve the survivability of the tortoises by examining the tortoise telemetry data with the other related data. As part of this goal, there were two objectives in this project. First, the project aimed to create a dynamic visualization framework to view the tortoise telemetry data and other related data, such as roads, curbs, and tortoise trot locations, precipitation, and temperature. Second, the developed tools should enable the client to perform different types of analysis on various related datasets. In doing so, the client will be able to analyze and predict the tortoise telemetry and possibly help lead JOTR to develop new means to help keep the tortoises from going near the roads. The project may also prove useful for analyzing other species at JOTR, as well as at other locations where tortoise or other animal behaviors are being studied.

1.3.2 Scope

The project study area included various regions within the extent of JOTR. Many spatial factors were considered as potentially affecting the survivability of the desert tortoise at JOTR. For this project, all of the telemetry data collected by JOTR going back to early 2005 was used. The project also considered other datasets that might be related to the tortoise movement. These included habitat extents, roads, curbs, tortoise trot locations, trails, wilderness areas, and camping areas. The temperature and precipitation data obtained from an online database of two NPS air quality stations in JOTR were also included.

The tools in the project include the *Tortoise Import Tool* for automated creation of multiple feature classes, the *Stationary Event from Table Tool* for creating temperature and precipitation polygons, the TimeStamp and TimeEnd Field *Creator Tool* for modifying all feature classes to be animated, the *Road and Trot Buffer Tool* for creating safety buffers around the roads and tortoise trots feature classes, the *Tortoise Maximum Speed Finder Tool* to calculate the maximum achieved speed by each tortoise over time, and the *Tortoise Potential Path Tool* to create ellipses representing potential area traversed between recorded positions.

The data were delivered as custom-built layers in an ArcMap project file (MXD). This MXD contains all of the custom-built tools that were used to automatically create the layers, and the tool that was used to export the layers into KML format. All of the layers from the MXD are delivered in a zipped KML, known as KMZ format. These can be viewed in Google Earth as well as AGX. Once loaded, the KMZ can be viewed dynamically using the time slider within the application.

1.3.3 Methods

To accomplish dynamic visualization of the layers within Google Earth or AGX, existing tools as well as new tools have been developed. The data underwent three phases. The initial phase brought the dataset into ArcMap. Data import was not a straightforward process, as feature classes of individual tortoises were required. The client provided a master feature class of all tortoise positions, as well as 68 manually created sub-feature classes of individual tortoises. However, these sub-feature classes will need to be constantly created, as updated data are collected. Therefore, extracting sub-feature class is a routine process faced by the client and manual processing is not efficient. As such, a new tool within the ArcMap toolbox was developed as part of this project to automate this process. The second phase was data preparation and analysis. This was accomplished by building six tools within the ArcMap toolbox. These tools prepared datasets for dynamic animation in relation to time. Additionally, the client utilized Hawth's Analysis Tools to create the MCP, 95PVC, and Kernel Density feature classes. These feature classes were also manipulated for animation. All of the tools created for this project were built from scratch using Python scripting and geoprocessing methods. The third phase, exporting to KML format for Google Earth or AGX animation, was accomplished by using an existing user-built tool. This tool, Export to KML 2.5.5, was evaluated and accepted for use within this project. These three phases are addressed in detail in this document.

1.4 Audience

The intended audience for both this report and the use of the custom tools of the project is any GIS user who is involved with space-time mapping and animations. The specific intended audience for this report, the custom tools, and the full set of deliverables, are the GIS Analysts and Park Rangers at JOTR. Additionally, anyone from the NPS or other organization involved with either animal tracking or monitoring time-based data changes may benefit from this research. With the new release of ArcGIS 10, there may be a great need for the tools that have been built for anyone working with time and animation.

1.5 Overview of the Rest of this Report

The remainder of this report goes into the details of the structure of data, the tools, the outputs, and the animations. Chapter 2 discusses research of previous tortoise or wildlife projects that formed the foundation for this project. Chapter 3 discusses the systems analysis and design work and the issues that were encountered. Chapter 4 gives an overview of the database design for the feature layers. Chapter 5 describes the implementation. Chapter 6 includes analyses of the results. Lastly, the conclusion and future work are addressed in Chapter 7. The Appendix contains the hardcopy Python code written for the tools delivered in the project, as well as screenshots and output of the various tools and animations.

Chapter 2 – **Background and Literature Review**

The desert tortoises have been, and continue to be, extensively studied by many government and private organizations to improve the tortoises' survivability. With the introduction of GIS as a tool to view geographically referenced data and to perform spatial analysis on collected data, the relationships between tortoise telemetry and data reflecting their environment can be spatially analyzed to reveal new correlations.

For this study, three aspects were reviewed relevant to tortoises: their physical environment, survivability, and tracking. Additionally, vegetation was considered, but adequate vegetation data for Joshua Tree National Park (JOTR) did not exist.

2.1 Physical Environment and Survivability

In a JOTR published report, the dangers that the roads impose to the desert tortoises' survival are outlined. The report gives the status of the curbing that has been implemented along certain portions of the roads to both deter off-road vehicles and allow the tortoise to cross the road (Joshua Tree National Park, 2008). This gap in the curbing is what JOTR calls a tortoise trot (Figure 2-1).

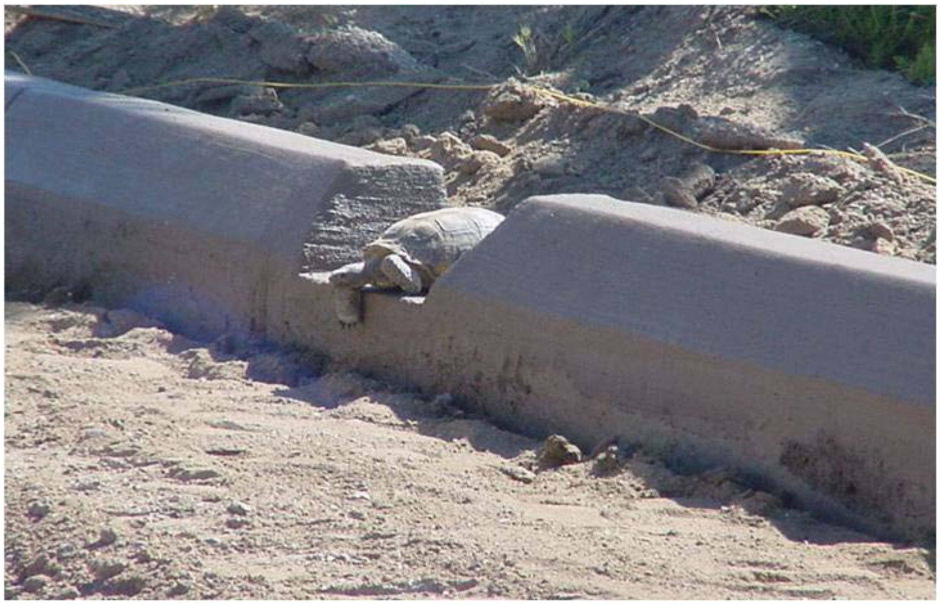


Figure 2-1: Tortoise in Tortoise Trot (asphalt not poured in road yet)

The tortoise trots allow the tortoise to get across the road by using the openings. Without the gaps, a tortoise may spend more time on the road as it may have trouble getting over the curb, or the mere existence of the curb may deter the tortoise from egression.

Boarman, Sazaki, and Jennings (1997) added further in-depth information regarding the threat of roads and highways to the tortoise populations. They also discussed the effects of the barrier fences and culverts used to minimize the danger. The study found that a considerable amount of dead tortoises were found along the roadsides in the Mojave Desert. The implementation of the fences and culverts for their study proved to help the tortoise survivablity as no tortoises were killed, and only one tortoise made it past the fence. The remaining nine torotises went away from the barriers. Of particular interest is their consideration of two different buffer areas to consider along roads when evaluating the potential cause of a discovered dead tortoise. The study suggests that tortoises found dead within 0.8km of the road may likely have been killed by a car. They also suggest that tortoises found dead up to 3.5km from the road may have been hit by a car but still have travelled a distance before die. Additionally, it is suggested that exotic plants, as well as native plants, are more abundant along roadsides. This increased vegetation may be attractive to the tortoises and should be considered.

Duda and Krzysik (1998) discussed rainfall as a factor that influences tortoise range. Peterson (1996) performed analysis on the effects of rainfall and drought on the desert tortoises. His study points to correlations between the amount of rainfall and the extent of the tortoises' range. The shortage of rainfall affects both the tortoises and vegetation that the tortoises require for food. These findings support the need to study rainfall and its ecological effect on the desert tortoise.

2.2 Movement Tracking

JOTR's 2008 Annual Report summarizes the minimum convex polygon (MCP) data collected for the year. The mapping and analysis of the MCPs used the kernel density method to estimate the probability densities. The MCP of each tortoise's home range was built by connecting the outer locations of its positions. Kernel density analysis was performed on each of the home ranges to produce 95 Percent Volume Contours (PVC) which represent where the tortoise can be found with a confidence of 95%. The Results indicate that the tortoises had a mean MCP of 76.97ha and mean 95 PVC of 29.87ha (JOTR, 2008).

Bissonette, Sherburne, and Ramsey (1994) provide further consideration regarding the analysis of animal telemetry data. Their article explains the science involved in the calculations regarding radiotelemetry data of free-ranging animals. Additonally, they suggest that general polygon data of the telemetry data may not be enough, as other environmental aspects may affect the range of the tortoises. Environmental data impacting their habitat should be considered. It is important to note that the data gathered from their animals did not include GPS-transmitted coordinates. The positions acquired were from the intersection of three bearings. With GPS transmitted coordinates, the error in the data can be minimized.

Many other studies also suggest integrating the telemetry data with other data, such as MCPs of tortoises, to further examine tortoise behavior. Duda and Krzysik (1998), in conjunction with the US Army Corps of Engineers, produced a very detailed technical report based on their radio telemetry study of the population of desert tortoises. The study included an area within the Sand Hill Training Area of the Marine Corps Ground Combat Center, Twentynine Palms, CA, as well as a comparison area at the Pinto Basin in JOTR. The study was conducted using different data, including the telemetry data, MCPs of the data, dispersal barriers, habitat fragmentation, burrow locations, vegetation, and rainfall. Analysis was performed on the data using various statistical calculations. The study also points out how GIS integrated with satellite telemetry receivers is a major stepping-stone in studying tortoise and animal behavior.

In a similar study, Riedle, Bolen, and Averill-Murray (2002) examined the tortoise population of the Florence Military Reservation (FMR) located in Arizona's Sonoran Desert. In the three-year study period, radiotelemetry data, MCPs of tortoise movement, vegetation, burrows, and the existence of eggs in the female tortoises were examined. However, the impact that roads may have on tortoise survivability was not excluded in the study. Of further interest is their use of the Animal Movement extension to ArcView®, which was used to estimate MCP home ranges.

Telemetry research on other animal types may also be beneficial to consider. Kernohan, Millspaugh, Jenks, and Naugle (1998) discuss how they used an adaptive kernel home-range estimator in a GIS environment to study the habitat of white-tailed deer. Like many other studies, they calculated ninety-five percent home-range contours by using the adaptive kernel method.

2.3 Summary

Studies of the tortoises, their habitat, and the dangers that they face, the information, and professional research reveal many similarities. Much duplication of effort exists with the main difference being that the studies were performed in a different location or at a different time.

The research has shown that the majority of the work related to tortoises uses the Minimum Convex Polygons (MCP) and 95 Percent Volume Contours (95PVC). With this project, the client provided both MCP and 95PVC data approved by JOTR naturalists. Tools already exist to create these types of features; however, there needs to be a way to visualize the many different features related to tortoise movement. Similarly, there is a lack of relating tortoise movement and their physical environment to time. Since many of the factors that may affect tortoise movement are events in time, there is much uncertainty as to what impacts tortoise movement when.

This project aimed to create something new and innovative to study tortoises. Space and time are intriguing, so the advantages of time attributes to try to animate the daily or hourly changes in the tortoises and their environment were included. Since the creation of many of the studies, new tools now exist with the capability for dynamic visualization of georeferenced features.

The project primarily focused on representing temperature and precipitation data in time to see how it affects tortoise telemetry. The studies did not address temperature, but examined vegetation. Temperature is an additional environmental factor that may contribute to changes in plant and water supply. The available vegetation data for JOTR was unfortunately unusable, as it did not identify annuals, the desert tortoises' primary source of food. Since the data were not available, the project focused on precipitation as the other environmental factor. Both temperature and precipitation affect the amount of vegetation that exists in the park, as well as affect the availability of water. The project also examined the roads and the barriers placed along the roads in JOTR.

In addition to the visualization, the project also included the development of the required tools for analysis and calculations of the various datasets.

Chapter 3 – **Systems Analysis and Design**

This chapter addresses the systems analysis and design for the project. The client, Sean Murphy of Joshua Tree National Park (JOTR), provided many datasets for the project. Through constant communication and review of progress, the client validated the direction and accompanying recommended solutions.

3.1 Problem Statement

JOTR has over four years of collected telemetry data from eighteen of their desert tortoises. Additionally, they have other data that may relate to tortoise telemetry. Since roads offer the biggest threat to the tortoises, the client needed to be able to dynamically visualize and perform analysis on the telemetry data as it relates to roads, curbs, and tortoise trots. In addition to the roads, many factors may influence the telemetry of the tortoises. It is possible that analysis of other collected data will indicate other factors that lead the tortoises towards the roads. Therefore, a dynamic visualization of the telemetry and related data were required for the analysis of the tortoises at JOTR.

3.2 Requirements Analysis

For this project, there were several functional requirements, as well as a few non-functional requirements.

3.2.1 Non-Functional Requirements

The non-functional requirements are listed with descriptions in Table 1.

Table 1. Non-Functional Requirements

Requirement	Description
Ease of Use	The tools must be easy to use and intuitive. The tools must include explanations for utilization and data entry.
Automated Processing	The tools shall automate processes that are complicated to perform manually. The tools shall minimize automated processing time to ensure efficiency and usability.
Flexibility of Tools	The tools shall be able to handle different data types and different field types for processing. The processes performed with the tools shall indicate any errors that arise from misuse of the tools and provide guidance.
Modifiable Tools	The tools shall be written in deliverable code that can be modified in the future by the client if needed.
Ease of Training	The client will be trained as part of the deliverables. The deliverables shall be organized and documentation provided to enable ease of training to further analysts by the client.

3.2.2 Functional Requirements

The client currently uses Hawth's Analysis Tools 3.27 for the majority of their analysis of tortoise data. Hawth's Analysis Tools is an extension within ArcMap designed to perform spatial analysis and functions that are not easily accessed or available within standard ArcMap (Beyer). The client would not benefit from recreation of similar tools; however, there was a need to enhance decision making by visualizing the results produced by Hawth's Tools. The recommendation to the client was to use Google Earth to dynamically depict their data and results with respect to time. The client approved of this idea. The client agreed that tools should be created to prepare the data for animation, as well as export the data into KML format for Google Earth viewing. For analysis, it was decided that tools would need to be created that would calculate maximum achieved speed per tortoise as well as build time-attributed ellipses depicting the maximum potential area covered between recorded positions. The types of data to be animated included Tortoise Positions, Minimum Convex Polygons (MCP), 95 Percent Volume Contours (95PVC), and Kernel Density polygons created by Hawth's Analysis Tools, temperature, precipitation, and road and tortoise trot curbing buffers. The functional requirements and descriptions are provided in Table 2.

Table 2. Functional Requirements

Requirement	Description
Convert a master Tortoise feature class into many sub-feature classes	The deliverables shall include a tool to automate creation of multiple feature classes from a master feature class based on tortoise name or tortoise name and year.
Create dynamic stationary event polygons for animating temperature and precipitation	The deliverables shall include a tool that will automate creation of polygon feature classes representing dynamic precipitation amounts and temperatures.
Add TimeStamp and EndTime attributes to feature classes	The deliverables shall include a tool that will automatically convert date and time fields into a standardized TimeStamp field. The tool shall also allow for automatic creation and population of an EndTime attribute based on user-selected criteria, including EndTime by interval and EndTime is start of next event.
Create road and tortoise trot buffers based on distance and time	The deliverables shall include a tool that will allow the client to create multiple road buffers and curbs with tortoise trot buffers based upon user entered parameters.
Calculate maximum achieved tortoise speeds	The deliverables shall include a tool, which will calculate the maximum achieved speed by each individual tortoise over the duration of the study as well as by year.
Create potential path ellipses between recorded positions	The deliverables shall include a tool to automatically create ellipse feature classes per tortoise that will depict the maximum potential area covered based upon the time difference between recorded positions and either the maximum achieved tortoise speed or a user-entered speed.
Export dynamic data to KML	The deliverables shall require the installation of a KML export tool that resides on the ESRI developer's network.
Capability to view static data in Google Earth, including roads, trails, and road buffers	These feature classes do not contain time attributes

Animate temporal data in Google Earth, which include tortoise movement and potential paths, curbing construction, temperature, precipitation, minimum convex polygons, and 95 percent volume contours.	The tools will be provided for the client to perform the steps necessary for creation of various KMLs.

3.3 System Design

The system design centered on the creation of new tools, as well as animating the results using Google Earth (Figure 3-1). The system architecture that existed at the client's location was discussed with the client and it did not require any hardware modifications to facilitate this delivery. However, the system setup was flexible, as the tools only require installation on the ArcGIS workstation.

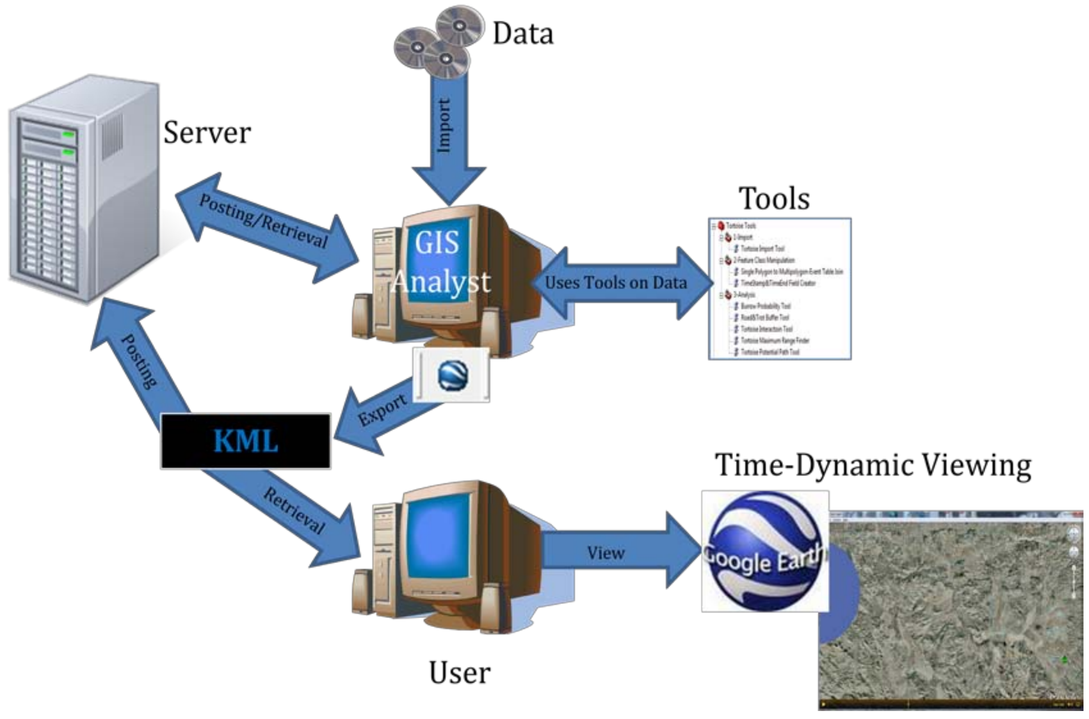


Figure 3-1: Recommended System Architecture

For software, ArcMap version 9.3, with the ArcInfo License level, was the core piece of software required. Export to KML 2.5.5 was required on the workstation as well, to export the data to KML with the necessary start and end times for dynamic Google Earth viewing. Installation of Google Earth was required to view the exported KML files dynamically. Additionally, a text editor, such as Notepad, was needed for preparing weather data to import into ArcMap.

Optionally, other software can work with the datasets created in this project. The extension to ArcMap called Tracking Analyst allows for advanced analysis and visualization of temporally dynamic data within ArcMap. With ArcGIS 10, the animation portion of Tracking Analyst is part of the baseline software, so Tracking Analyst was not required for this delivery. ArcGIS Explorer is an ESRI provided application that is free to download and similar to Google Earth. This tool has the ability to animate feature layers; however, it lacks the capability to change KML layer attributes, such as color, symbol, and elevation.

The tools of the project within ArcMap were developed with Python scripting. Python has become one of the primary coding languages within ArcMap for geoprocessing tools. The tools are part of the ArcMap Toolbox. This format allows for future editing or enhancing by the client or other user. Because of the complex functions carried out by the custom-built Python tools, ArcGIS ModelBuilder was only used for one of the eight custom tools. However, the tools can be used as needed within ModelBuilder, should a user wish to take advantage of their capabilities.

The utilization of the tools and the system setup could be enhanced to use served databases, but it was not required. Additionally, the data in the project, as well as the final products, could also be served to the client's users. This was recommended as an efficient way to distribute the data, but it was not required. The requirements of this project could exist entirely on a single workstation.

3.4 Project Plan

The planning for the project was simple conceptually, as the goals of the project were achieved through much discussion with the client. However, as all projects go, unforeseen circumstances, as well as discovery of new methods or solutions, can alter the outcome. In the first part of this section is the initial project plan. It addresses the phases involved in the project, as well as the tasks required. This will be followed by analysis of how the original plan evolved to accommodate the reality. The changes in direction will be addressed, as well as lessons learned. The initial approach followed the waterfall method of project management.

3.4.1 Initial Project Plan

This section outlines the phases, milestones, and tasks of the initial project plan.

Phase 1- Planning.

The planning phase includes acquiring data, developing simulated prototypes, studying the format of the data, accompanying the client in the field to study the tortoise, and gather information and needs from the client.

- Milestone-Data gathering
  - Task- Build Simple Dynamic Prototypes: Google Earth and AGX Models
  - Task- Acquire Tortoise Data: Tortoise position data for past five years- provided by JOTR
  - Task- Acquire Road, Trails and Camping data
  - Task- Acquire Weather data: Temperature min/max and precipitation

    - Task- Acquire Vegetation data: Vegetation layer depicting annuals
    - Task- Acquire Tortoise Trot Data: Polyline feature class of roadside barriers
- Milestone- Client Technical Exchange
    - Task- Accompany JOTR in the field: Assist at JOTR with tortoise locating
    - Task- Submit Information Needs Questionnaire to Client
    - Task- Analyze feedback from Questionnaire

Phase 2- Development

The development phase includes the development of the required tools within ArcMap, development of a symbolized ArcGIS template that includes all of the data, and utilization of the KML exporting to create dynamic KMLs of the data for utilization in Google Earth.

- Milestone- ArcGIS Template (MXT)
    - Task- Create Base MXT: This will be the foundation for the deliverables
- Milestone- Import Tools
    - Task- Build Habitat Base Map Import: Simple importing
    - Task- Build Import Roads Tool with Buffering: Import roads, and create two buffers
    - Task- Build Import Tortoise Trot Tool with buffering: Import roads and one buffer
    - Task- Build Import Tortoise Telemetry Tool: This will import the tool and modify the existing tables to add timestamp attribute
    - Tool-Build Import Weather tool: This will import the weather data and add timestamp attribute
- Milestone- Analysis Tools
    - Task- Build MCP By Year Tool: This tool will create a Minimum Convex Polygon by tortoise by year with timestamp attribute
    - Task- Build 95PVC Tool: This will build a 95 percent volume contour
    - Task- Build Tortoise Intersect Tool: This will create area intersects between all MCPs depicting density of tortoise visitation
    - Task- Build Potential Path Tool: This will calculate the farthest distance over time that each tortoise has travelled and create a potential path polygon
    - Task- Build Burrow Probability Tool: This will search the telemetry data, extract positions labeled as burrows, and look for frequented positions as being potential burrow locations
- Milestone- Symbology
    - Task- Create project symbology: This task involves establishing project symbology as agreed upon with the client
- Milestone- Export Tools
    - Task- Build KML/KMZ Export Tool: This tool will be built in Python and export all of the project feature classes to KML format, which will be able to be viewed in AGX and Google Earth

Phase 3- Testing

The testing phase includes thorough testing of all the created tools to ensure that they run successfully on the client-provided data. Time is allotted for fixing any uncovered errors.

- Milestone- Validation and Repair
    - Task- Create New blank .MXD: Testing this to ensure that custom tools can also be added to a new dataset
    - Task- Load Custom Tools and Representation
    - Task- Load Data and verify tools work: Load all project data, run tools
    - Task- Export Data and View: Export the data into KML, verify it works
    - Task- Fix errors

Phase 4- Delivery

The delivery phase includes installation of required software, delivery of all tools, project files, data, and reference material. Additionally, the delivery will include training the JOTR personnel.

- Milestone- Material
    - Task- Have Necessary Software Installed: AGX and Google Earth
    - Task- Assemble Project Deliveries: CDs, plots, instructions
    - Task- Deliver Project: visit JOTR for delivery
- Milestone- Training
    - Task- Train JOTR employees: Half to full day at JOTR on delivery day for training and demonstration

3.4.2 Changes to the Initial Project Plan

That initial project plan changed immensely since its original creation. The necessary changes to the plan were agreed upon by both the client and advisor. The many changes in direction that affected the final delivery are discussed below.

For the datasets that were to be used, vegetation data were removed from the project. After evaluation of the vegetation data, it was determined that the data does not adequately identify annuals, the main source of food for the tortoises. JOTR hopes to someday create a useful annual vegetation dataset. This is recommended in Chapter 7. The weather data presented some problems as well. Initially it was not decided upon as to which weather data to use. Upon consulting the rangers at JOTR, it was realized that temperature and precipitation data had been collected hourly in two locations in the park by JOTR. These data were used for the project. Fortunately, the tools that were developed did not require alteration to accept the new data.

In addition to the changes in data sources, there were other changes in system implementation. Originally, the ArcMap project was designed to be a normal template (MXT) so it could be shared easily without modification. While this still was an alternate way to provide the data, an ArcMap document (MXD) was deemed more flexible for the client. Additionally, it allowed for being an ArcServer service should the client want to serve an interactive view of the data.

Originally, the project consisted of many import tools; however, most of the data were stored in a file geodatabase format that is compatible with ArcMap. However, all of the feature classes provided were manually created by the client from the master database. Therefore, the only import tool that was created was to facilitate future automated creation of individual tortoise feature classes by tortoise name, by tortoise name and year, or for all tortoises by year from the master tortoise feature geodatabase. For the roads and tortoise trots curbing, rather than an import tool, a buffer tool was created to facilitate creation of multiple buffers along the roads and trot curbs. The weather import tool was also removed; however, it was replaced with a tool that creates polygons representing temperatures or precipitation in time.

Originally, the project consisted of both a 95 Percent Volume Contour (95PVC) and a Minimum Convex Polygon (MCP) creation tool. Since the client already has a tool to create these and the features have already been created and validated by the rangers, these tools were not developed in the project. Instead, the project included modified 95PVCs and MCPs to animate them through time.

Because of the complexity of the tools needed, the tortoise intersection tool and burrow probability tool were not included in the project. Instead, the tortoise maximum speed tool, the *Lost Tortoise Tool*, and the *Tortoise Potential Path Tool* were built for the project. These tools are quite useful as the calculated maximum achieved speed is used to show potential areas covered by the tortoise between recorded positions as well as enables the client to try to find a tortoise that cannot be located.

The project also originally called for the creation of a KML/KMZ export tool. Two factors contribute to this not being built. Not only would creation of a tool like this be a project in itself, but also a user-built tool already exists to perform such an export. This tool was tested and evaluated. A few bugs in the tool were discovered that have been resolved through collaboration with the developer. As a result, the developer is planning to release a new and improved version of the tool.

3.5 Summary

The final project accounted for the obstacles and changes in direction encountered through the project plan producing deliverables that met the needs for JOTR to perform analysis on the tortoises to help with their survivability. Throughout the project, various aspects were discovered that changed the direction of the project. Such aspects as data availability, data accuracy, data format, complexity of code, and availability of existing tools were investigated early on in the project to ensure minimal resources were wasted and that the project stayed in scope.

Chapter 4 – **Database Design**

This chapter outlines the database design for the project, which includes the conceptual data model, the logical data model, and the scrubbing and loading required. The input data for the project consisted of both geodatabases provided by the client, and the tabular data downloaded from the internet. In both cases, some custom-built tools built in ArcMap modified the data. Additional custom-built tools created datasets within the database that were required for analysis.

4.1 Conceptual Data Model

The design of the conceptual module for this project was straightforward. The client works with data in a schema that they have developed, and therefore, their data structure remained in the project. However, additional fields were required to be added to their data model to perform some of the analyses. These fields were automatically created by using the delivered custom-built tools. This section describes the objects within the tortoise environment and how they relate to each other (Figure 4-1).

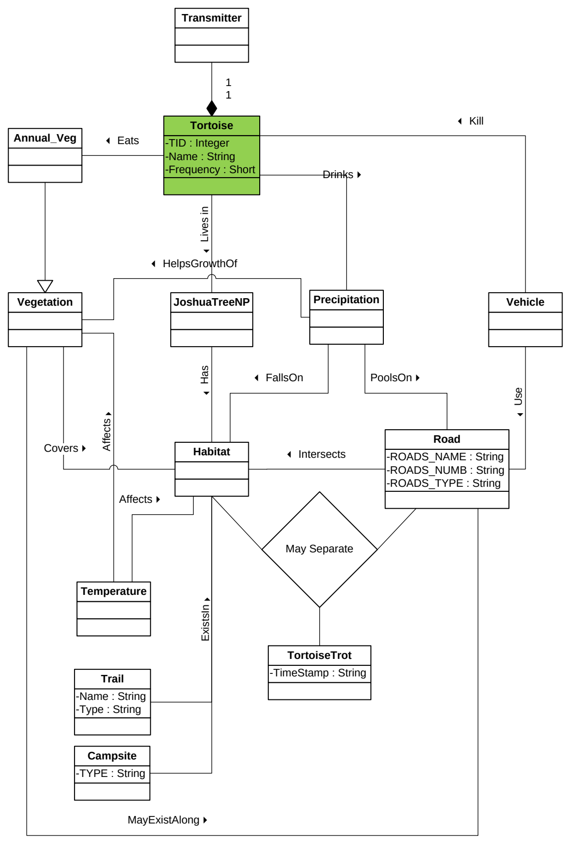


Figure 4-1: Conceptual Model – Tortoise Environment

Each tortoise has a unique name and is identifiable by means of a radio transmitter. Physical characteristics of the tortoise are on file to identify the tortoise, in case the transmitter becomes separated from the tortoise. The radio transmitters are affixed to 18 desert tortoises in JOTR. The transmitters are affixed to the shell using a two-part epoxy. Each tortoise's transmitter is on a unique frequency for identification purposes. The transmitter has an 18-month battery, which is replaced before expiration.

The tortoises eat mainly the annual vegetation that exists in JOTR. The tortoises also drink from the sparse water supply that temporarily exists after it precipitates (rains). Many roads intersect the habitat at JOTR. When it rains in JOTR, most of the precipitation absorbs into the ground, but can pool on the roads, creating an oasis for the tortoises. Because of the rain running off the roads, vegetation can thrive alongside the roads as well. The roads can be appealing for the tortoise as it can offer water and vegetation.

The problem with the tortoises' attraction to the area near roads is the threat of being hit by vehicles. Solid curbing had been installed in sections of roads to deter off-road vehicle traffic. However, if a tortoise enters the road, he may spend more time on the road. For that reason, some of the roads have tortoise trots implemented along either one or both of their sides to allow quick egression by the tortoises. JOTR is trying to get a better grasp of the effectiveness of these curbing and trots. Currently the trots only cover about one-third of the roads at JOTR.

The temperature at JOTR affects vegetation growth and the habitat of the tortoise. As a result, it is possible there will be a correlation between the tortoise movement and temperature. Within JOTR, there are also scattered campsites and many trails. At this point, there are no known associations between the tortoise behavior and these features.

4.2 Logical Data Model

The client used ArcGIS file geodatabases to store most of their data and provided them in that format. Other data were provided in personal geodatabase and shapefile format. In general, the given schema of their geodatabases and shapefiles were not changed; however, new fields were created and populated using the delivered customized tools. These new fields enable dynamic visualization within the various interfaces used for animation. It is noted that JOTR should consider revising the format of certain collection fields, such as time, for easier utilization. For example, time is currently collected as a Double field rather than as a Date field; however, the delivered tools were created to handle either case.

The geodatabases that were provided by the client include Tortoise_Telemetry.gbd, Exotics.gdb, and JOTR_AISVegLayer.mdb. The last two databases both relate to vegetation and were not used for this project, as they did not identify annual vegetation. Tortoises primarily eat annual vegetation. The shapefiles provided by the client included curb lines (tortoise trot and standard curbs), camping (campsites, visitor and nature centers), no camping (Areas where camping is not allowed), roads, trails, and wilderness areas.

For the weather data, data were extracted from the NPS Gaseous Pollutant and Meteorological Data website. Four files were created as a result of querying by location and by the years covering this study, including: Cottonwood Canyon ambient temperature (aspirated), Cottonwood Canyon precipitation, Black Rock ambient temperature (aspirated), and Black Rock precipitation. Aspirated means that air is constantly forced over an enclosed sensor; non-aspirated is when the air is free to flow around an open sensor.

4.2.1 Tortoise Geodatabase

This geodatabase includes four feature datasets: Tortoise_Locations, Minimum_Convex_Polygons, Percent_Volume_Contours, and Kernel_by_Year. In addition, TimeStamp and EndTime attributes were added where appropriate using the custom-built tools. Additionally, new feature classes and tables were created by using the tools. This included a tortoise potential path feature class for each tortoise and the tables showing maximum speed achieved by tortoise and maximum speed achieved by tortoise per year.

4.2.1.1 Tortoise Locations Feature Dataset

This feature dataset contains 69 feature classes. Figure 4-2 shows a sample of the All_Tortoises feature class and Elizabeth feature class. One of them, the All_Tortoises feature class, contains all of the data for the 18 tortoises for five years. Additionally, there are four All_Tortoises by year, 18 feature classes by tortoise, and 46 tortoise by year feature classes (a total of 68 feature classes). Only the All_Tortoises feature class is needed, since the other 68 feature classes are duplications of information that exist within the All_Tortoises feature class. It is unnecessary to permanently keep these many additional feature classes; however, they are needed to export KML files by tortoise and by year. Originally, the client had to manually create the 68 feature classes.

In addition, two new attribute fields, TimeStamp and EndTime, were automatically added and populated using the custom-built tools. These fields are of type String, using the International Organization for Standardization (ISO) approved date format, YYYY-MM-DD HH:MM:SS.

This feature dataset also includes the newly created tortoise potential path feature classes. These feature classes also include TimeStamp and TimeEnd attribute fields.

In general, these feature classes are used to represent tortoise movement for all of the tracked tortoises.

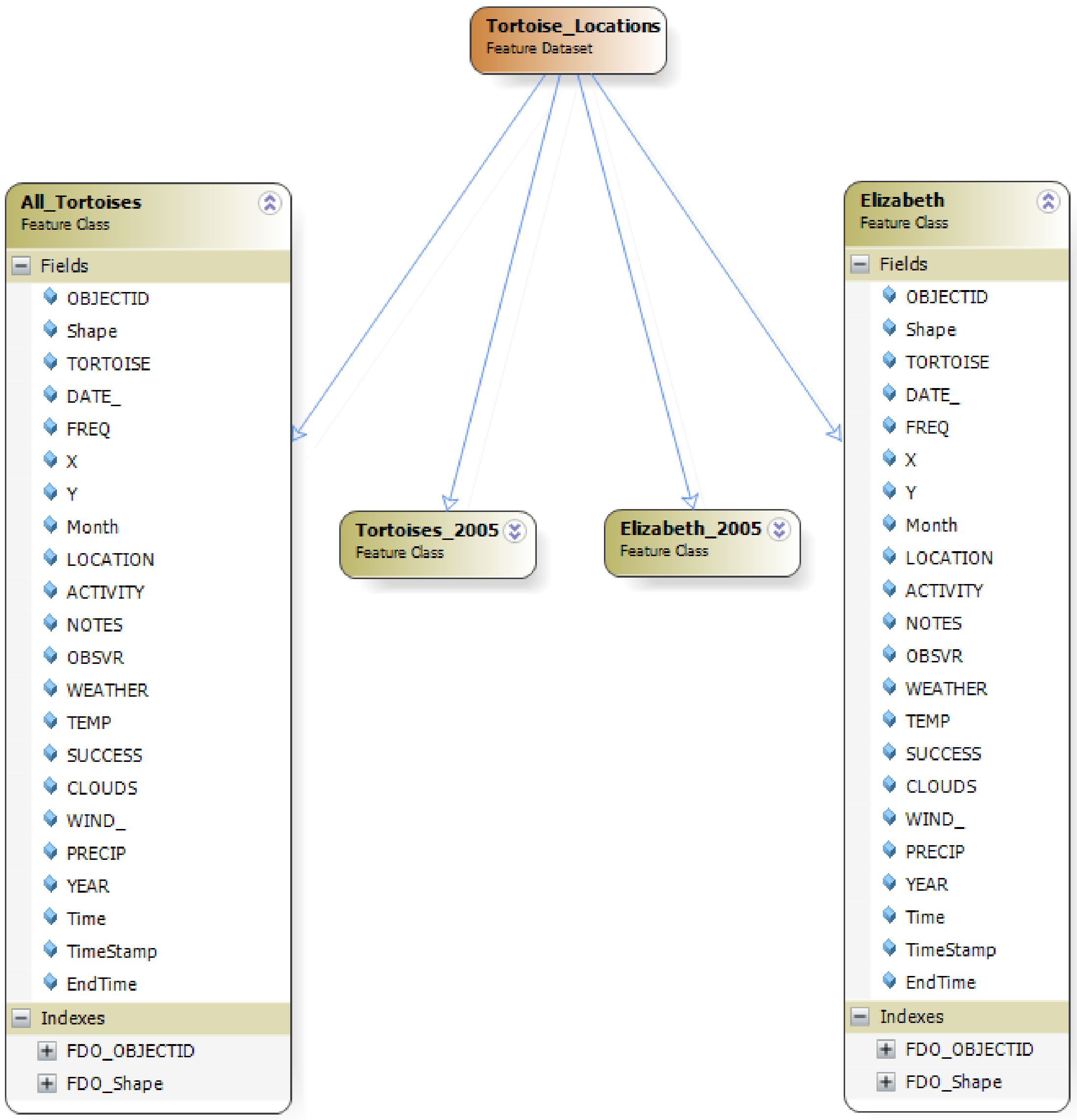


Figure 4-2: Tortoise Locations Feature Dataset Schema

4.2.1.2 Minimum_Convex_Polygons Feature Dataset

This feature dataset contains 85 feature classes. All except one represents a Minimum Convex Polygon (MCP) for a tortoise for either summer or winter of a given year. The other feature class was an All_Tortoises feature class; however, it only contains the polygons without the year, season, or tortoise name identified in the attributes. Additionally, this feature class includes the name of the tortoise and season within the feature class name and not within the table. For this project, it would have been ideal to have an All_Tortoises feature class that contained the tortoise name and time attributes. As a result, the individual feature classes were manually merged into one feature class and manually populated with new attributes: Name, TimeStamp, and TimeEnd. This was required to recreate the sub-feature classes when using the delivered import tool.

Figure 4-3 shows a sample of one of the feature classes within the dataset. Additionally, the dataset contains one feature class for both winter and summer for each year per tortoise for the years 2005 through 2008. They have the same schema as the depicted feature class MCP_Elizabeth_Summer_2006. The MCP_All_Tortoises feature class has the same schema as well. The Name, TimeStamp, and EndTime attributes were manually added.

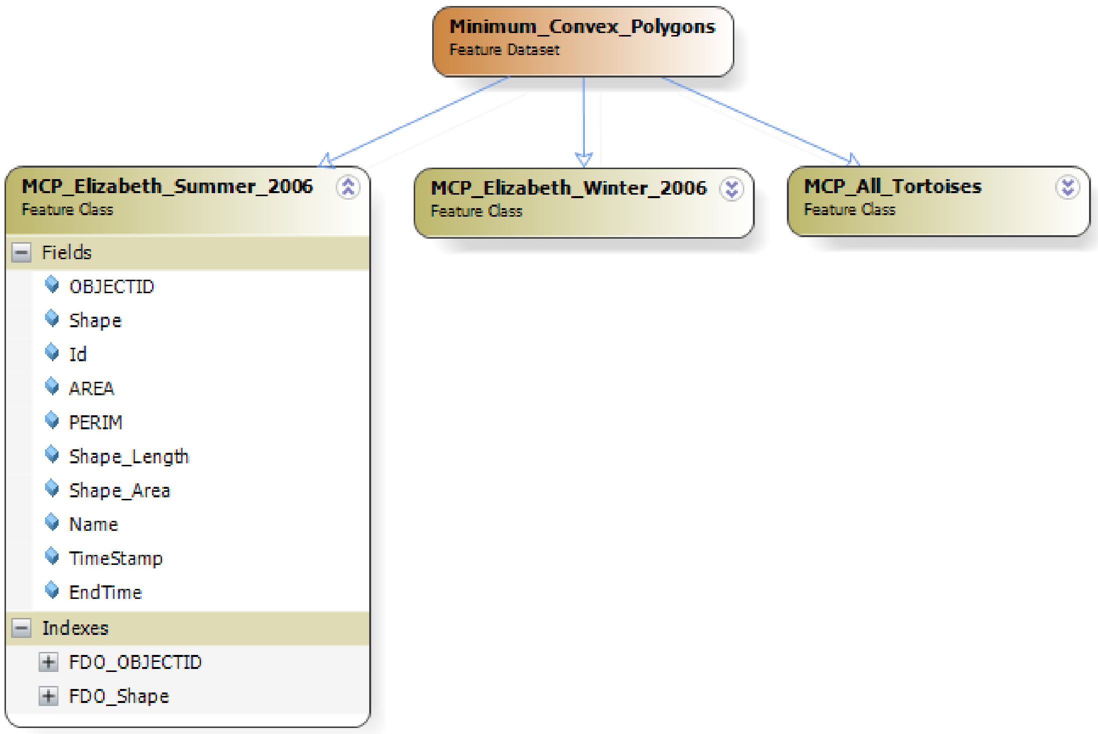


Figure 4-3: Minimum Convex Polygons Feature Dataset Schema

4.2.1.3 Percent_Volume_Contours Feature Dataset

This feature dataset contains 18 feature classes, each representing the 95 percent volume contours for each tortoise over the entire period of the study. Since they cover the entire period of the study, they were not animated and therefore, they did not have to be manipulated.

Figure 4-4 shows a sample of one of the feature classes within the dataset. Additionally, the dataset contains one feature class for each of the 17 other tortoises. They have the same schema as the depicted feature class elizabeth_poly. The Name attribute was manually added.

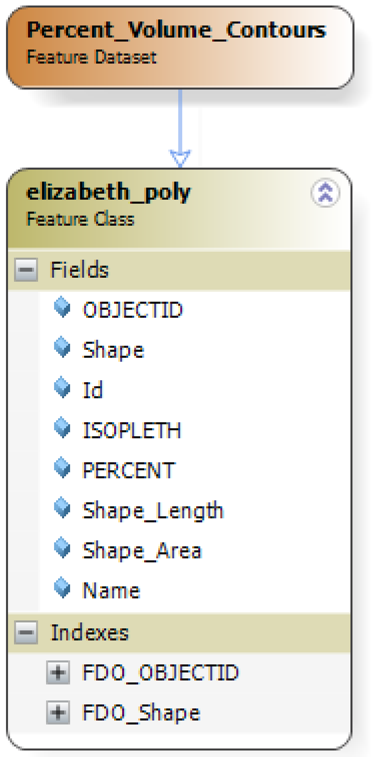


Figure 4-4: 95 Percent Volume Contour Polygon Feature Dataset Schema

4.2.1.4 Kernel_by_Year Feature Dataset

This feature dataset includes 44 feature classes, each representing a tortoise's Kernel Density polygon for a given year. The name of the tortoise and season are contained within the feature class name and not as attributes. For this project, the individual feature classes were manually merged into one feature class and the new attributes, Name, TimeStamp, and TimeEnd were populated. This was required to recreate the sub-feature classes when using the delivered import tool.

Figure 4-5 shows a sample of one of the feature classes within the dataset. Additionally, the dataset contains one feature class for each year per tortoise for the years 2005 through 2008. They have the same schema as the depicted feature class eliz05_poly. The Name, TimeStamp, and EndTime attributes were manually added.

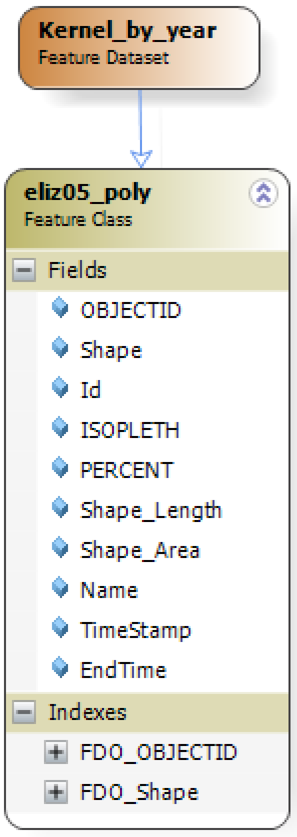


Figure 4-5: 95 Kernel Density by Year Polygon Feature Dataset Schema

4.2.2 Joshua Tree Features Geodatabase

These data were originally provided as separate shapefiles. They were compiled into a single geodatabase, JOTR.gdb, which includes mostly static features within the park, including camping areas, no camping areas, park limits, trails, wilderness areas, roads, and tortoise trots. Additionally, new feature classes were added to the geodatabase that have been automatically created from the utilization of the tools. These feature classes include Road_Buffer_Close, Road_Buffer_Far, and Tortoise_Trot_Buffer.

4.2.2.1 Foundation Feature Dataset

This feature dataset includes the shapefiles Camping, NoCampingAreas, limits, Trails and WildernessArea. The Camping feature class also includes facilities within JOTR, including visitor centers, nature centers, and park headquarters (Figure 4-6).

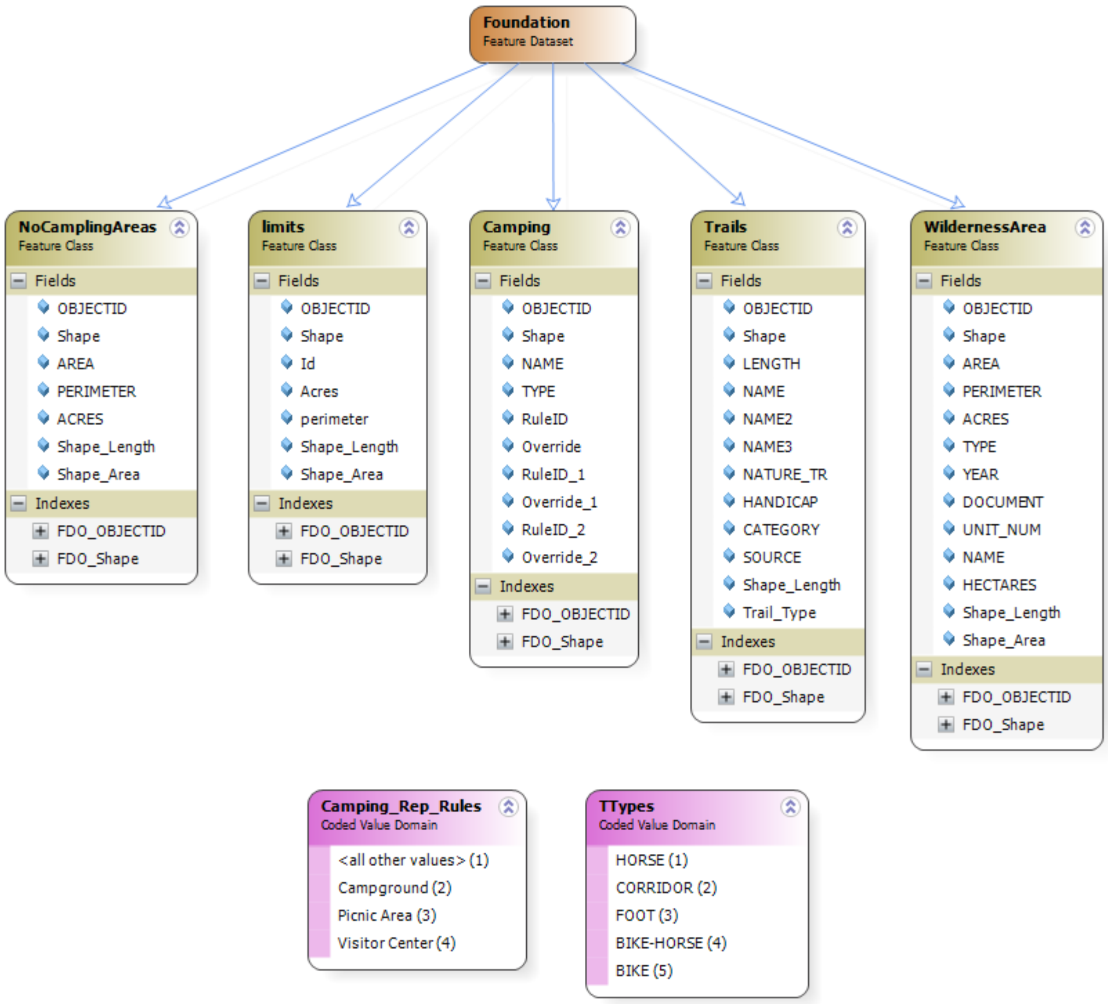


Figure 4-6: JOTR – Foundation Feature Dataset Schema

4.2.2.2 Roads Feature Dataset

This feature dataset includes supplied shapefiles TortoiseTrots and Roads. Additionally, it includes the feature classes Road_Buffer_Close, Road_Buffer_Far, and Tortoise_Trot_Buffer that were created using the delivered tools (Figure 4-7).

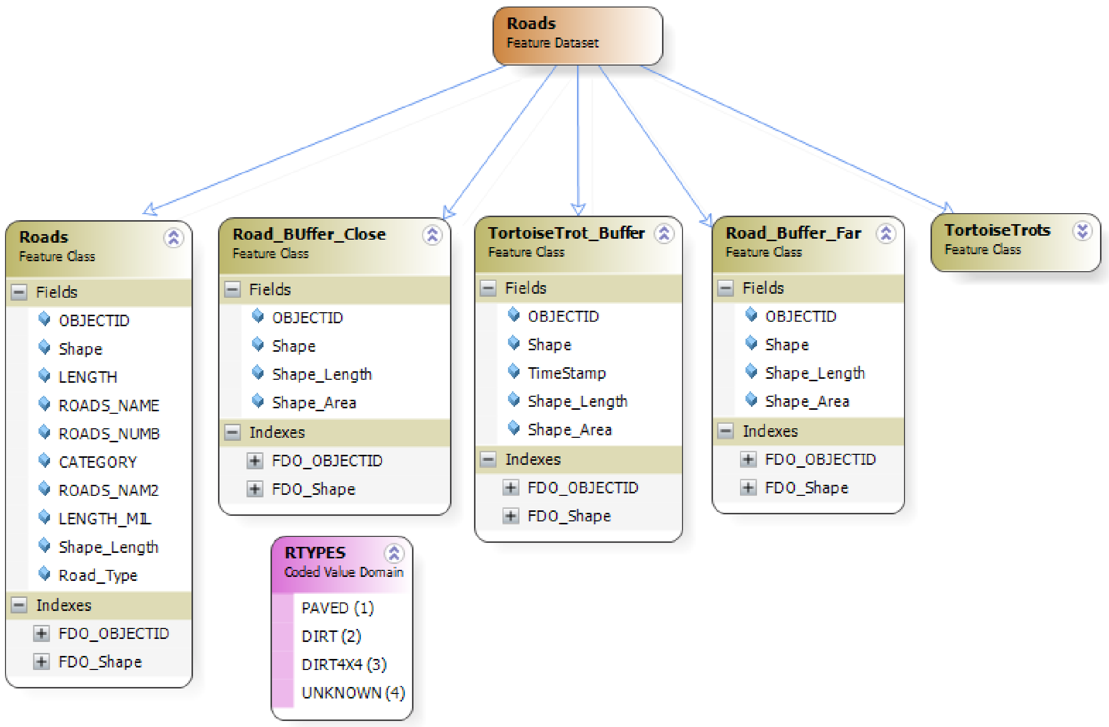


Figure 4-7: JOTR – Roads Feature Dataset Schema

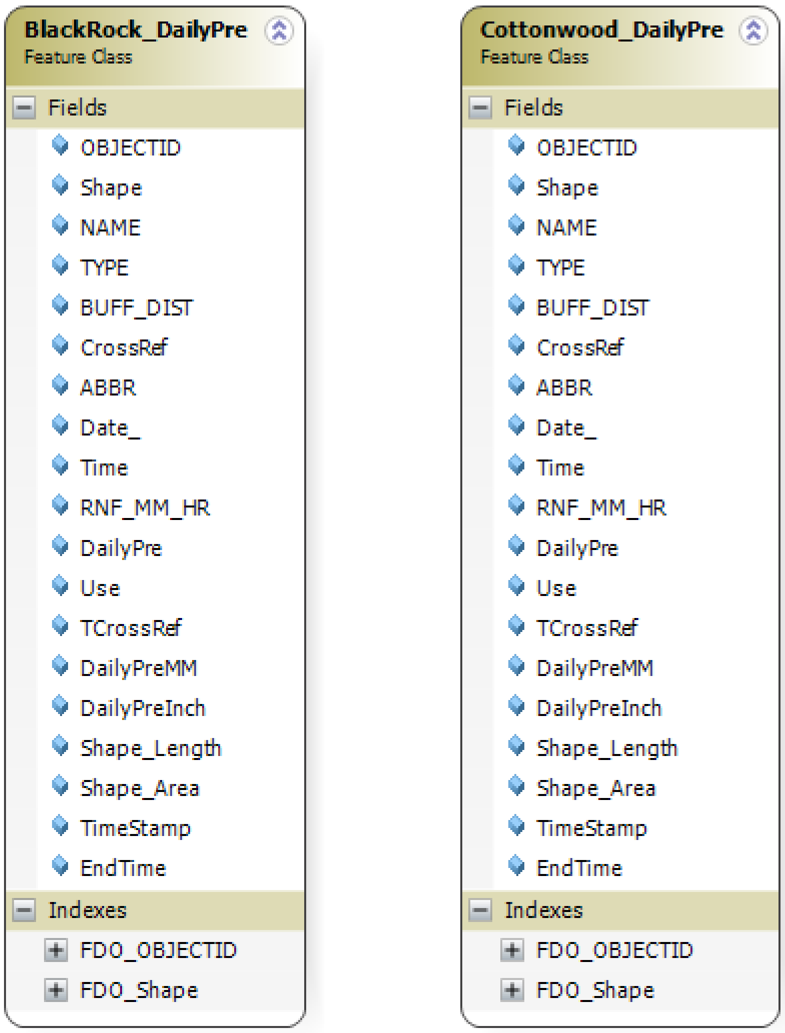
The TortoiseTrots Feature Class has an extensive number of attributes; therefore, the feature class has not been expanded within the diagram. This feature class was not modified for this project.

4.2.3 Weather Geodatabase

This geodatabase was created to contain the feature classes created from the weather data obtained from the Black Rock Nature Center and the Cottonwood Canyon Visitor Center in the park, which specifically contains daily precipitation, daily average temperature, daily high temperature, and daily low temperature. Eight feature classes are included: BlackRock_DailyPrecip, BlackRock_AveTemp, BlackRock_HighTemps, BlackRock_LowTemps, CottonwoodCanyon_DailyPrecip, CottonwoodCanyon_AveTemp, CottonwoodCanyon _HighTemps, and CottonwoodCanyon _LowTemps.

4.2.3.1 Daily Cumulative Precipitation Feature Class

Two feature classes were designed to include stationary polygons attributed with daily precipitation totals by date (Figure 4-8). Similar to other feature classes, timestamp fields are necessary.



BlackRock_DailyPre
Feature Class

Fields
- OBJECTID
- Shape
- NAME
- TYPE
- BUFF_DIST
- CrossRef
- ABBR
- Date_
- Time
- RNF_MM_HR
- DailyPre
- Use
- TCrossRef
- DailyPreMM
- DailyPreInch
- Shape_Length
- Shape_Area
- TimeStamp
- EndTime

Indexes
- FDO_OBJECTID
- FDO_Shape

Cottonwood_DailyPre
Feature Class

Fields
- OBJECTID
- Shape
- NAME
- TYPE
- BUFF_DIST
- CrossRef
- ABBR
- Date_
- Time
- RNF_MM_HR
- DailyPre
- Use
- TCrossRef
- DailyPreMM
- DailyPreInch
- Shape_Length
- Shape_Area
- TimeStamp
- EndTime

Indexes
- FDO_OBJECTID
- FDO_Shape

Figure 4-8: Weather – Precipitation Feature Class Schema

4.2.3.2 Average Daily Temperature Feature Class

These feature classes include stationary polygons attributed with the average daily temperature by date (Figure 4-9).

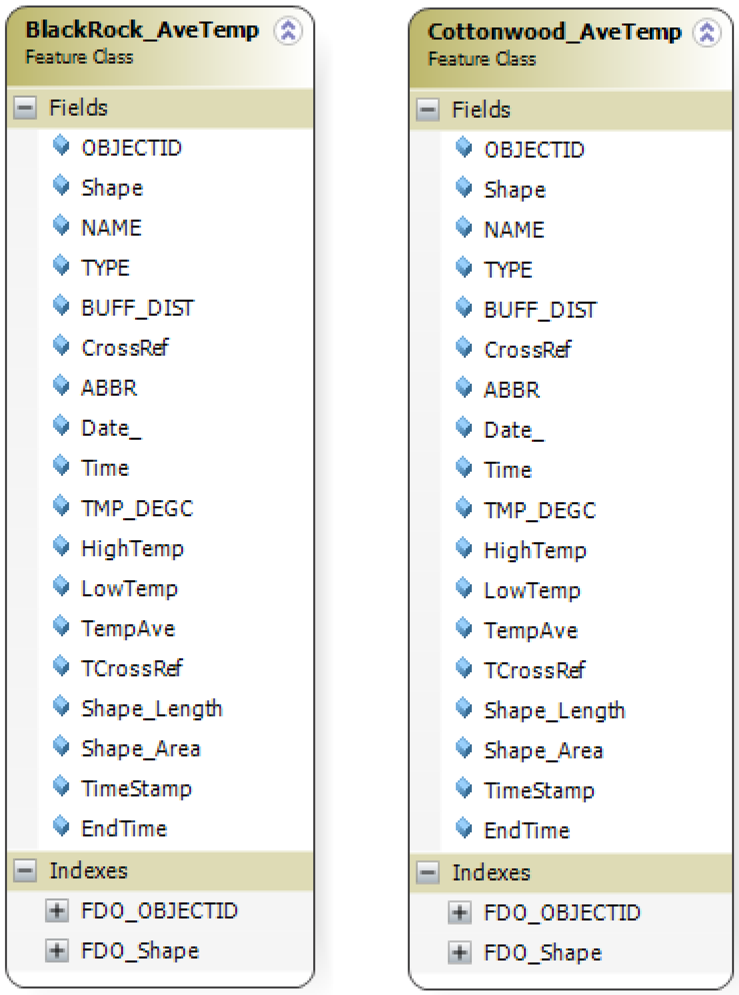


Figure 4-9: Weather – Average Daily Temperature Feature Class Schema

4.2.3.3 Daily High and Daily Low Temperature Feature Classes

These feature classes include stationary polygons attributed with the daily high temperature by date, as well as feature classes attributed with daily low temperature (Figure 4-10).

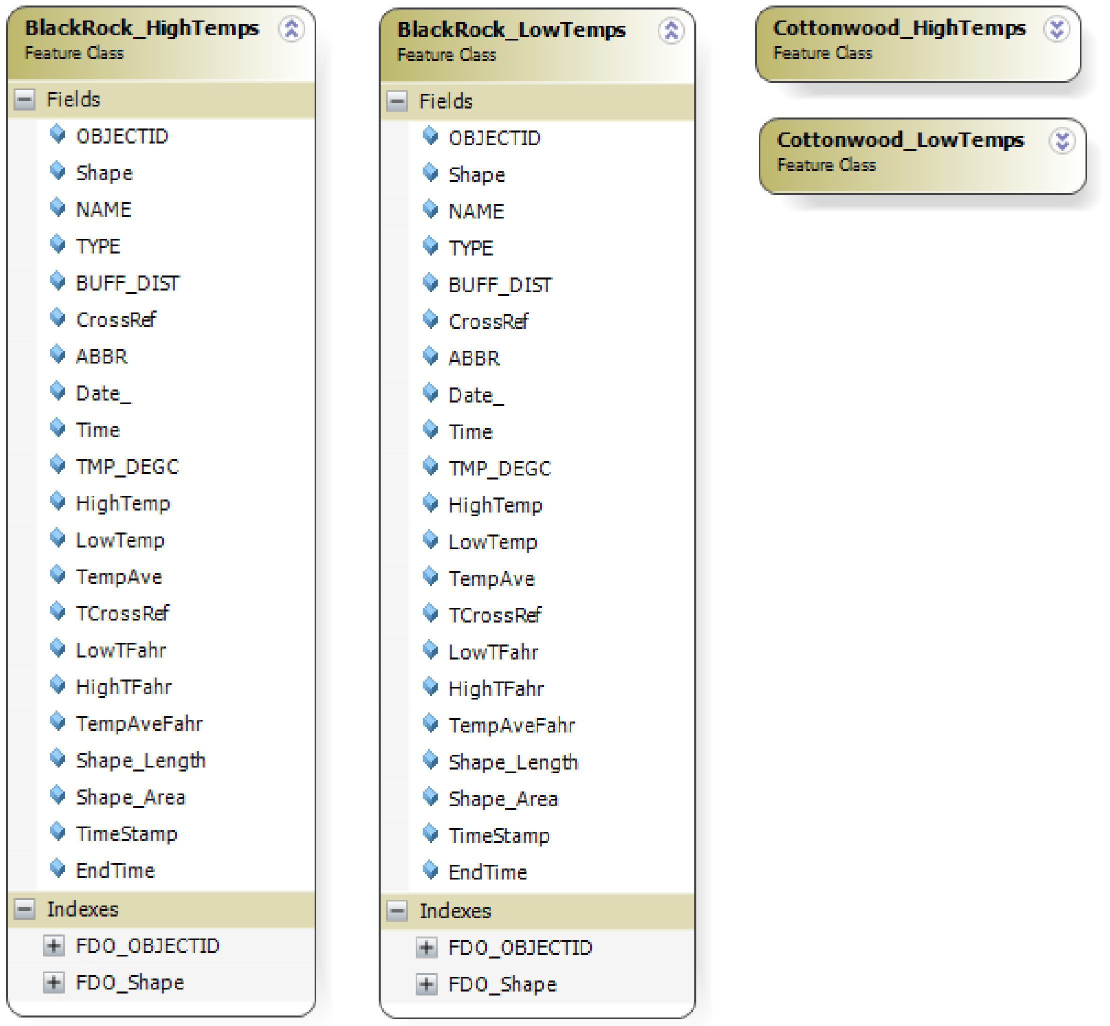


Figure 4-10: Weather – Daily High and Daily Low Feature Class Schema

4.2.4 Summary of Data Sources

For the project, JOTR provided most of the datasets directly. For the temperature and precipitation data, JOTR data were extracted from the Air Resource Specialists, Inc. website. The data sources are listed in Table 3.

Table 3. Summary of Data Sources

Data	Description	Processing
All_Tortoises JOTR	A single feature class that includes all recorded tortoise positions	A delivered tool can be used to automatically create the Tortoise and Tortoise_byYear feature classes. Another tool was used to automatically create and populate TimeStamp and EndTime fields.
Tortoise JOTR	18 feature classes, one for all positions of each tortoise	A delivered tool was used to automatically create and populate TimeStamp and EndTime fields.
Tortoise_byYear JOTR	69 feature classes, one for each tortoise in each year	A delivered tool was used to automatically create and populate TimeStamp and EndTime fields.
MCP_All_Tortoises JOTR	A single feature Class that includes Minimum Convex Polygons for each tortoise	No processing or utilization of this feature class performed. This feature class was rebuilt to include Tortoise Name and Year.
MCP_byTortoise_Winter_byYear JOTR	44 feature classes, by tortoise, by year and by winter.	These feature classes was used to rebuild a new All_Tortoises feature class that is complete. The Name, TimeStamp, and EndTime fields had to be manually created and populated.
MCP_byTortoise_Summer_byYear JOTR	44 feature classes, by tortoise, by year and by winter.	These feature classes was used to rebuild a new All_Tortoises feature class that is complete. The Name, TimeStamp, and EndTime fields had to be manually created and populated.

PVC: Tortoise_poly JOTR	18 feature classes of 95 Percent Volume Contours by tortoise.	The Name field had to be manually added and populated with the tortoise name.
Kernel by Year: TortoiseYear_poly JOTR	44 feature classes of Kernel Density by tortoise and by year.	The Name, TimeStamp, and EndTime fields had to be manually created and populated.
Trails JOTR	A single feature class represents the trails in JOTR. The trails are classified by trail use type.	The data are represented by the trail use type in ArcMap.
Camping JOTR	A single feature class represents camping areas, picnic areas, and visitor centers in JOTR.	The data are represented by the type in ArcMap.
NoCampingAreas JOTR	A single feature class represents the areas where camping is not prohibited in JOTR.	This is a minor feature class, which is included, but is optional to display.
WildernessAreas JOTR	A single feature class represents the wilderness areas of JOTR.	This is a minor feature class, which is included, but is optional to display.
Limits JOTR	A single feature class represents the limit of JOTR.	This is included in the deliverables with no manipulation.
Roads JOTR	A single feature class representing the roads, and categorized by Road Type.	The roads were used to create road buffers that were included in the project.
Curbing JOTR	A single feature class contains both standard curbing and tortoise trot curbing.	The curbs are categorized and include their implementation date. These were used to create tortoise trot buffers.

Black Rock Precipitation JOTR/ Air Resource Specialists, Inc	A single table created from importing comma-delimited precipitation data.	This table is used by the delivered tools to create the time-attributed polygon feature class representing daily precipitation.
Black Rock Temperatures JOTR/ Air Resource Specialists, Inc	A single table created from importing comma-delimited temperature data.	This table is used by the delivered tools to create the time-attributed polygon feature classes representing daily high, daily low, and daily average temperatures.
Cottonwood Canyon Precipitation JOTR/ Air Resource Specialists, Inc	A single table created from importing comma-delimited precipitation data.	This table is used by the delivered tools to create the time-attributed polygon feature class representing daily precipitation.
Cottonwood Canyon Temperatures JOTR/ Air Resource Specialists, Inc	A single table created from importing comma-delimited temperature data.	This table is used by the delivered tools to create the time-attributed polygon feature classes representing daily high, daily low, and daily average temperatures.

4.3 Data Collection Methods

All of the data used for the project were collected by the National Park Service at JOTR. The data were provided in file geodatabase, shapefile, or comma separated value (CSV) formats. To gain an understanding of the process involved in collecting tortoise position data and to gain insight into the tortoises and their environment, the client was accompanied in the field to observe the process as well as locate and identify a tortoise. This section describes the objects related to how the tortoise, vegetation, and weather data were collected (Figure 4-11). It is noted that vegetation data were removed from this project after conceptual design, as adequate delineation between annuals and perennials was not available within the data.

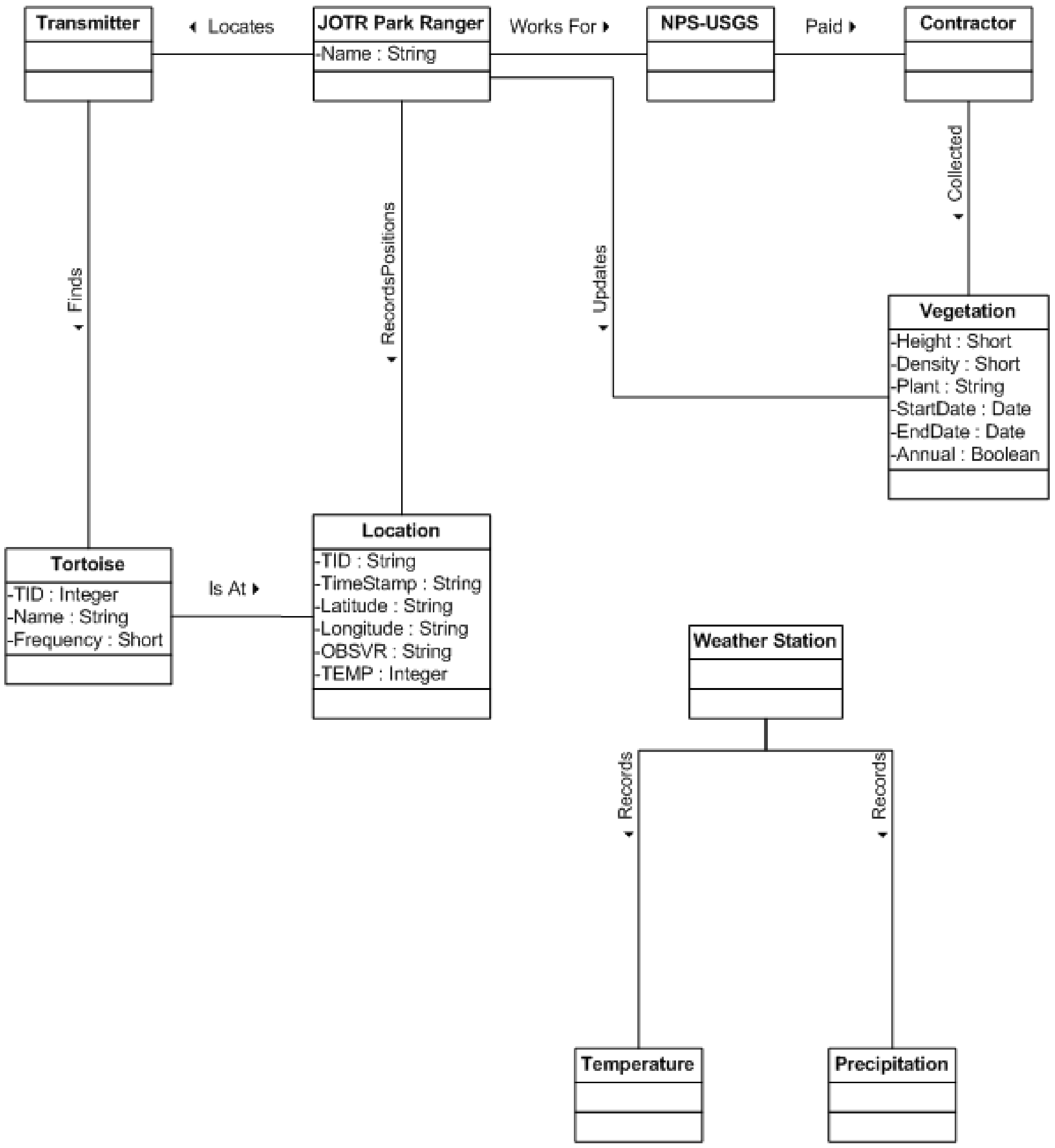


Figure 4-11: Data Collection Process

Each tortoise is fitted with a transmitter that has a unique frequency (Figure 4-12). Each tortoise was located by the JOTR park ranger by tuning a hand-held radio receiver, equipped with a directional antenna, to the frequency and walking towards the strongest return signal received (Figures 4-12, 4-13). The park ranger attempts to locate tortoises three to five times per week. Once a tortoise is located, the ranger uses a Magellan Mobile GPS unit to record the location of the tortoise along with additional metadata (Figure 4-14).

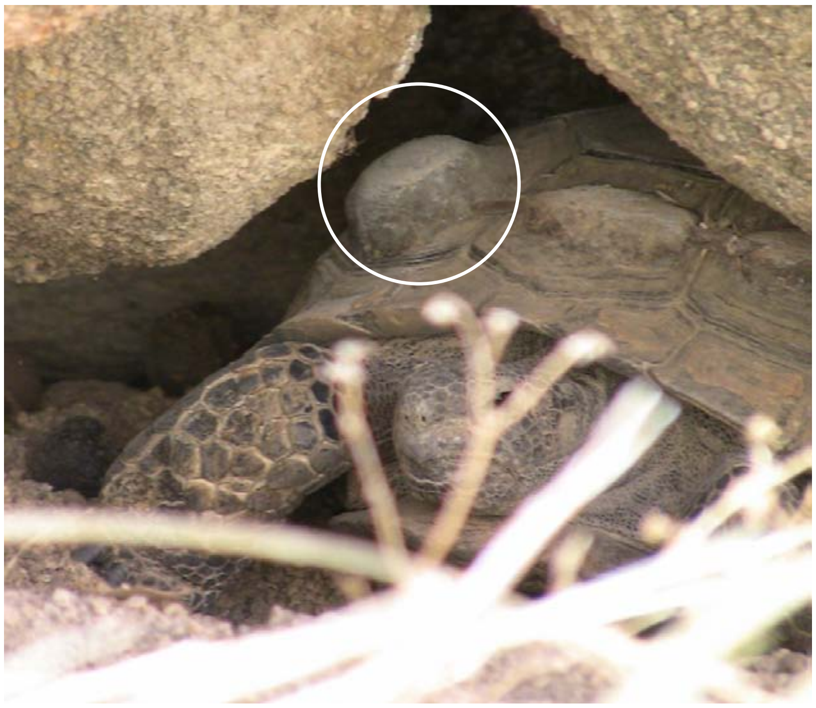


Figure 4-12: Desert Tortoise at JOTR with Mounted Transmitter Visible

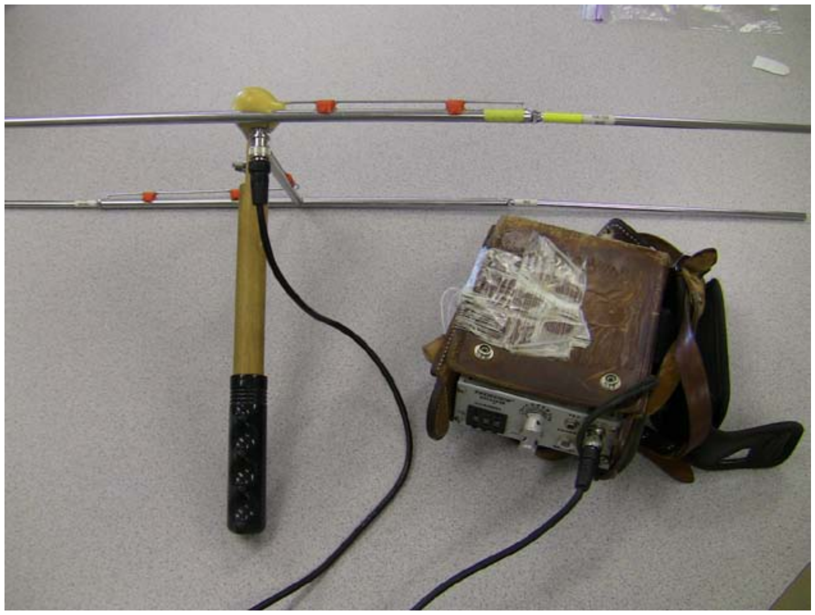


Figure 4-13: Directional Antenna and Radio Receiver

Figure 4-14: Magellan® Mobile Mapper 6 with ArcPad®

The National Park Service (NPS) and the U.S. Geological Survey (USGS) contracted Aerial Information Systems (AIS) to use photo interpretation to identify vegetation types and create attributed polygons depicting the vegetation area. Since the dataset was developed in the mid 1990s, JOTR park rangers have modified and updated it as needed. The dataset is improved continually. Currently, however, it is generalized. Annuals are only depicted if they dominate a polygon. Since there was little detail in the dataset concerning annuals, vegetation analysis was removed from the project.

JOTR has weather stations at the Black Rock Nature Center and the Cottonwood Canyon Visitor Center that collect weather data. The NPS contracted Air Resource Specialists, Inc. to house the collected weather database for the NPS. These data are available through their Gaseous Pollutant and Meteorological Data website: http://ard-request.air-resource.com/. Air Resource Specialists, Inc. operates this website under contract with the National Park Service. The downloaded data for this project include hourly precipitation and hourly temperature. These data can be downloaded as tab-delimited text tables.

4.4 Data Scrubbing and Loading

The datasets used in this project required some organization, formatting, and modification to be used.

For the project, the geodatabases and feature classes all had to be in an NAD 1983 UTM Zone 11N Projection, as it is required by JOTR. This format is additionally useful for calculating distance measurements in meters with the geoprocessing tools that were delivered. Feature classes were reprojected as necessary.

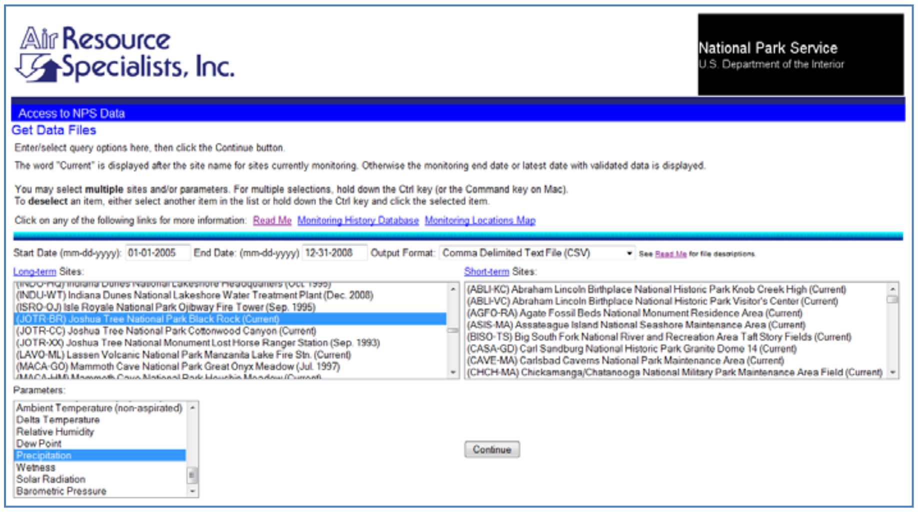
Both provided datasets and downloaded data were manipulated within the geodatabases. For delivered data, the MCP, 95PVC, and Kernel Density datasets needed extensive manipulation. Initially, the master feature class for each had to be manually created that contained all of the necessary attribution. Then, several custom tools were built to automate further data preparation. Additionally, the weather data extracted from the web needed conversion to bring the data into the created weather geodatabase.

The Minimum Convex Polygon (MCP) dataset included 85 feature classes that only included the tortoise name, year, and either Winter or Summer in the individual feature class file names. All of these feature classes had to have the Name, TimeStamp, and EndTime attribute fields manually added and populated. Once these were populated, they were additionally migrated into the MCP_All_Tortoises feature class. This enabled dynamic display of all the MCPs.

The 95 Percent Volume Contour (95PVC) dataset covered the entire time frame of the data, so TimeStamp and EndTime are not applicable fields. However, the tortoise name was only part of the file name and not collected as an attribute. The Name field had to be manually added to the 18 feature classes. The datasets were then brought into a single PVC_All_Tortoises feature class should the analyst want to turn all of the PVCs on at once.

The Kernel Density dataset included 44 feature classes, one for each tortoise for each year. Again, these feature classes only included the tortoise name and year within their file name. All of these feature classes had to have the Name, TimeStamp, and EndTime attribute fields manually added and populated. Once these were populated, they were additionally all migrated into a Kernel_All_Tortoises feature class. This enables dynamic display of all the kernel density polygons.

The weather data for the time of the project from 2005 through 2008 were queried and downloaded from the NPS Gaseous Pollutant and Meteorological Data website. The downloaded data included the precipitation and temperature (aspirated) data for the Black Rock Nature Center (Figure 4-15).


Air Resource Specialists, Inc.
National Park Service
U.S. Department of the Interior
Access to NPS Data
Get Data Files

Enter/select query options here, then click the Continue button.

The word "Current" is displayed after the site name for sites currently monitoring. Otherwise the monitoring end date or latest date with validated data is displayed.

You may select **multiple** sites and/or parameters. For multiple selections, hold down the Ctrl key (or the Command key on Mac).
To **deselect** an item, either select another item in the list or hold down the Ctrl key and click the selected item.

Click on any of the following links for more information: Read Me Monitoring History Database Monitoring Locations Map

Start Date (mm-dd-yyyy): 01-01-2005 End Date: (mm-dd-yyyy) 12-31-2008 Output Format: Comma Delimited Text File (CSV) See Read Me for file descriptions.

Long-term Sites:
(INDU-WT) Indiana Dunes National Lakeshore Water Treatment Plant (Dec. 2008)
(ISRO-OJ) Isle Royale National Park Ojibway Fire Tower (Sep. 1995)
(JOTR-BR) Joshua Tree National Park Black Rock (Current)
(JOTR-CC) Joshua Tree National Park Cottonwood Canyon (Current)
(JOTR-XX) Joshua Tree National Monument Lost Horse Ranger Station (Sep. 1993)
(LAVO-ML) Lassen Volcanic National Park Manzanita Lake Fire Stn. (Current)
(MACA-GO) Mammoth Cave National Park Great Onyx Meadow (Jul. 1997)

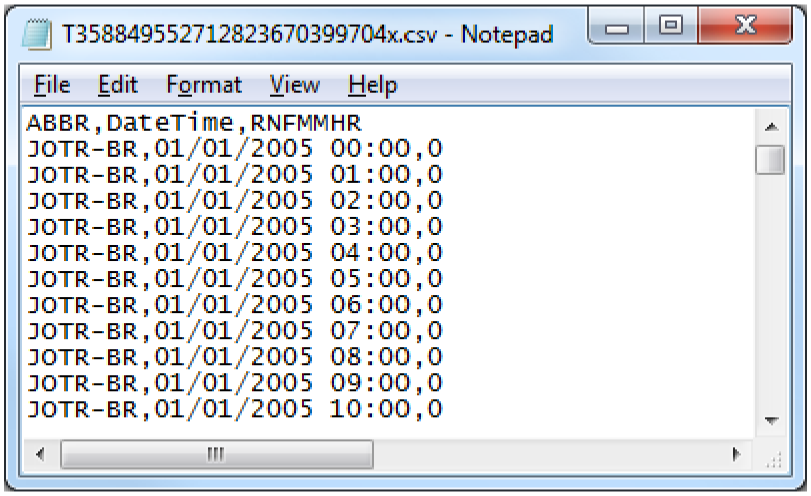
Short-term Sites:
(ABLI-KC) Abraham Lincoln Birthplace National Historic Park Knob Creek High (Current)
(ABLI-VC) Abraham Lincoln Birthplace National Historic Park Visitor's Center (Current)
(AGFO-RA) Agate Fossil Beds National Monument Residence Area (Current)
(ASIS-MA) Assateague Island National Seashore Maintenance Area (Current)
(BISO-TS) Big South Fork National River and Recreation Area Taft Story Fields (Current)
(CASA-GD) Carl Sandburg National Historic Park Granite Dome 14 (Current)
(CAVE-MA) Carlsbad Caverns National Park Maintenance Area (Current)
(CHCH-MA) Chickamanga/Chatanooga National Military Park Maintenance Area Field (Current)

Parameters:
Ambient Temperature (non-aspirated)
Delta Temperature
Relative Humidity
Dew Point
Precipitation
Wetness
Solar Radiation
Barometric Pressure

Continue

Figure 4-15: Air Resource Specialists, Inc. Website for NPS Weather Data

Clicking continue lead to the next screen to generate the comma separated value file (CSV) that became displayed on the screen. On the website output screen, a right-mouse click followed by Save Page As, allowed the CSV file to be saved. This file needed to be modified prior to bringing it into ArcMap. Using Notepad, all of the header information was deleted, except the field row above the data. In addition, the spaces were removed from the field names (Figure 4-16). The file was then ready to add to ArcMap as a simple table. This process was replicated for the Temperature (aspirated).



```
ABBR,DateTime,RNFMMHR
JOTR-BR,01/01/2005 00:00,0
JOTR-BR,01/01/2005 01:00,0
JOTR-BR,01/01/2005 02:00,0
JOTR-BR,01/01/2005 03:00,0
JOTR-BR,01/01/2005 04:00,0
JOTR-BR,01/01/2005 05:00,0
JOTR-BR,01/01/2005 06:00,0
JOTR-BR,01/01/2005 07:00,0
JOTR-BR,01/01/2005 08:00,0
JOTR-BR,01/01/2005 09:00,0
JOTR-BR,01/01/2005 10:00,0
```

Figure 4-16: Correctly Formatted CSV File in Notepad

4.5 Summary

In summary, all data provided for this project were collected by JOTR and, in general, required little scrubbing. The scrubbed data were modified as needed by custom-built tools to both add attribute fields to the feature classes and to create new feature classes. These modifications were necessary to enable the data to be used for dynamic animations in ArcMap 10 as well as Google Earth. The attributes that were added included TimeStamp and EndTime, as well as attributes that were created to support generation of those fields.

The conceptual model described the relationships amongst the various components of the tortoises and their environment. This provided the foundation for developing the logical model. The logical model depicts the three geodatabases of the project: tortoise locations, JOTR features, and weather. There are many data sources used in this project, as well as many feature classes that were delivered to the client. These delivered feature classes were the result of running the various custom-built Arc Toolbox tools. These tools were also provided to the client.

Chapter 5 – **Implementation**

This project created a custom ArcGIS toolbox to prepare Joshua Tree National Park (JOTR) tortoise and habitat data for time-based animations using Google Earth, ArcGIS Explorer (AGX), and ArcMap. This chapter describes the seven Python-based tools and one ModelBuilder tool, in the order of their creation (Figure 5-1).

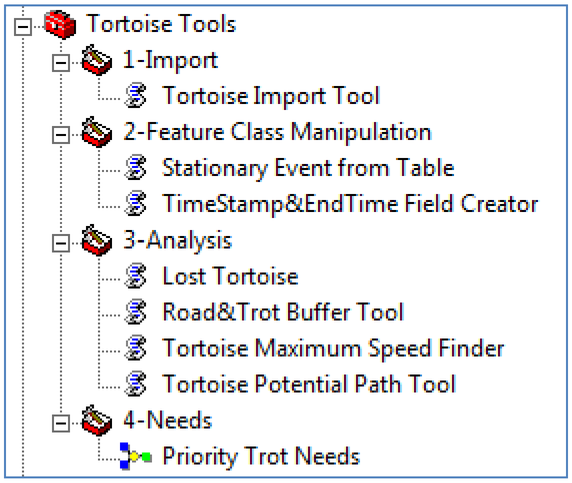

Tortoise Tools
1-Import
Tortoise Import Tool
2-Feature Class Manipulation
Stationary Event from Table
TimeStamp&EndTime Field Creator
3-Analysis
Lost Tortoise
Road&Trot Buffer Tool
Tortoise Maximum Speed Finder
Tortoise Potential Path Tool
4-Needs
Priority Trot Needs

Figure 5-1: Custom-Built ArcMap Tortoise Toolbox

To animate the tortoises' movements using their temporal attributes, the TimeStamp and EndTime attributes were needed. The *TimeStamp and EndTime Field Creator Tool* was built to automate the creation and population of these attributes. This tool was used in many of the use-cases for the project, as these attributes are the mechanism for time-based animation.

The *Road and Trot Buffer Tool* was created to calculate either one or two buffers from the roads feature class, and one buffer for the curbing. This tool helps in visualizing whether a tortoise's recorded positions or potential paths overlap the danger zones close to the roads.

For the weather data, the *Stationary Event from Table Tool* allows for animation of changes in recorded values of precipitation and temperatures.

The *Tortoise Import Tool* speeds up creation of multiple sub-feature classes for the project and tools, as well as for future data utilization. This tool automates a lengthy manual process and assists the users who do not have much experience with ArcGIS.

The *Tortoise Maximum Speed Finder Tool* creates a table in ArcMap that includes calculated maximum achieved speed for each tortoise for either the duration of the study or by year. The *Tortoise Potential Path Tool* and the *Lost Tortoise Tool* use this table.

The *Tortoise Potential Path Tool* uses the maximum speed table for calculating potential distances travelled by a tortoise between every two consecutive positions. The potential distance can then be used to create the time-attributed potential path ellipse that the tortoise might travel during the lapse between the two recorded times.

The *Lost Tortoise Tool* also uses the information from the maximum speed table to create a search polygon for a lost tortoise. After the last known position and datetime is entered, the tool calculates its potential areas travelled.

Besides the above custom tools, Export to KML 2.5.5 is used to export the feature classes and into KML format for time-based animation in Google Earth. This tool was created by Kevin Martin, and is available for free download from the ESRI developer's network (Martin, 2010). Both the feature classes and KMLs make up the data deliverables to the client. The KML files are viewable in AGX; however, the KML layer display attributes cannot be modified.

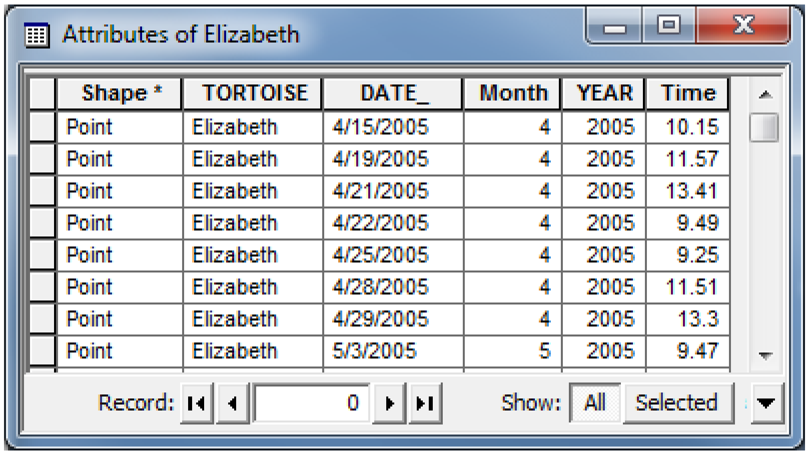
5.1 TimeStamp and EndTime Field Creator

Much of the provided data include date and time attributes; however, inconsistent formatting and the timestamp being broken into two separate attributes were problems. The other tools require consistency of a single format for timestamp for both start time and end time. Therefore, the *TimeStamp and EndTime Field Creator Tool* was created to standardize temporal attributes, which provide the backbone of the toolbox. Additionally, the tool provides methods to add an EndTime field and populate it based on the chosen method. The TimeStamp and EndTime fields created by the tool are the key elements for time-based animation in KML, as well as for the ArcMap animation in ArcMap 10.

5.1.1 Formatting Time

The original date field was in ESRI's date format and the time was in a separate field as a Double. Also, there were unnecessary fields called Month and Year. However, the timestamp needed to be a single field in a consistent usable format.

Originally, the created tool made the new attribute a date field. However, in Python, the date fields were passed in as MM/DD/YYYY format. Many of the tools that were created later need to perform many sorting routines by date. Sorting dates in that format becomes quite complicated as a normal sort in Python orders them by month, then by day, then by year. Since many of the tools would have to perform date sorting, this tool changes the format for the timestamp fields to be text in the ISO date-time format: YYYY-MM-DD HH:MM:SS. Sorting in Python becomes much more efficient with this format, as it orders the times correctly. The EndTime attribute applies the same format (Figure 5-2).

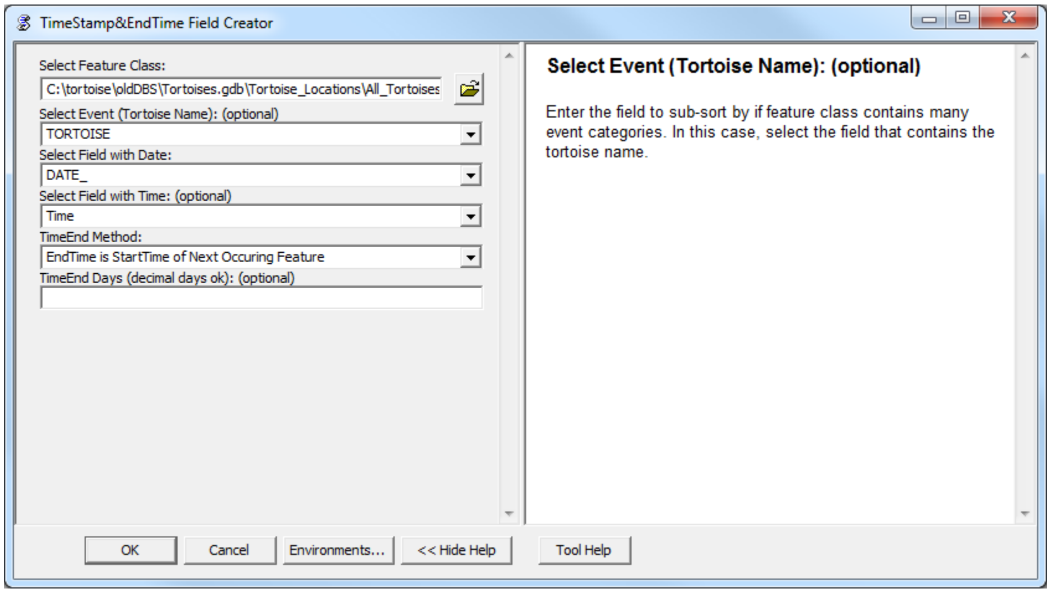

Attributes of Elizabeth

Shape *	TORTOISE	DATE_	Month	YEAR	Time
Point	Elizabeth	4/15/2005	4	2005	10.15
Point	Elizabeth	4/19/2005	4	2005	11.57
Point	Elizabeth	4/21/2005	4	2005	13.41
Point	Elizabeth	4/22/2005	4	2005	9.49
Point	Elizabeth	4/25/2005	4	2005	9.25
Point	Elizabeth	4/28/2005	4	2005	11.51
Point	Elizabeth	4/29/2005	4	2005	13.3
Point	Elizabeth	5/3/2005	5	2005	9.47

Figure 5-2: Original Tortoise Date and Time Fields

5.1.2 Populating TimeStamp (Start Time)

The TimeStamp field (start time) is created by the tool and populated by either just taking the date field or by taking the date and time fields, and converting to the ISO format as text (Figure 5-3).


TimeStamp&EndTime Field Creator

Select Feature Class:
C:\tortoise\oldDBS\Tortoises.gdb\Tortoise_Locations\All_Tortoises
Select Event (Tortoise Name): (optional)
TORTOISE
Select Field with Date:
DATE_
Select Field with Time: (optional)
Time
TimeEnd Method:
EndTime is StartTime of Next Occuring Feature
TimeEnd Days (decimal days ok): (optional)

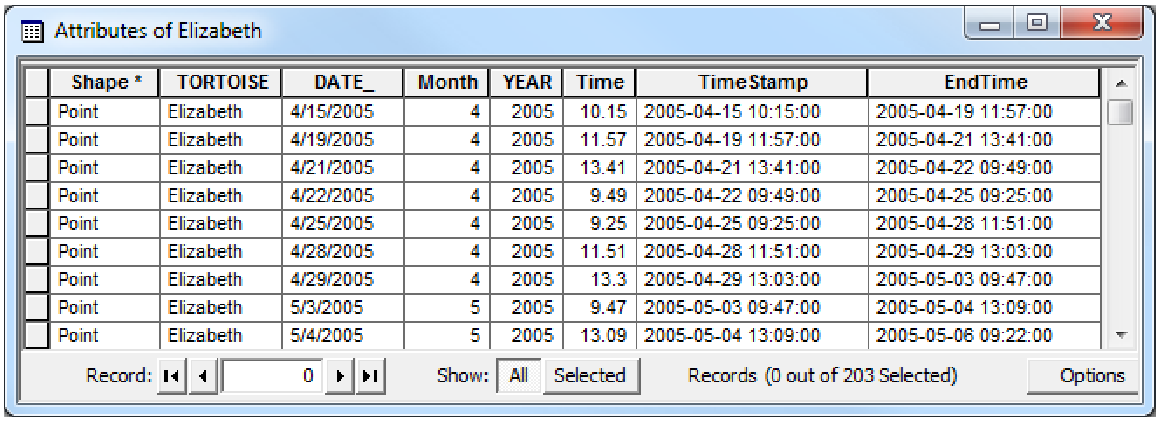
Select Event (Tortoise Name): (optional)

Enter the field to sub-sort by if feature class contains many event categories. In this case, select the field that contains the tortoise name.

Figure 5-3: TimeStamp and EndTime Field Creator Tool

5.1.3 Populating EndTime

The user has the option to add and populate an EndTime field with the tool. The EndTime field is in the same format as the TimeStamp field. EndTime can be calculated by one of two methods that is selected by the user. The method EndTime is StartTime of Next Occurring Feature creates a sorted Python list of the event dates, then populates EndTime by finding its matched date in the table and retrieving the next date from the list. Each tortoise is handled separately, based on the user-selected tortoise name. The method EndTime is Start of Next Day(s), allows the user to enter days or decimal days. The EndTime is calculated and populated by adding the entered days to the TimeStamp (Start Time) field (Figure 5-3). Adding days requires Python to convert the field string back into a date object. Once the tool has been run, the feature class contains the new populated fields (Figure 5-4).

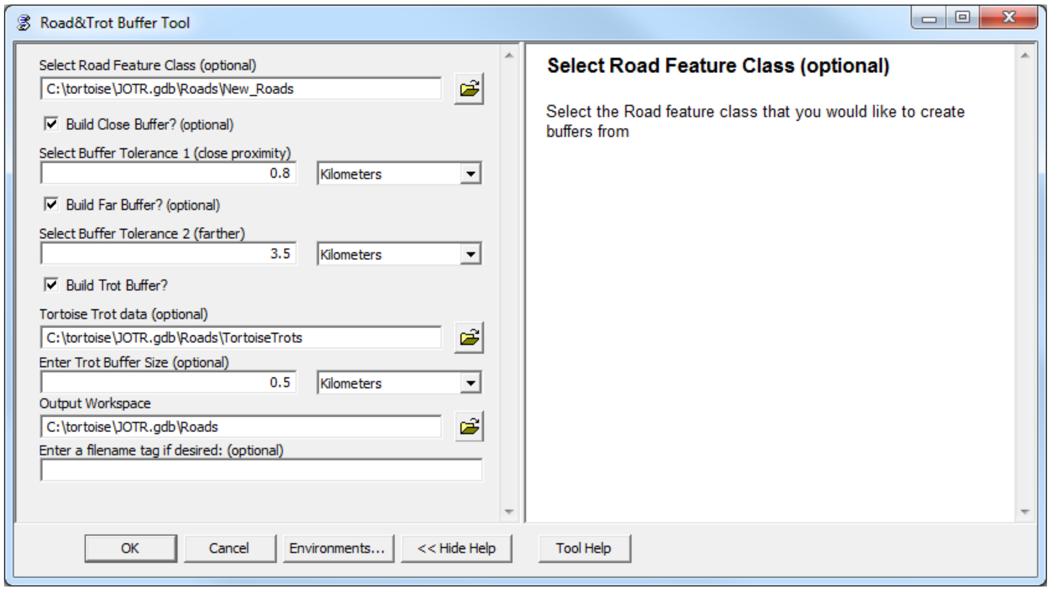

Attributes of Elizabeth

Shape *	TORTOISE	DATE_	Month	YEAR	Time	TimeStamp	EndTime
Point	Elizabeth	4/15/2005	4	2005	10.15	2005-04-15 10:15:00	2005-04-19 11:57:00
Point	Elizabeth	4/19/2005	4	2005	11.57	2005-04-19 11:57:00	2005-04-21 13:41:00
Point	Elizabeth	4/21/2005	4	2005	13.41	2005-04-21 13:41:00	2005-04-22 09:49:00
Point	Elizabeth	4/22/2005	4	2005	9.49	2005-04-22 09:49:00	2005-04-25 09:25:00
Point	Elizabeth	4/25/2005	4	2005	9.25	2005-04-25 09:25:00	2005-04-28 11:51:00
Point	Elizabeth	4/28/2005	4	2005	11.51	2005-04-28 11:51:00	2005-04-29 13:03:00
Point	Elizabeth	4/29/2005	4	2005	13.3	2005-04-29 13:03:00	2005-05-03 09:47:00
Point	Elizabeth	5/3/2005	5	2005	9.47	2005-05-03 09:47:00	2005-05-04 13:09:00
Point	Elizabeth	5/4/2005	5	2005	13.09	2005-05-04 13:09:00	2005-05-06 09:22:00

Figure 5-4: Feature Class with New Fields Added

5.2 Road and Trot Buffer Tool

The *Road and Trot Buffer Tool* builds buffers around the roads and curbing. The tool allows the user to enter a near road buffer distance, a far road buffer distance, and a single buffer distance for the curbs. Since the curbs with tortoise trots have an implementation year, the tool also dissolves the buffers based on the year. The *TimeStamp and EndTime Field Creator Tool* creates a TimeStamp field for the curbs with tortoise trots to support dynamic animation. The tool has default values of 0.8 kilometer populated for the near road buffer distance, and 2.5 kilometers for the far road buffer distance. The default road buffers are based on suggested values in the study by Boarman, Sazaki, and Jennings (1997). The road buffers created by these distances values indicate the area in which any tortoises found dead may have been hit by a car. Additionally, a buffer may be created for the tortoise trot curbing. This can be used for erasing sections of road that already have curbing, and utilized for the *Priority Trot Needs Tool* as described in Section 5.8. The user may change the default values to create other buffers. The user may also change the measurement units (Figure 5-5).


Road&Trot Buffer Tool

Select Road Feature Class (optional)
C:\tortoise\JOTR.gdb\Roads\New_Roads

☑ Build Close Buffer? (optional)

Select Buffer Tolerance 1 (close proximity)
0.8 Kilometers

☑ Build Far Buffer? (optional)

Select Buffer Tolerance 2 (farther)
3.5 Kilometers

☑ Build Trot Buffer?

Tortoise Trot data (optional)
C:\tortoise\JOTR.gdb\Roads\TortoiseTrots

Enter Trot Buffer Size (optional)
0.5 Kilometers

Output Workspace
C:\tortoise\JOTR.gdb\Roads

Enter a filename tag if desired: (optional)

Select Road Feature Class (optional)

Select the Road feature class that you would like to create buffers from

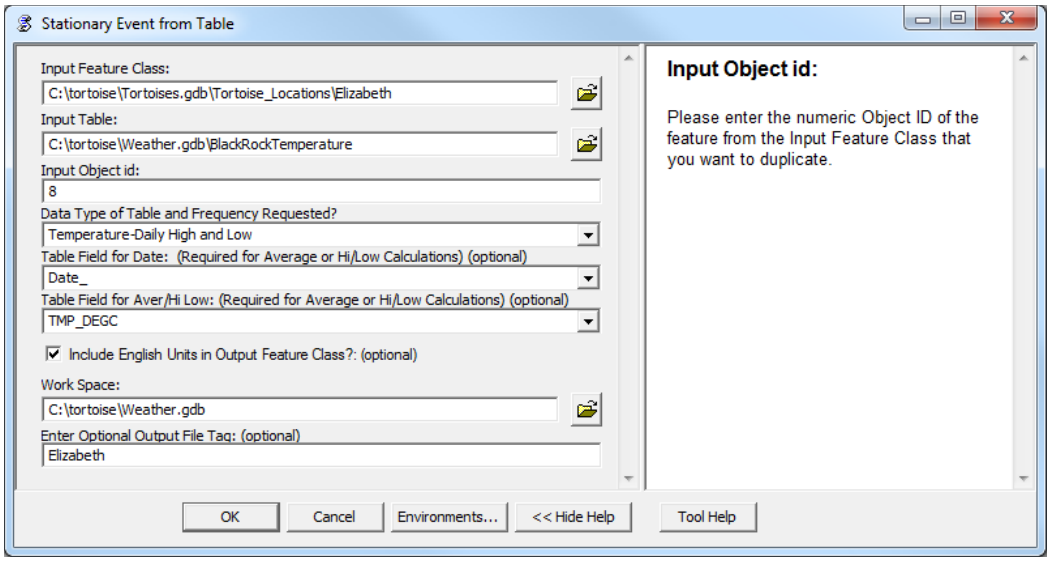
Figure 5-5: Road and Trot Buffer Tool

5.3 Stationary Event from Table Tool

The *Stationary Event from Table Tool* converts any non-geographic event table data into a geographic, dynamic feature class to prepare it for animated visualizations in Google Earth, AGX, and ArcGIS 10.

5.3.1 Event Table

The event table defined in this project is an ArcMap table that contains a TimeStamp and, optionally, an EndTime field such that any record in the table represents an event occurring at a particular time. The event tables for weather data were created by adding comma-delimited weather data to a text editor to remove header information and remove spaces from the field names, and then importing it into ArcMap. The temperature and precipitation tables each consist of hourly recordings from January of 2005 to January of 2009. The *TimeStamp and EndTime Field Creator Tool* was then used to create the proper fields (Figure 5-6).


Stationary Event from Table

Input Feature Class:
C:\tortoise\Tortoises.gdb\Tortoise_Locations\Elizabeth

Input Table:
C:\tortoise\Weather.gdb\BlackRockTemperature

Input Object id:
8

Data Type of Table and Frequency Requested?
Temperature-Daily High and Low

Table Field for Date: (Required for Average or Hi/Low Calculations) (optional)
Date_

Table Field for Aver/Hi Low: (Required for Average or Hi/Low Calculations) (optional)
TMP_DEGC

[✓] Include English Units in Output Feature Class?: (optional)

Work Space:
C:\tortoise\Weather.gdb

Enter Optional Output File Tag: (optional)
Elizabeth

Input Object id:

Please enter the numeric Object ID of the feature from the Input Feature Class that you want to duplicate.

Figure 5-6: Stationary Event from Table Tool

5.3.2 Geometry for the Stationary Event

The tool allows the user to select any single feature, of any feature type, from any georeferenced feature class. This feature's geometry will be used for all of the geometries for the stationary event. The feature class and Object ID of the feature must be specified in the tool. This single geometry will represent the event location, enabling stationary animation by symbol and label.

5.3.3 Options for Creating Event Tables

The tool has several methods to choose from in the parameter Data Type of Table and Frequency Requested, to create the stationary event.

- Temperature High and Low

  This option produces two feature classes: a feature class that has a single record for each day with the high temperature of that day, and a feature class that has a single record for each day with the low temperature of that day.

- Average Daily Temperature

  This option produces a feature class that has a single record for each day with the average temperature of that day.

- Hourly Temperature

  If the data from the table are stored hourly, this option merely creates a feature for each row of the table. The number of objects in the output feature class will equal the number of records in the table. This process was not used for the data in the project due to the large number of records for the project.

- Daily Precipitation

  This option produces a feature class that has a single record for each day with the total precipitation of that day. This is calculated by adding up the hourly totals for the day.

- Other Method

  This option can be used when the user would like to tie the records from other types of time-attributed tabular data to a stationary geometry. It creates a feature for each row of the table. The number of objects in the output feature class will equal the number of records in the table. This process was not used for the data in the project.

5.3.4 Metric/English Units

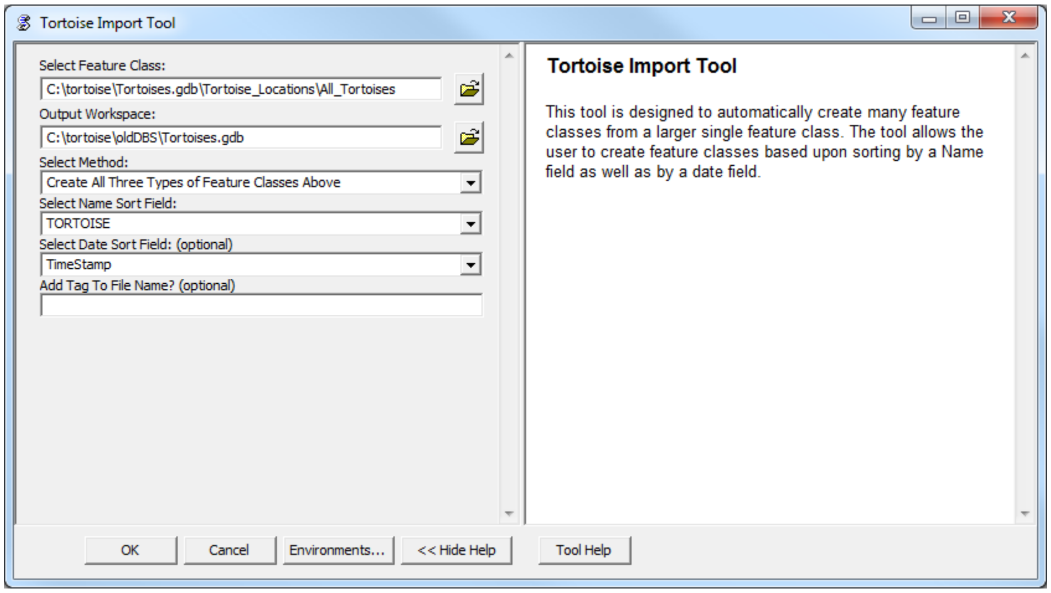
The Include English Units in Output Feature Class option in the tool creates a field that contains the English units that result from converting the metric units. This is used for both temperature and precipitation.

5.4 Tortoise Import Tool

The *Tortoise Import Tool* allows the user to create multiple sub-feature classes from a single feature class, based on tortoise name, year, or tortoise name and year. This tool replaces the client's manual process used to create 68 tortoise feature classes within seconds. This will also be very useful for the processing the updated tortoise data in the future. Additionally, it can be used to extract sub-datasets from any data based on an attribute field or time.

5.4.1 Options for Importing Data

The tool contains several methods that the user can choose from depending on their data need (Figure 5-7).



Select Feature Class:
C:\tortoise\Tortoises.gdb\Tortoise_Locations\All_Tortoises
Output Workspace:
C:\tortoise\oldDBS\Tortoises.gdb
Select Method:
Create All Three Types of Feature Classes Above
Select Name Sort Field:
TORTOISE
Select Date Sort Field: (optional)
TimeStamp
Add Tag To File Name? (optional)

Tortoise Import Tool

This tool is designed to automatically create many feature classes from a larger single feature class. The tool allows the user to create feature classes based upon sorting by a Name field as well as by a date field.

Figure 5-7: Tortoise Import Tool

- Create Feature Classes by Tortoise

  This queries the master database by tortoise and produces a feature class for each tortoise, which results in 18 new feature classes.

- Create Feature Classes by Tortoise and Year

  This queries the master database by tortoise and year and produces a feature class for each tortoise for each year. There are 46 feature classes in the output.

- Create All Tortoises by Year

  This queries the master database by year and produces a feature class for each year that contains all of the tortoises. There are four feature classes in the output.

- Create All Three Types of Feature Classes

  This option applies all of the previous methods to produce all of the different sub-feature class types. This was the method used for the project to create the 68 delivered feature classes for the client.

5.4.2 Naming Feature Classes

When multiple sub-feature classes are generated, it is important to define a naming schema to differentiate them. The tortoise name, all_tortoises, or name and year together are tagged to the name of the output files where appropriate. Additionally, there is an optional parameter to add an extra preceding tag to the name to further identify the output.

5.5 Tortoise Maximum Speed Finder

The *Tortoise Maximum Speed Finder Tool* calculates the maximum sustained speed achieved by each tortoise. Not only is the table useful for the study of the tortoise speed, but it supports the *Tortoise Potential Path Tool* and the *Lost Tortoise Tool* (Sections 5.6 and 5.7). The *TimeStamp and EndTime Field Creator Tool* must be run prior to using this tool.

5.5.1 Methods to Calculate the Maximum Speed

The user can choose from two methods.

- Tortoise by Name

  This calculates the maximum speed achieved by each particular tortoise over the period covered in the feature class.

- Create All Three Types of Feature Classes

  This calculates the maximum speed achieved by each particular tortoise for each year.

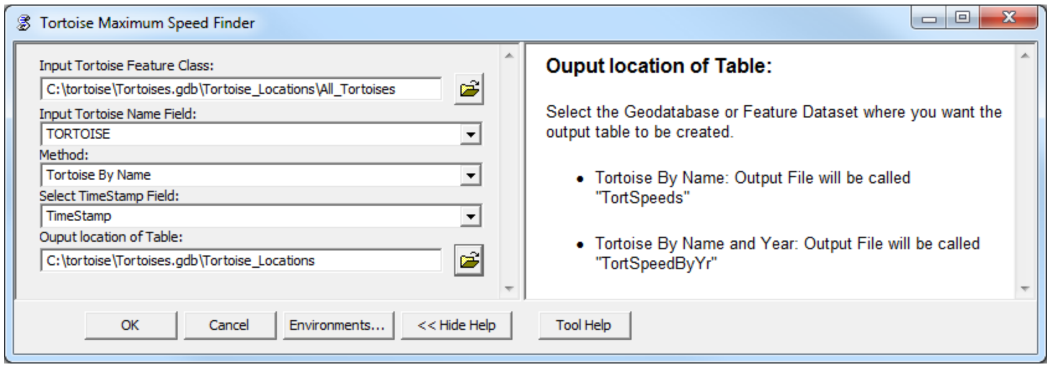
Speed is determined by dividing the distance travelled between any two consecutive positions by the time elapsed between the same two positions. The resulting speed is in meters per hour.

In this project, the Euclidean distance formula was used to calculate the distance between any two points with X and Y coordinates. The coordinates were defined in the Universal Transverse Mercator (UTM) projection, as it is the requested projection for NPS data. UTM has the advantage of being measured in linear meters, rather than angular degrees. This makes it more reliable than geographic coordinate systems and ideal for calculating distances (Cole, 1977). The coordinates are measured in meters North and East from a local origin.

The time is measured by the difference between the TimeStamps of the two positions. When subtracting two dates in Python, the result is in decimal hours, so the equation provides the result in the needed units.

5.5.2 Finding Maximum Tortoise Speed with Python

The Python script first creates a list of the possible tortoises, and optionally a list of the years from the tortoise feature class. The script iterates through each tortoise and creates a date-ordered list of position dates of the particular tortoise. The script then iterates through the rows in date order and calculates the tortoise speed between consecutive positions. If the time difference is less than 18 hours, the speed is not calculated. Tortoise positions are acquired once a day, unless a ranger has moved the tortoise. In that case, unrealistic speeds are calculated. The 18-hour minimum time difference prevents those speeds from being used. If the new tortoise speed is greater than the maximum tortoise speed (initially set to zero), the tortoise speed becomes the new maximum tortoise speed. Once the iteration is finished for the tortoise, or the tortoise and year, the script adds a new record to the output table indicating the tortoise, year, maximum speed, start time, end time, distance travelled, and time elapsed. This script performs the same process by iterating through each tortoise (Figure 5-8).


Tortoise Maximum Speed Finder

Input Tortoise Feature Class:
C:\tortoise\Tortoises.gdb\Tortoise_Locations\All_Tortoises
Input Tortoise Name Field:
TORTOISE
Method:
Tortoise By Name
Select TimeStamp Field:
TimeStamp
Ouput location of Table:
C:\tortoise\Tortoises.gdb\Tortoise_Locations

Ouput location of Table:

Select the Geodatabase or Feature Dataset where you want the output table to be created.

- Tortoise By Name: Output File will be called "TortSpeeds"
- Tortoise By Name and Year: Output File will be called "TortSpeedByYr"

Figure 5-8: Tortoise Maximum Speed Finder

5.6 Tortoise Potential Path Tool

The *Tortoise Potential Path Tool* uses the calculated tortoise's maximum speed to find the potential area that the tortoise could travel between any two consecutive recorded positions. The potential area is defined by an ellipse with the two recorded locations as the focal points (Figure 5-8). The tool populates a row in an ellipse table with all of the parameters needed, as well as TimeStamp and EndTime. Once all ellipse data are loaded into the table, a new feature class is created that includes all of the ellipses. Additionally, the tool has an option to create and populate a symbol angle field in the original tortoise feature class.

5.6.1 Building the Ellipse

An ellipse is defined by multiple parameters, including foci, semi-major axis, semi-minor axis, center, and the rotation angle. In Figure 5.9, F1 and F2 are the two focal points determined by the two consecutive positions of a tortoise.

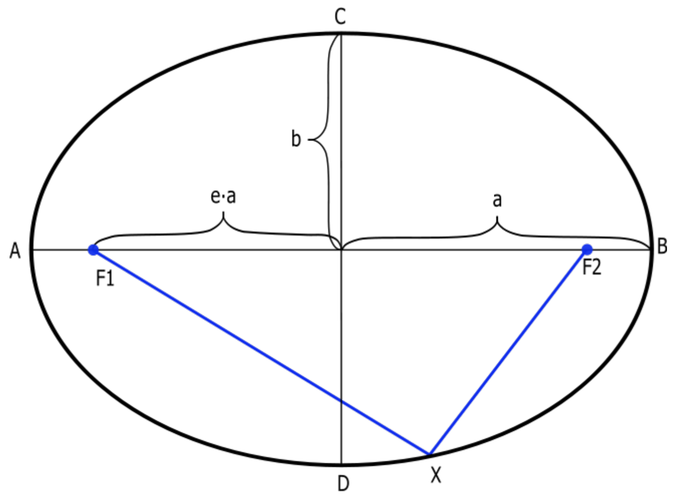


Figure 5-9: Components of an Ellipse.

The blue line represents the maximum potential distance that the tortoise can travel within the time lapse between the first recorded position (F1) and the second recorded position (F2). This distance equals the semi-major axis (labeled as a in Figure 5-9) multiplied by two. The maximum potential distance is calculated in the script by finding the time difference in hours between the two foci positions and multiplying it by the tortoise's maximum speed. It is worth noting that the time difference in Python is measured in seconds. The formula is listed below:

$$Total\ Hour\ Difference = (date\ difference * 24) + \left(time\ difference\ in\ seconds * \frac{1.0}{3600}\right)$$

In the above equation, the seconds are multiplied by 1.0, which is necessary in Python to make it a Double type. Once the maximum potential distance is obtained, the length of the semi-major axis can be calculated by dividing it in two.

By the nature of an ellipse, the sum of the distances from any point on the ellipse to those two foci is constant. Therefore, the length of blue and red lines in Figure 5-10 are the same. According to the Pythagorean Theorem, the semi-minor axis (denoted by *b*) squared plus one-half of the distance between the two foci (denoted by *ea*) squared equals to one-half the maximum potential distance squared. The semi-minor axis is then solved. See equation below:

$$\left(\frac{Potential\ Distance}{2}\right)^2 = b^2 + ea^2$$

The value for *ea* can be found because *ea* is exactly one-half of the actual distance calculated between the two consecutive positions designated by F1 and F2 in Figure 5-10. The actual distance is calculated by using the Euclidean distance formula and the UTM coordinates.

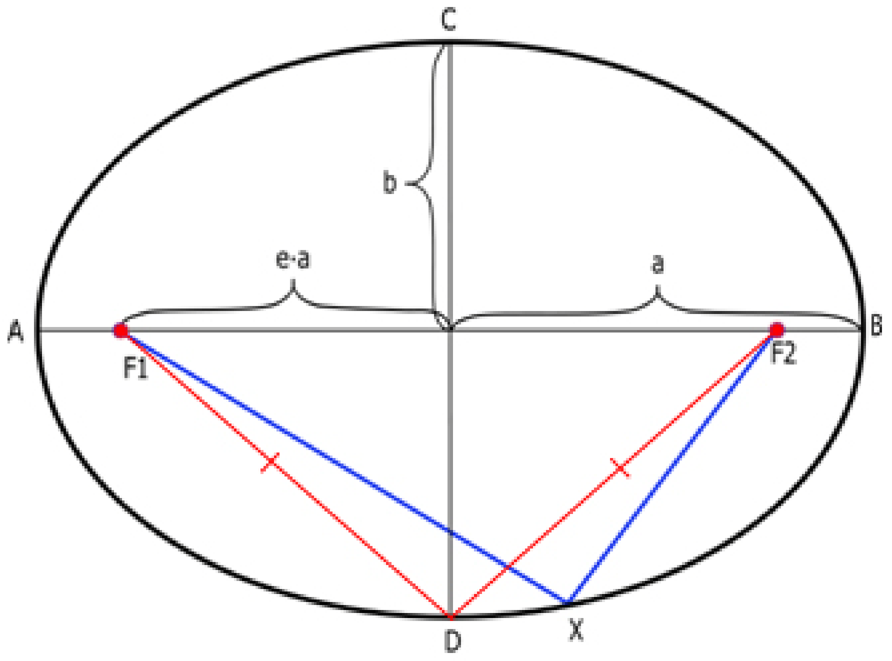


Figure 5-10: Ellipse with Potential Distance in Red and Blue

In some cases, when the user enters the tortoise maximum speed, the maximum potential distance may be less than the actual travel distance by the tortoise. Put differently, the blue line in Figure 5-10 after calculation may be shorter than the distance between the two foci. In those cases, the ellipse would be a collapsed ellipse. A calculation was added to the code that checked to see if this is the case. If so, the minor axis will be set to zero. Because the collapsed ellipses are not easy to see, the value for this case will be changed to 1.0. This makes the ellipses distinguishable and nearly resembling a line.

The center point of the ellipse is also a necessary parameter. The X-coordinate of the center is found by the sum of the X-coordinates of two foci and then divided by 2. The Y-coordinates are used in the same manor to find the Y-coordinate of the center. For the ellipse builder, integer values are required for the coordinates.

Lastly, the rotation angle is calculated in two steps. First, the Python formula for arctangent is used to calculate angle θ in radians (Figure 5-11). This angle is measured from the positive y-axis to the Major Axis.

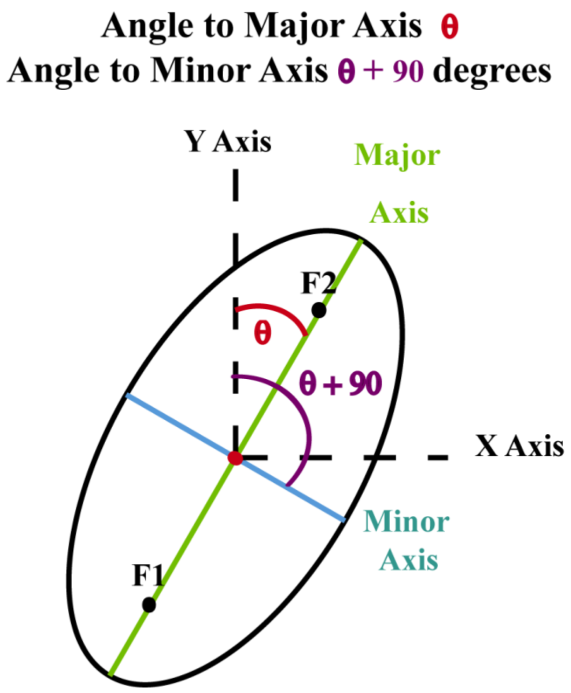


Figure 5-11: Angles from Y-Axis to Ellipse Major and Minor Axis

To avoid a divide by zero circumstance, a result of zero for Y_{F2}-Y_{F1} needs to be considered, as shown in the formula below:

$$\theta = \arctan\left(\frac{X_{F2} - X_{F1}}{Y_{F2} - Y_{F1}}\right),$$

$$where\ if\ Y_{F2} - Y_{F1} = 0\ and\ X_{F2} - X_{F1} \geq 0,\ \ \theta = 0\ and$$

$$if\ Y_{F2} - Y_{F1} = 0\ and\ X_{F2} - X_{F1} < 0,\ \ \theta = \pi$$

Once this angle is calculated, it needs to be converted to degrees. Normally this would be simple; however, the actual direction of the major axis is needed. This value will be used in another function of the tool that populates the symbol angle for the tortoise. For this reason, four different radian to degree conditions will have to be checked for every two consecutive pairs of coordinates, depending on whether X_{F2}-X_{F1} is positive or negative and whether Y_{F2}-Y_{F1} is positive or negative. The *Tortoise Potential Path Tool* applies the TableToEllipse command, which is part of the Military Analyst Extension. This extension is free to download from ESRI. This tool, however, measures its angles from the positive y-axis to the semi-minor axis; therefore, once the angle in radians is converted to degrees, 90 degrees will be added to the result to get the proper rotation angle. The following formulas convert to degrees, accounting for the true direction of the major axis, as well as adding 90 degrees to get the bearing of the minor axis needed for the TableToEllipse command:

$$If\ X_{F2} - X_{F1} \geq 0\ and\ Y_{F2} - Y_{F1} > 0, then\ rotation\ angle = \theta * \frac{180}{\pi} + 90$$

$$If\ X_{F2} - X_{F1} \geq 0\ and\ Y_{F2} - Y_{F1} < 0, then\ rotation\ angle = \theta * \frac{180}{\pi} + 270$$

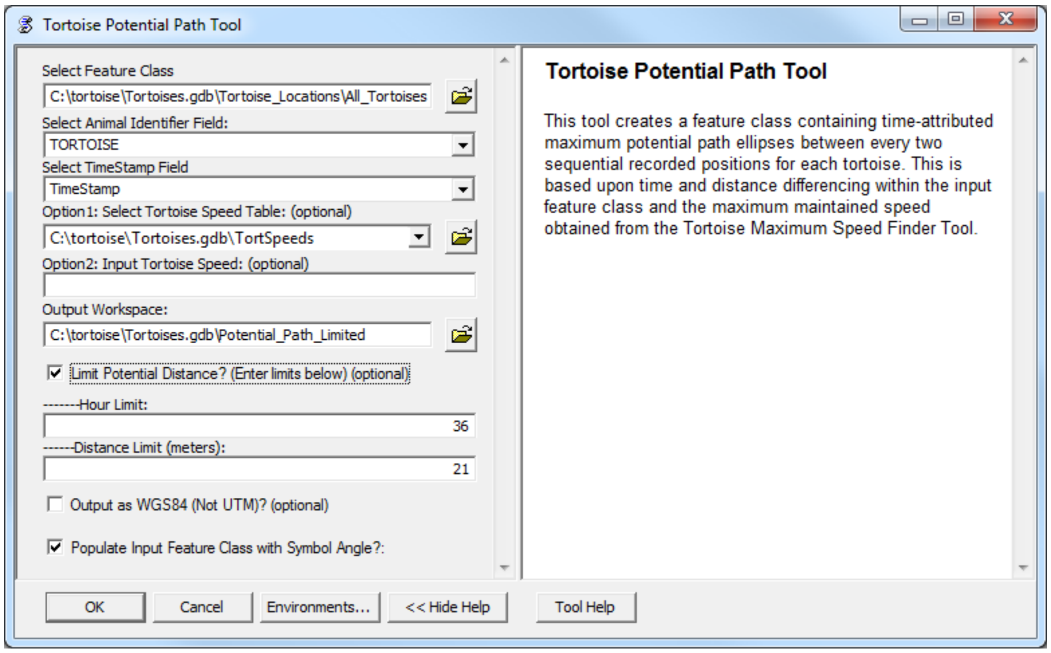
$$If\ X_{F2} - X_{F1} < 0\ and\ Y_{F2} - Y_{F1} < 0, then\ rotation\ angle = \theta * \frac{180}{\pi} + 270$$

$$If\ X_{F2} - X_{F1} < 0\ and\ Y_{F2} - Y_{F1} > 0, then\ rotation\ angle = \theta * \frac{180}{\pi} + 90$$

When the tortoise SymAngle field is populated further in the tool, 90 degrees is subtracted from the rotation angle to get the proper direction for the tortoise.

5.6.2 Tool Options

When using the tool, the user can select different options for two of the parameters (Figure 5-12).


Tortoise Potential Path Tool

Select Feature Class: C:\tortoise\Tortoises.gdb\Tortoise_Locations\All_Tortoises

Select Animal Identifier Field: TORTOISE

Select TimeStamp Field: TimeStamp

Option1: Select Tortoise Speed Table: (optional) C:\tortoise\Tortoises.gdb\TortSpeeds

Option2: Input Tortoise Speed: (optional)

Output Workspace: C:\tortoise\Tortoises.gdb\Potential_Path_Limited

☑ Limit Potential Distance? (Enter limits below) (optional)

-------Hour Limit: 36

------Distance Limit (meters): 21

☐ Output as WGS84 (Not UTM)? (optional)

☑ Populate Input Feature Class with Symbol Angle?:

Tortoise Potential Path Tool

This tool creates a feature class containing time-attributed maximum potential path ellipses between every two sequential recorded positions for each tortoise. This is based upon time and distance differencing within the input feature class and the maximum maintained speed obtained from the Tortoise Maximum Speed Finder Tool.

Figure 5-12: Tortoise Potential Path Tool

5.6.2.1 Tortoise Speed Options

The user can either reference the Tortoise Maximum Speed Table or choose to enter a specific speed in meters/hour for the creation of the ellipses. It is important to note that small user-entered speeds result in ellipses that do not contain the two positions when the potential distance is less than the actual distance travelled.

5.6.2.2 Limiting Potential Distance Options

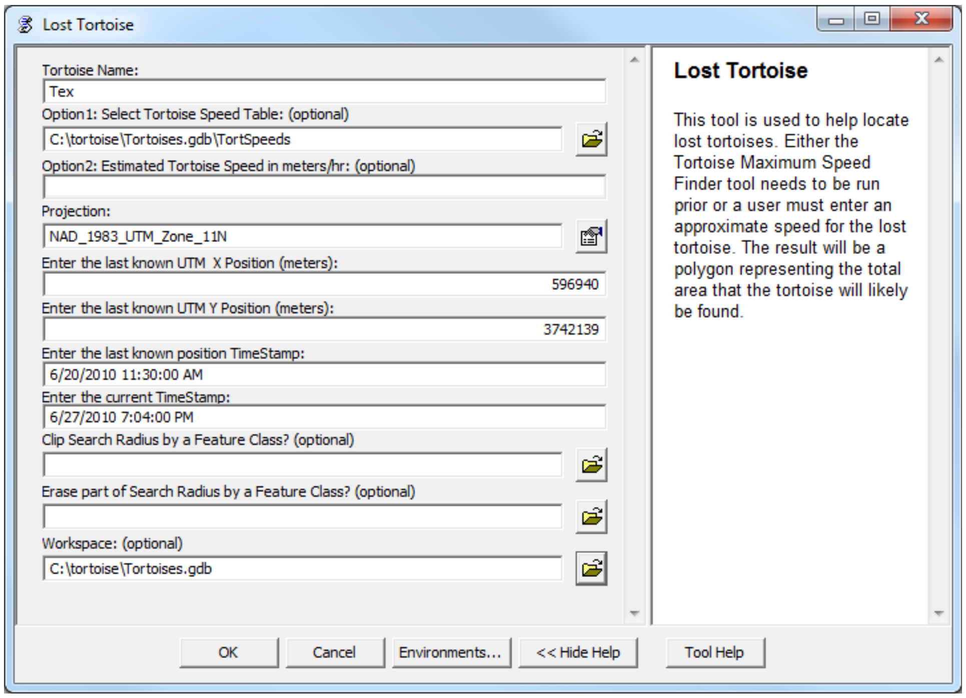
Normally the maximum speed for the tortoise is multiplied by the time difference to obtain the potential distance used to find the ellipse parameters. However, this can result in large potential path ellipses if there is a large time gap between consecutive positions. An option exists in the tool to override this maximum potential distance.

When the box is checked to limit the maximum potential distance, the values for hour limit and distance limit will be used. The hour limit represents the maximum time gap between positions for calculating potential distance. The distance limit is the minimum distance difference that should be considered.

If the time gap is larger than the hour limit value and the distance between positions is less than the distance limit, the maximum tortoise speed will be multiplied by 24 hours. This limits excessively large ellipses. The theory behind this method reflects that if a tortoise has not moved far and considerable time has passed, then it is likely that it did not take a long trip and return. The output of the tool yields an initial ellipse table, as well as the ellipse feature class.

5.7 Lost Tortoise Tool

The *Lost Tortoise Tool* uses the *Tortoise Maximum Speed Finder Tool* to help to locate tortoises that cannot be located (Figure 5-13).



Tortoise Name: Tex
Option1: Select Tortoise Speed Table: (optional) C:\tortoise\Tortoises.gdb\TortSpeeds
Option2: Estimated Tortoise Speed in meters/hr: (optional)
Projection: NAD_1983_UTM_Zone_11N
Enter the last known UTM X Position (meters): 596940
Enter the last known UTM Y Position (meters): 3742139
Enter the last known position TimeStamp: 6/20/2010 11:30:00 AM
Enter the current TimeStamp: 6/27/2010 7:04:00 PM
Clip Search Radius by a Feature Class? (optional)
Erase part of Search Radius by a Feature Class? (optional)
Workspace: (optional) C:\tortoise\Tortoises.gdb

Lost Tortoise

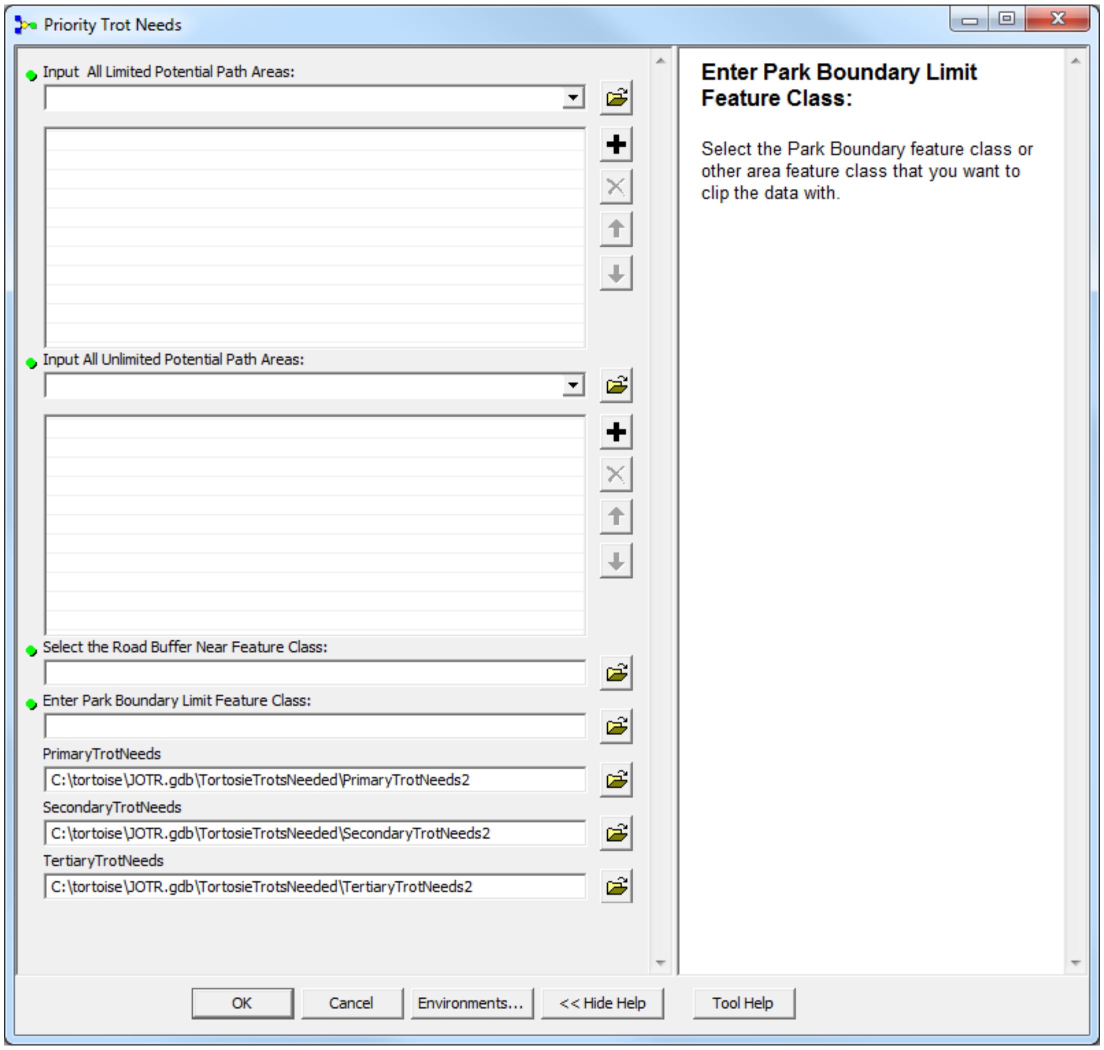
This tool is used to help locate lost tortoises. Either the Tortoise Maximum Speed Finder tool needs to be run prior or a user must enter an approximate speed for the lost tortoise. The result will be a polygon representing the total area that the tortoise will likely be found.

Figure 5-13: Lost Tortoise Tool

The tool uses the maximum achieved speed for the tortoise and the elapsed time to obtain a maximum potential distance travelled. Using this maximum potential distance as the input buffer distance, a buffer can be created at the last recorded position of the lost tortoise to represent the potential area where could have travelled. The projection parameter is required by the geoprocessing command used to create the new feature class. Optionally, feature classes can be selected to both clip and erase portions of the search buffer. This could be used when there are known barriers that inhibit tortoise movement.

5.8 Priority Trot Needs Tool

The *Priority Trot Needs Tool* was the only tool developed in ModelBuilder. The tool uses core ArcMap tools and data produced by the *Tortoise Potential Path Tool* and the *Road and Trot Buffer Tool*. By selecting all the unlimited and limited potential path feature classes, the near road buffer feature class, and a clip feature class, the tool processes the data to produce three feature classes depicting three levels of priority for road lengths needing curbing with tortoise trots (Figure 5-14).


Priority Trot Needs

Input All Limited Potential Path Areas:

Input All Unlimited Potential Path Areas:

Select the Road Buffer Near Feature Class:

Enter Park Boundary Limit Feature Class:

PrimaryTrotNeeds
C:\tortoise\JOTR.gdb\TortosieTrotsNeeded\PrimaryTrotNeeds2

SecondaryTrotNeeds
C:\tortoise\JOTR.gdb\TortosieTrotsNeeded\SecondaryTrotNeeds2

TertiaryTrotNeeds
C:\tortoise\JOTR.gdb\TortosieTrotsNeeded\TertiaryTrotNeeds2

Enter Park Boundary Limit Feature Class:

Select the Park Boundary feature class or other area feature class that you want to clip the data with.

Figure 5-14: *Priority Trot Needs Tool*

The tool can best be described using Boolean logic.

Primary Trot Needs =

(Limited Potential Paths AND Near Road Buffers) AND NOT

Existing Curb Buffers

Secondary Trot Needs =

(Unlimited Potential Paths AND Near Road Buffers) AND NOT

(Primary Trot Needs OR Existing Curb Buffers)

Tertiary Trot Needs =

Near Road Buffers AND NOT

(Primary Trot Needs OR Secondary Trot Needs OR Existing Curb Buffers)

The tool uses the core tools for this logic. The intersect tool is used for the AND operator, and the erase tool and the clip tool are used for the NOT operator. The OR operator is achieved by performing the appropriate operator again.

Initially, all of the limited potential path polygons are dissolved by tortoise name into a new feature class. This new feature class is then dissolved entirely. This two-step dissolve considerably speeds up the dissolving needed as fewer dissolves are needed. The same process is performed on the unlimited potential path polygons.

The unlimited and limited potential path feature classes are then clipped by a selected feature class. In this project, the JOTR park boundary was used for clipping. The two new feature classes are then clipped by the Road_Buffer_Near feature class. The clipped limited potential path feature class is output as the PrimaryTrotNeeds feature class. This output is additionally used to erase portions of the unlimited potential path feature class to output the SecondaryTrotNeeds feature class. The PrimaryTrotNeeds and SecondaryTrotNeeds are then erased from the Road_Buffer_Near feature class to output the TertiaryTrotNeeds feature class (Figure 5-15).

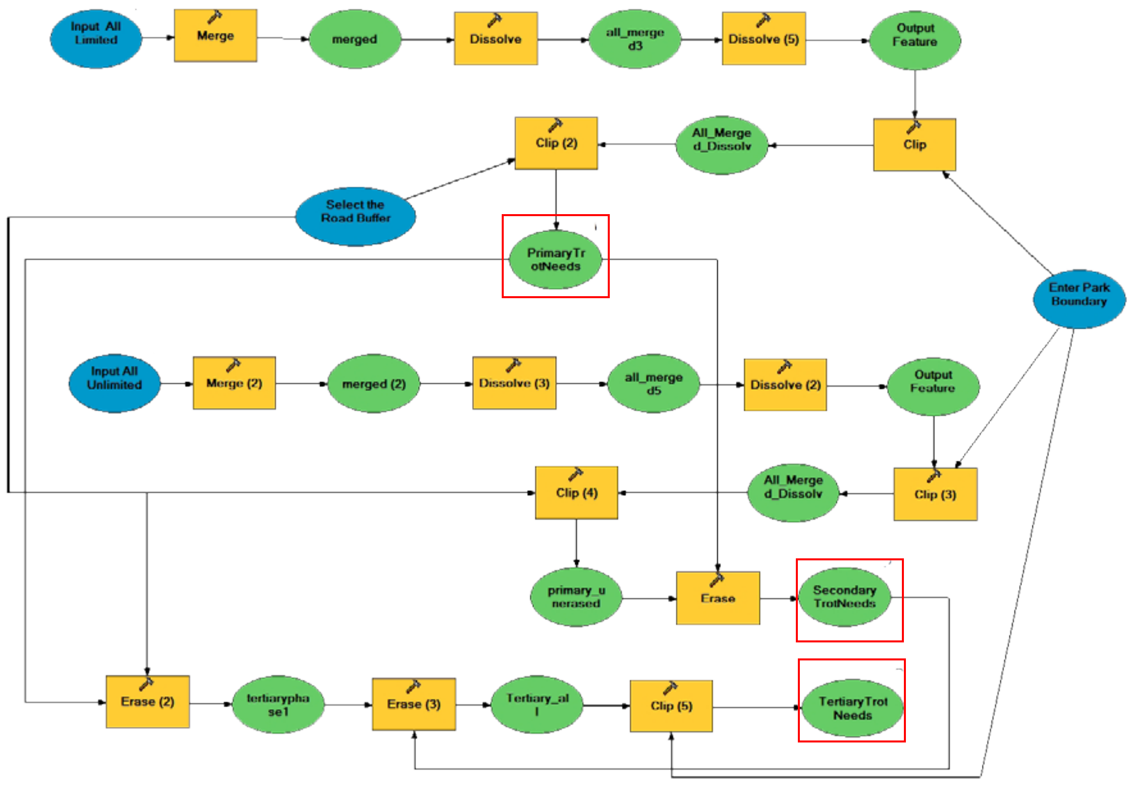


Figure 5-15: Priority Trot Needs Tool Outputs in ModelBuilder

5.9 Export to KML

Before the output files can be dynamically visualized in Google Earth or ArcGIS Explorer (AGX), they need to be exported to KML format. This was achieved by using the Export to KML 2.5.5 tool downloaded from the ESRI Developers' Network. The tool supports both the TimeStamp and EndTime fields created by the custom-built tools. Collaboration with the developer, Kevin Martin, included platform testing and fine-tuning the tool to produce dynamic labeling. The capability of this tool exceeded the baseline ArcMap Layer to KML tool, which currently does not have the capability to export TimeStamp and EndTime.

To visualize KML files, both Google Earth and AGX allow for multiple KML time animations. In AGX, all of the KMLs must be wrapped into a single KMZ to allow simultaneous animations. However, AGX does not allow for representation editing of the KMLs. Such things as symbology and elevation value or method cannot be manipulated on the KML layers. Since most of the KMLs needed their representation changed for the deliverables, AGX was not used as the primary display software.

Additionally, the feature classes created in this project can be animated within ArcGIS 10. ArcGIS 10 includes the capability to turn on animations for any feature class using the TimeStamp and EndTime attributes. This new capability adds even more value to the custom-built tools.

5.10 Summary

This chapter outlined the various tools built and used for the project. Custom-built tools were used for importing, manipulating, and analyzing the data. The user-built tool, Export to KML, was used to create KML files from the data created from the tools to create time-based animations in Google Earth and AGX. ArcMap 10 promises to exploit the capability of the created tools even further.

Chapter 6 – **Results and Analysis**

The objectives of the project were accomplished through the seven custom tools built with Python in ArcGIS, a ModelBuilder tool, and the existing Export to KML 2.5.5 tool. The results can be animated and visualized in Google Earth.

To depict danger areas related to the roads in JOTR, the *Road and Trot Buffer Tool* can generate static buffers around the roads and dynamic buffers around the curbing with tortoise trots. The precipitation and temperature data are used in the *Stationary Event from Table Tool* to create dynamic stationary weather events. The dynamic tortoise movement data are produced by the *TimeStamp and EndTime Field Creator Tool*. The *Tortoise Maximum Speed Finder Tool* analyzes the tortoise movement data to yield a Tortoise Speed Table that contains the maximum achieved speed for each tortoise. The output from this tool is used by the *Tortoise Potential Path Tool* to generate the tortoise dynamic potential path ellipses between every two consecutive recorded positions of the tortoises. The *Lost Tortoise Tool*, along with the Tortoise Speed Table, produces a buffer that depicts the search extent for locating the tortoise. All of these tools require data that has been processed by the *TimeStamp and EndTime Field Creator Tool*. This tool also processed the client-provided 95PVC contours, MCPs, and Kernel Density polygons to visualize them dynamically.

All of the output feature classes from these tools are exported to dynamic KML format using the *Export to KML Tool*. For the project, the KML files were packaged together for the client, which enables selective animation of multiple feature types in Google Earth. However, these files can also be viewed in ArcGIS Explorer (AGX) with limitations in customized viewing. Additionally, with the newly released ArcGIS 10, the client will be able to animate the various feature classes within ArcMap. This chapter discusses the installed components, different scenarios in which the tools can be used for analysis, and issues that arose in the project.

6.1 Installation of Components

6.1.1 Custom-Built Tools

The seven custom-built tools were created in ArcMap using Python and ArcMap's geoprocessing engine. The other tool was built using ModelBuilder. The eight tools are contained in an ArcMap toolbox called Tortoise Tools. The tools were designed in ArcGIS 9.3.1 with ArcInfo License.

Within an ArcMap session, open the ArcToolbox Window. Within the white space of the window, a right mouse-button click will open a menu (Figure 6-1) in which Add Toolbox must be selected. When the browse window opens, navigate to the folder with the Tortoise Tools toolbox and highlight it, then click open. The toolbox is now loaded for only the current session.

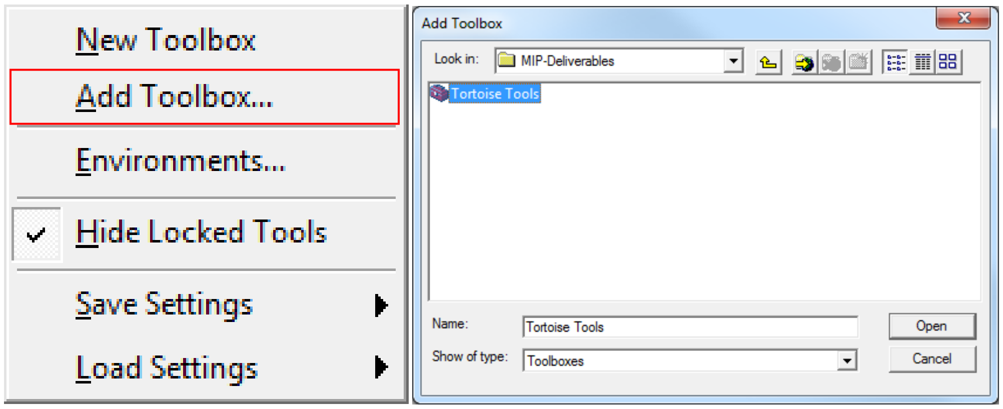


Figure 6-1: Loading the Tortoise Tools, Step 1

After the previous step has been completed, the user may desire the tools to be available for all sessions. Within the white space of the window, a right mouse-button click will open a menu (Figure 6-2) in which Save Settings must be selected, followed by To Default. Now the toolbox will be available for all sessions. It is noted that each user must perform this, as the ArcMap Toolboxes are stored in the personal profile.

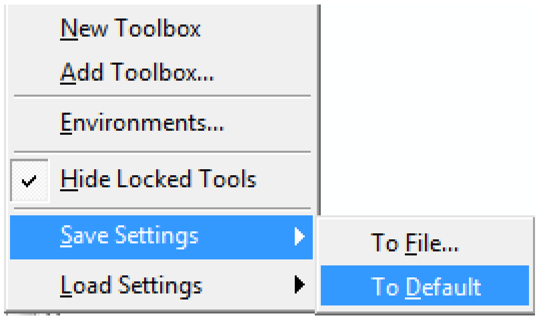


Figure 6-2: Saving Toolbox for All Sessions

6.1.2 Military Analyst Extension

Within ArcMap 9.3.1, the Military Analyst Extension must be installed in order for the *Tortoise Potential Path Tool* to work. This extension contains the TableToEllipse geoprocessing function required to build ellipses. The Python script calls this function. ArcGIS 10 includes a core TableToEllipse Python function and does not require Military Analyst. The function in ArcGIS 10 uses a different set of parameters and syntax than Military Analyst extension for 9.3. The *Tortoise Potential Path Tool* accounts for this by including a check for version number. If ArcGIS 10 is used, the tool will use the newer function.

6.1.3 Export to KML 2.5.5

The Export to KML 2.5.5 tool must be installed within ArcMap for exporting the dynamic feature classes into KML.

The tool can be downloaded from the ArcScripts website: http://arcscripts.esri.com. The installation of the tool varies depending on the version of Windows and an administrator must install it. The instructions and requirements are included in the download.

6.1.4 Google Earth

Google Earth is the primary means for viewing the dynamic layers of this project. Google Earth is free to download from the Google website: http://earth.google.com.

The Google Earth website contains the instructions necessary to install the application. The provided KML files can be opened by either dragging the file to Google Earth, or by selecting File from the main menu and then clicking Open. A listing of the delivered KML layers is provided in Appendix C.

6.1.5 ArcGIS Explorer

ArcGIS Explorer (AGX) is a powerful tool very similar to Google Earth; each application has its unique benefits and capabilities. If AGX had the capability to change symbol representations of KML, like Google Earth, it would be another primary means for animating the results of this project.

AGX is free to download from the ESRI website and simple to install using the provided instructions. This section does not cover the specifics on opening the KML files in AGX, as the next version may require a new method.

6.2 Use-Cases and Results

The developed tools can be used in various combinations to perform analysis of tortoise movement and their environment under different scenarios. This section addresses eight use-cases for the tools and demonstrates the results of each use-case.

The tools were built to allow flexibility for the user, as well as to be compatible with each other. The feature classes created are based upon user-entered parameters that affect the results. Parameters for this project and the provided data are based on a combination of previous studies, JOTR criteria, and by achieved results.

Since Google Earth can be used for all use-cases, a brief introduction to using the Export to KML 2.5.5 tool and a listing of the delivered KML layers is provided in Appendix C.

6.2.1 What are the danger zones near roads?

The roads within JOTR are mostly unchanging; however, the curbing with tortoise trots were added to portions of the roads where tortoises have been observed in the relative vicinity. The curbing has been added incrementally through the past ten years and is intended to prevent people from driving off-road, yet allows the tortoises to egress from the road. JOTR can examine the roads and curbs with trots, in relation to the tortoises by using the *Road and Trot Buffer Tool* to create a close road buffer, a far road buffer, and a curb with trot buffer. The curb with trot buffers are dissolved, or merged, by their year of implementation so they will only show up in a dynamic display once the year of their implementation is reached. Additionally, the analyst may want to create a buffer around the solid curbs using the tool.

6.2.1.1 Process

The workflow of this scenario is shown in Figure 6-3, in which blue represents feature classes, red represents tools, green represents KML files, and yellow represents animations. This color scheme will be used for the workflows in this chapter.

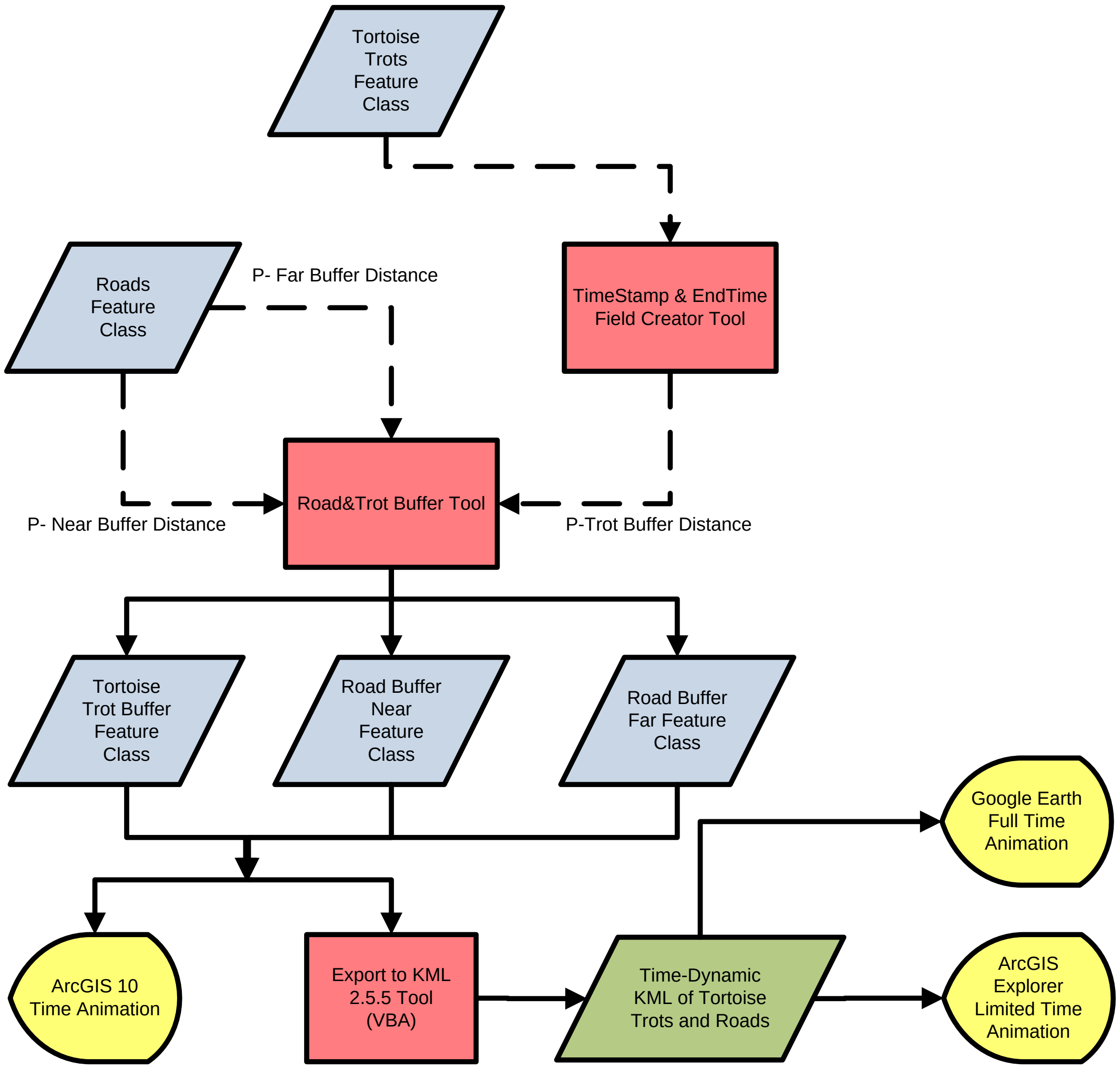

Tortoise Trots Feature Class
Roads Feature Class
P- Far Buffer Distance
TimeStamp & EndTime Field Creator Tool
Road&Trot Buffer Tool
P- Near Buffer Distance
P-Trot Buffer Distance
Tortoise Trot Buffer Feature Class
Road Buffer Near Feature Class
Road Buffer Far Feature Class
Google Earth Full Time Animation
ArcGIS 10 Time Animation
Export to KML 2.5.5 Tool (VBA)
Time-Dynamic KML of Tortoise Trots and Roads
ArcGIS Explorer Limited Time Animation

Figure 6-3: Flow Diagram, Road and Trot Use-Case

The *TimeStamp and EndTime Field Creator Tool* is first used on the Tortoise Trot feature class to create and populate the TimeStamp attribute with the DATE_ attribute's value. The *Road and Trot Buffer Tool* can then create up to three buffers. The user has the option to choose which buffers are produced. The buffer distances can be changed, as well as their units. For this scenario, the default buffer distances were used and all three buffers were created. Once these feature classes were created, the buffers were clipped using the park limit feature class. This removed the buffer areas outside of the park. As with the other scenarios in this project, the feature classes can be exported into KML for dynamic visualization in Google Earth.

6.2.1.2 Results

The results are depicted in ArcMap (Figure 6-4). The buffers created show that 54.7% of the area within JOTR falls within the far road buffer of 3.5 kilometers, and 13.4% fall within the near road buffer of 0.8 kilometers. It can be seen in the visualization that there are many areas that appear to be protected by the curbing; however, it is more apparent where gaps between curbs open up considerable dangerous area.

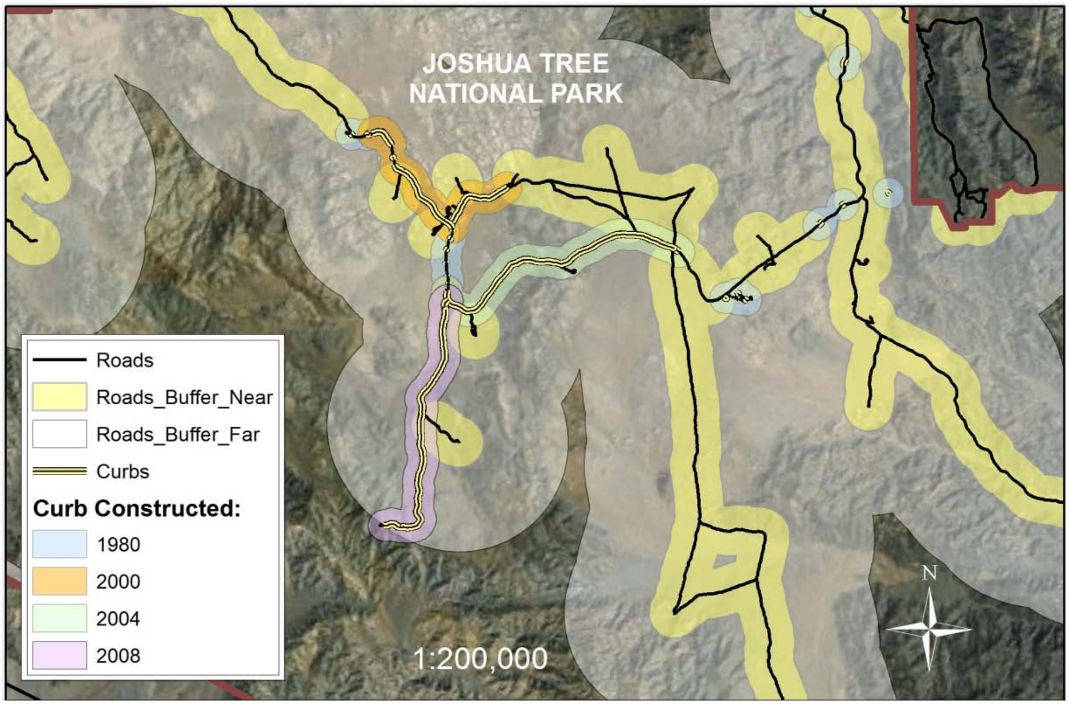


Figure 6-4: Road and Curbing Buffer Results Shown in ArcMap

The output becomes particularly useful when it is combined with dynamic movements of tortoises. By visualizing the protected areas, roads, curbing, and dangerous zones with tortoise trajectory, we can see whether a tortoise passes within or near a buffer that is considered dangerous. This can be used to help determine curb implementation effectiveness and needs. The positions of tortoises found dead can be added to the display to help determine if the probable cause of death may be related to the tortoise's proximity to these areas.

6.2.2 Where have the tortoises been?

Having recorded telemetry data of 18 tortoises over the past five years, the JOTR staff is interested in visualizing the movement of these tortoises and exploring their paths in relation to the physical environment in the park. With the custom tools *TimeStamp and EndTime Field Creator Tool* and *Tortoise Import Tool*, they now can easily view the dynamic trajectories of the tortoises.

6.2.2.1 Process

The workflow of this process is illustrated in Figure 6-5. Initially, the *TimeStamp and EndTime Field Creator Tool* was run on the All_Tortoises feature class to create and populate the TimeStamp and EndTime fields. Once this was completed, the *Tortoise Import Tool* was used to create all of the necessary sub-feature classes for animation. These feature classes were brought into ArcMap. These feature classes were each exported to KML twice. The first time, the feature class is exported with both the TimeStamp and EndTime fields with the tortoise symbol. The second time, the feature class is exported with only the TimeStamp field and symbolized with a black circle representing a footprint. This results in the tortoise leaving behind a footprint as he moves through time.

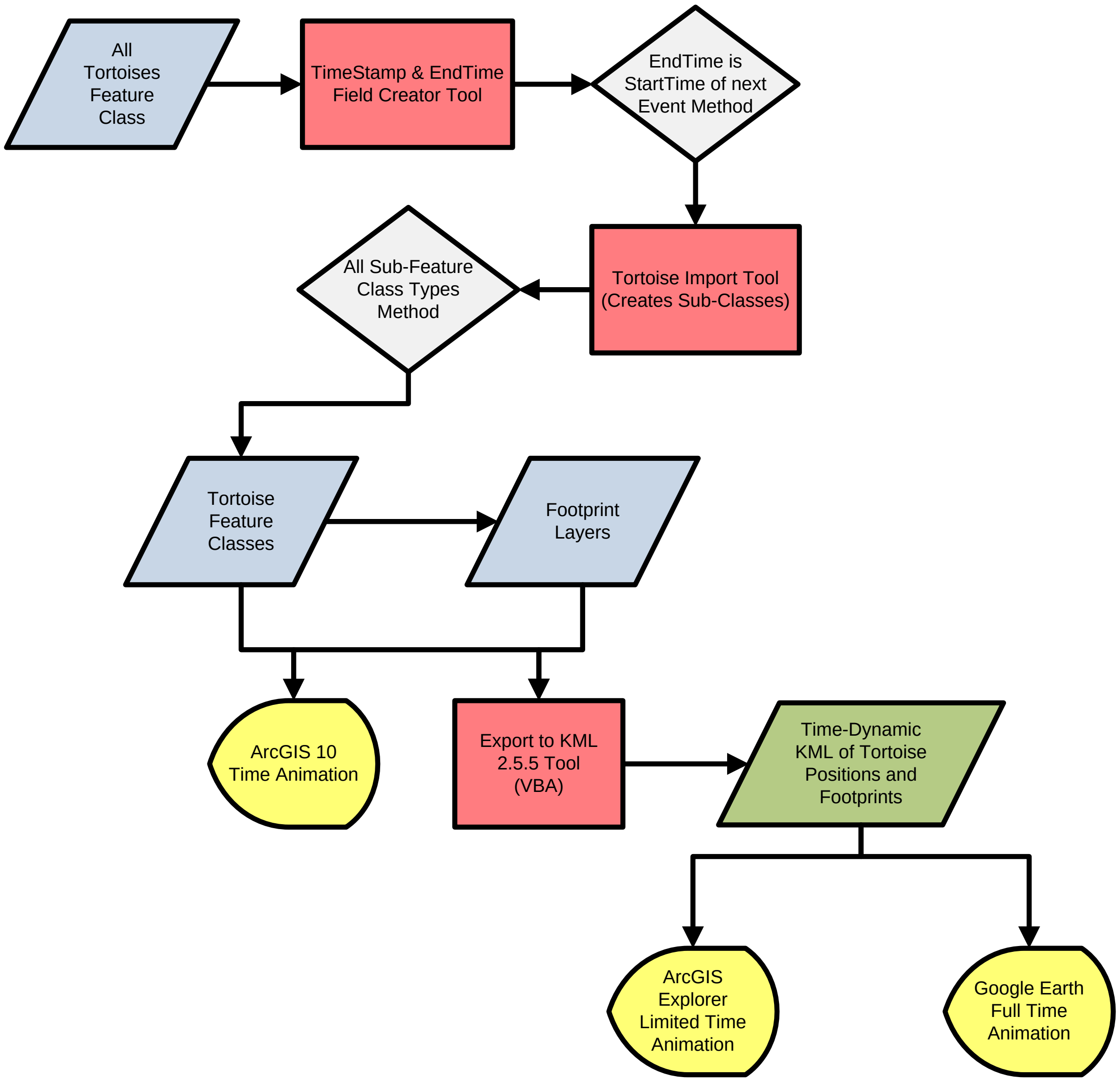
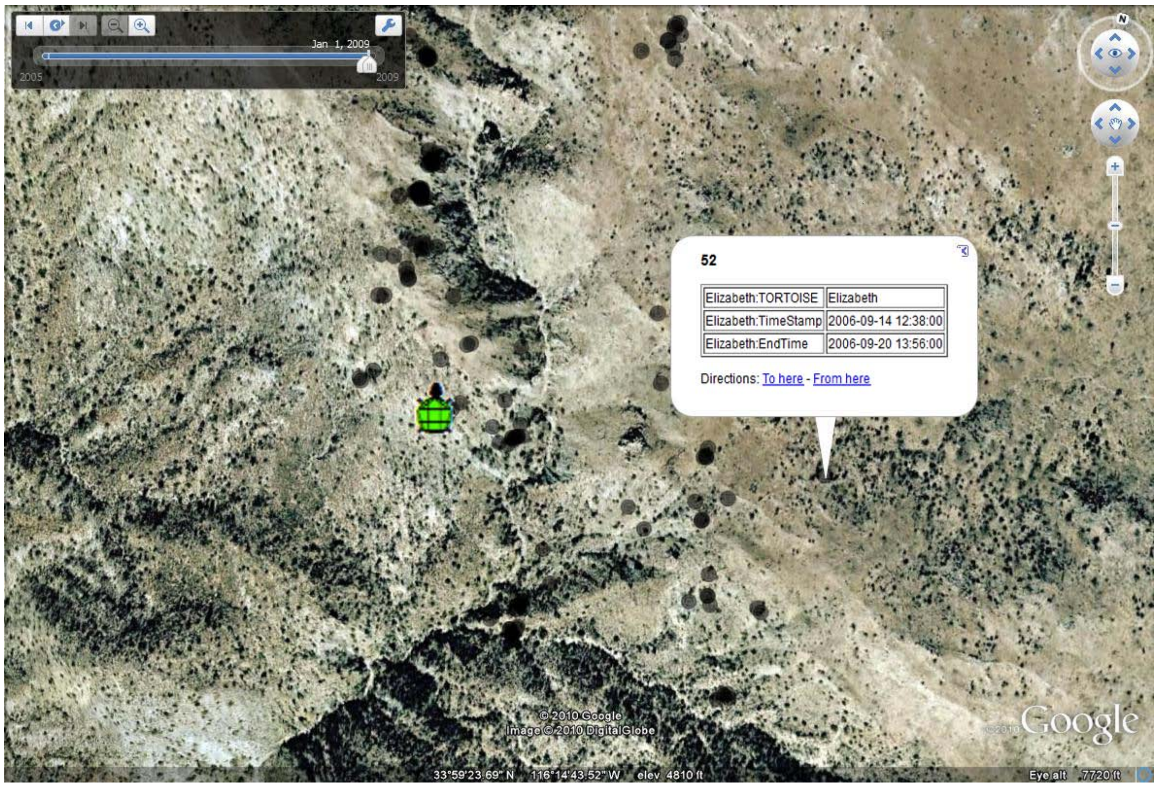

All Tortoises Feature Class
TimeStamp & EndTime Field Creator Tool
EndTime is StartTime of next Event Method
Tortoise Import Tool (Creates Sub-Classes)
All Sub-Feature Class Types Method
Tortoise Feature Classes
Footprint Layers
ArcGIS 10 Time Animation
Export to KML 2.5.5 Tool (VBA)
Time-Dynamic KML of Tortoise Positions and Footprints
ArcGIS Explorer Limited Time Animation
Google Earth Full Time Animation

Figure 6-5: Flow Diagram, Tortoise Movement

6.2.2.2 Results

The results shown in Google Earth depict the current tortoise position, as well as icons indicating the locations that the tortoise has previously visited. When animated in Google Earth, the icons show up based upon their TimeStamp. Figure 6-6 shows the footprint of the tortoise called Elizabeth, which includes 207 recorded positions over 43 months. All icons are clickable, producing a pop-up balloon that contains the tortoise name and TimeStamp attributes. The pop-up balloon in the figure points to one of Elizabeth's many footprints. By giving the footprint symbols 50% opacity, a sense of density is observed. Where footprints overlap, the color is darker. This can help to determine possible tortoise burrows, or favorite spots. They also could indicate a pattern of behavior, or normal extent for the tortoise.


Jan 1, 2009
2005
2009
© 2010 Google
Image © 2010 DigitalGlobe
33°59'23.69" N 116°14'43.52" W elev 4810 ft
Eye alt 7720 ft
Google

52

Elizabeth:TORTOISE	Elizabeth
Elizabeth:TimeStamp	2006-09-14 12:38:00
Elizabeth:EndTime	2006-09-20 13:56:00

Directions: To here - From here

Figure 6-6: Animated Tortoise Movement in Google Earth.

Whether it is one or all of the tortoises, they can be animated in Google Earth. Since a tortoise icon indicates the current position, the user can see the tortoise location in respect to another tortoise. Since the tortoise symbol represents the current position, it may be easier to observe possible tortoise interactions as tortoise symbols group together. The locations can be shown with the road and curb buffers, as well as other features such as the dynamic weather features discussed in Section 6.2.7, to see if there are dangers or correlation to weather conditions.

6.2.3 How fast are the tortoises?

For the project, many of the tools depend on knowing the maximum achieved speed of each tortoise. Knowing the maximum moving speed does not meet the needs of the tools, as a tortoise can have a faster speed if recorded while the tortoise moving in a short time. Since it is not known where the tortoise went between any two consecutive positions, a maximum achieved speed was obtained by using the Euclidean distance between the two consecutive positions divided by the lapse in time. The tool can also calculate the maximum achieved speed for each tortoise by year, with which the client can further examine the speed variation of tortoise population over years. The tool does not consider speeds that were achieved in a duration of less than 18 hours. This eliminates high speeds being recorded for tortoises that were relocated within 18 hours. This effect can be seen in the graph in Figure 6-7, in which a speed of 108.6 was calculated after 49 minutes.

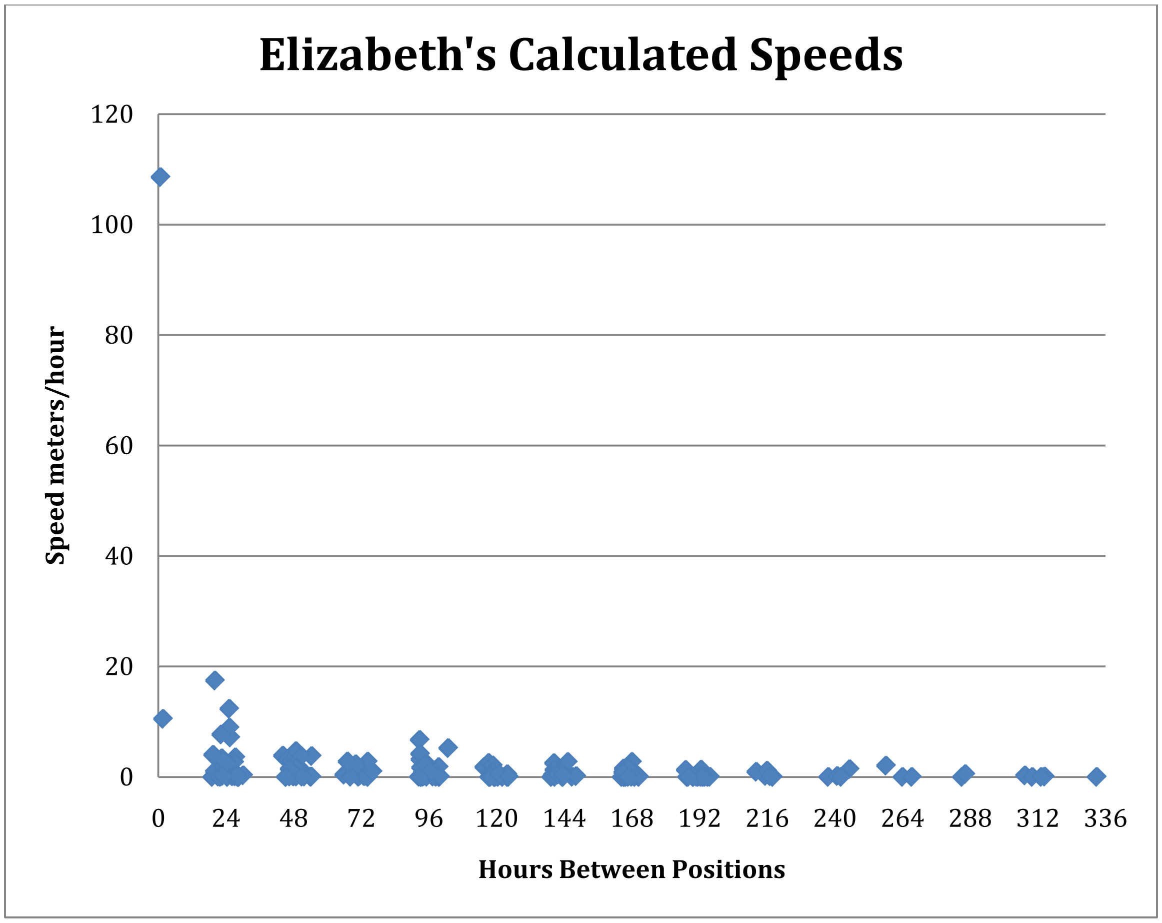


Figure 6-7: Elizabeth's Calculated Speeds

6.2.3.1 Process

The workflow to address this question is summarized in Figure 6-8, in which the *TimeStamp and EndTime Field Creator Tool* and the Tortoise Maximum Speed Finder are the tools used. The light blue color in the figure represents a decision needs to be made by selecting an option in the tool. The output of the *Tortoise Maximum Speed Finder Tool* is an ArcMap table. The maximum achieved speed is calculated in meters per hour.

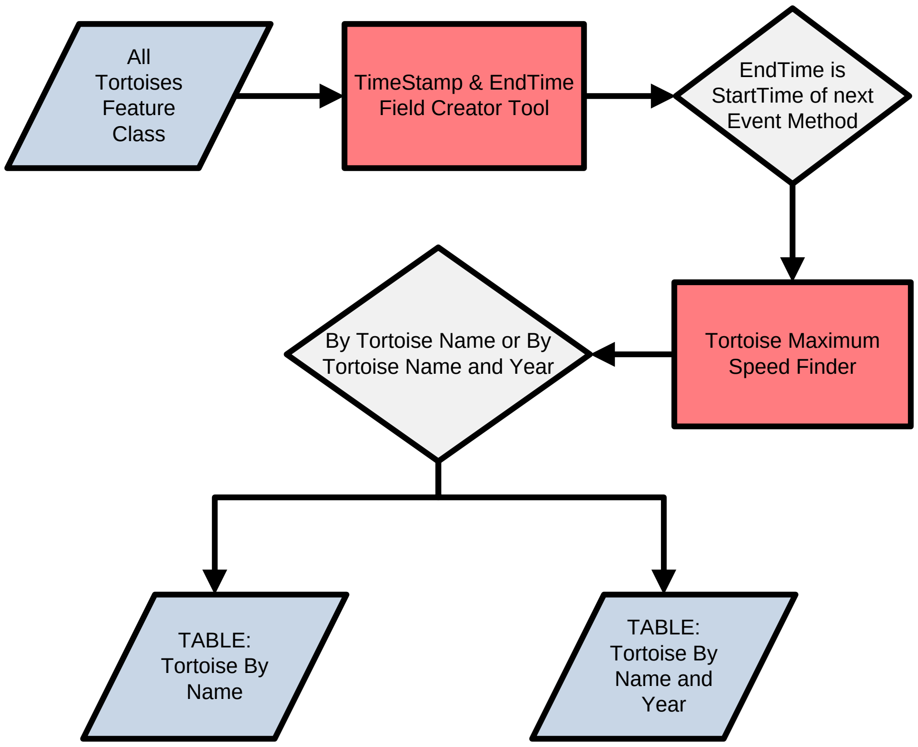
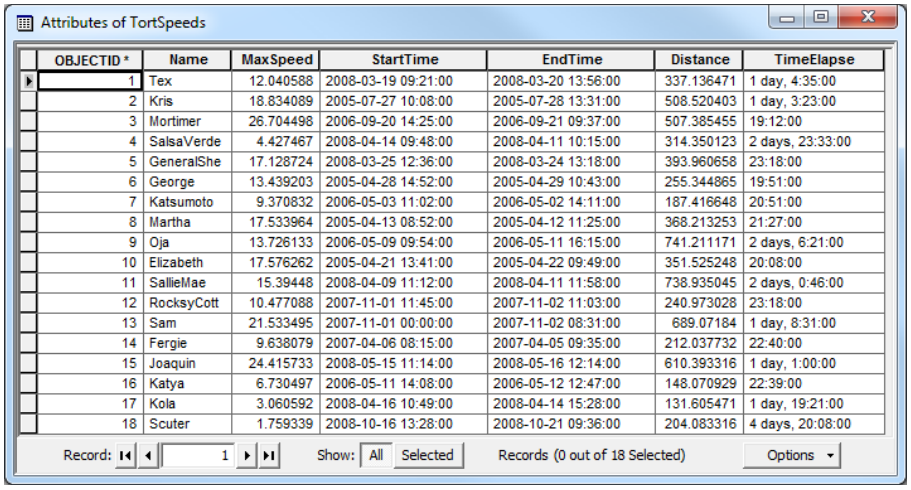

All Tortoises Feature Class
TimeStamp & EndTime Field Creator Tool
EndTime is StartTime of next Event Method
Tortoise Maximum Speed Finder
By Tortoise Name or By Tortoise Name and Year
TABLE: Tortoise By Name
TABLE: Tortoise By Name and Year

Figure 6-8: Flow Diagram, Tortoise Speeds

6.2.3.2 Results

This process was run on the All_Tortoises feature class that had been previously been processed by the *TimeStamp and EndTime Field Creator Tool*. The process was run twice to create the two different types of output tables: Tortoise By Name table that contains the maximum achieved speed of each tortoise and when that speed was achieved during the past four years (Figure 6-9), and Tortoise By Name and Year table including the maximum achieved speed of each tortoise in each year (Figure 6-10).

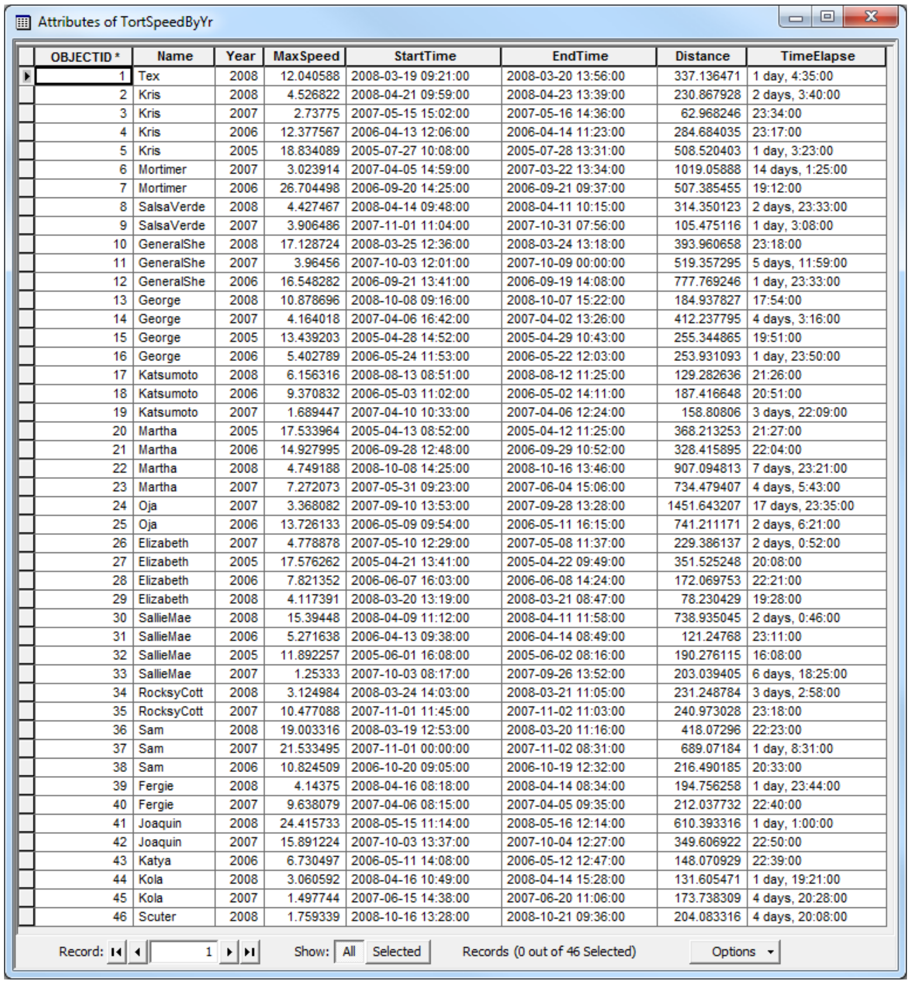


Attributes of TortSpeeds

OBJECTID *	Name	MaxSpeed	StartTime	EndTime	Distance	TimeElapse
1	Tex	12.040588	2008-03-19 09:21:00	2008-03-20 13:56:00	337.136471	1 day, 4:35:00
2	Kris	18.834089	2005-07-27 10:08:00	2005-07-28 13:31:00	508.520403	1 day, 3:23:00
3	Mortimer	26.704498	2006-09-20 14:25:00	2006-09-21 09:37:00	507.385455	19:12:00
4	SalsaVerde	4.427467	2008-04-14 09:48:00	2008-04-11 10:15:00	314.350123	2 days, 23:33:00
5	GeneralShe	17.128724	2008-03-25 12:36:00	2008-03-24 13:18:00	393.960658	23:18:00
6	George	13.439203	2005-04-28 14:52:00	2005-04-29 10:43:00	255.344865	19:51:00
7	Katsumoto	9.370832	2006-05-03 11:02:00	2006-05-02 14:11:00	187.416648	20:51:00
8	Martha	17.533964	2005-04-13 08:52:00	2005-04-12 11:25:00	368.213253	21:27:00
9	Oja	13.726133	2006-05-09 09:54:00	2006-05-11 16:15:00	741.211171	2 days, 6:21:00
10	Elizabeth	17.576262	2005-04-21 13:41:00	2005-04-22 09:49:00	351.525248	20:08:00
11	SallieMae	15.39448	2008-04-09 11:12:00	2008-04-11 11:58:00	738.935045	2 days, 0:46:00
12	RocksyCott	10.477088	2007-11-01 11:45:00	2007-11-02 11:03:00	240.973028	23:18:00
13	Sam	21.533495	2007-11-01 00:00:00	2007-11-02 08:31:00	689.07184	1 day, 8:31:00
14	Fergie	9.638079	2007-04-06 08:15:00	2007-04-05 09:35:00	212.037732	22:40:00
15	Joaquin	24.415733	2008-05-15 11:14:00	2008-05-16 12:14:00	610.393316	1 day, 1:00:00
16	Katya	6.730497	2006-05-11 14:08:00	2006-05-12 12:47:00	148.070929	22:39:00
17	Kola	3.060592	2008-04-16 10:49:00	2008-04-14 15:28:00	131.605471	1 day, 19:21:00
18	Scuter	1.759339	2008-10-16 13:28:00	2008-10-21 09:36:00	204.083316	4 days, 20:08:00

Record: 1 Show: All Selected Records (0 out of 18 Selected) Options

Figure 6-9: Tortoise by Name Table



Attributes of TortSpeedByYr

OBJECTID *	Name	Year	MaxSpeed	StartTime	EndTime	Distance	TimeElapse
1	Tex	2008	12.040588	2008-03-19 09:21:00	2008-03-20 13:56:00	337.136471	1 day, 4:35:00
2	Kris	2008	4.526822	2008-04-21 09:59:00	2008-04-23 13:39:00	230.867928	2 days, 3:40:00
3	Kris	2007	2.73775	2007-05-15 15:02:00	2007-05-16 14:36:00	62.968246	23:34:00
4	Kris	2006	12.377567	2006-04-13 12:06:00	2006-04-14 11:23:00	284.684035	23:17:00
5	Kris	2005	18.834089	2005-07-27 10:08:00	2005-07-28 13:31:00	508.520403	1 day, 3:23:00
6	Mortimer	2007	3.023914	2007-04-05 14:59:00	2007-03-22 13:34:00	1019.05888	14 days, 1:25:00
7	Mortimer	2006	26.704498	2006-09-20 14:25:00	2006-09-21 09:37:00	507.385455	19:12:00
8	SalsaVerde	2008	4.427467	2008-04-14 09:48:00	2008-04-11 10:15:00	314.350123	2 days, 23:33:00
9	SalsaVerde	2007	3.906486	2007-11-01 11:04:00	2007-10-31 07:56:00	105.475116	1 day, 3:08:00
10	GeneralShe	2008	17.128724	2008-03-25 12:36:00	2008-03-24 13:18:00	393.960658	23:18:00
11	GeneralShe	2007	3.96456	2007-10-03 12:01:00	2007-10-09 00:00:00	519.357295	5 days, 11:59:00
12	GeneralShe	2006	16.548282	2006-09-21 13:41:00	2006-09-19 14:08:00	777.769246	1 day, 23:33:00
13	George	2008	10.878696	2008-10-08 09:16:00	2008-10-07 15:22:00	184.937827	17:54:00
14	George	2007	4.164018	2007-04-06 16:42:00	2007-04-02 13:26:00	412.237795	4 days, 3:16:00
15	George	2005	13.439203	2005-04-28 14:52:00	2005-04-29 10:43:00	255.344865	19:51:00
16	George	2006	5.402789	2006-05-24 11:53:00	2006-05-22 12:03:00	253.931093	1 day, 23:50:00
17	Katsumoto	2008	6.156316	2008-08-13 08:51:00	2008-08-12 11:25:00	129.282636	21:26:00
18	Katsumoto	2006	9.370832	2006-05-03 11:02:00	2006-05-02 14:11:00	187.416648	20:51:00
19	Katsumoto	2007	1.689447	2007-04-10 10:33:00	2007-04-06 12:24:00	158.80806	3 days, 22:09:00
20	Martha	2005	17.533964	2005-04-13 08:52:00	2005-04-12 11:25:00	368.213253	21:27:00
21	Martha	2006	14.927995	2006-09-28 12:48:00	2006-09-29 10:52:00	328.415895	22:04:00
22	Martha	2008	4.749188	2008-10-08 14:25:00	2008-10-16 13:46:00	907.094813	7 days, 23:21:00
23	Martha	2007	7.272073	2007-05-31 09:23:00	2007-06-04 15:06:00	734.479407	4 days, 5:43:00
24	Oja	2007	3.368082	2007-09-10 13:53:00	2007-09-28 13:28:00	1451.643207	17 days, 23:35:00
25	Oja	2006	13.726133	2006-05-09 09:54:00	2006-05-11 16:15:00	741.211171	2 days, 6:21:00
26	Elizabeth	2007	4.778878	2007-05-10 12:29:00	2007-05-08 11:37:00	229.386137	2 days, 0:52:00
27	Elizabeth	2005	17.576262	2005-04-21 13:41:00	2005-04-22 09:49:00	351.525248	20:08:00
28	Elizabeth	2006	7.821352	2006-06-07 16:03:00	2006-06-08 14:24:00	172.069753	22:21:00
29	Elizabeth	2008	4.117391	2008-03-20 13:19:00	2008-03-21 08:47:00	78.230429	19:28:00
30	SallieMae	2008	15.39448	2008-04-09 11:12:00	2008-04-11 11:58:00	738.935045	2 days, 0:46:00
31	SallieMae	2006	5.271638	2006-04-13 09:38:00	2006-04-14 08:49:00	121.24768	23:11:00
32	SallieMae	2005	11.892257	2005-06-01 16:08:00	2005-06-02 08:16:00	190.276115	16:08:00
33	SallieMae	2007	1.25333	2007-10-03 08:17:00	2007-09-26 13:52:00	203.039405	6 days, 18:25:00
34	RocksyCott	2008	3.124984	2008-03-24 14:03:00	2008-03-21 11:05:00	231.248784	3 days, 2:58:00
35	RocksyCott	2007	10.477088	2007-11-01 11:45:00	2007-11-02 11:03:00	240.973028	23:18:00
36	Sam	2008	19.003316	2008-03-19 12:53:00	2008-03-20 11:16:00	418.07296	22:23:00
37	Sam	2007	21.533495	2007-11-01 00:00:00	2007-11-02 08:31:00	689.07184	1 day, 8:31:00
38	Sam	2006	10.824509	2006-10-20 09:05:00	2006-10-19 12:32:00	216.490185	20:33:00
39	Fergie	2008	4.14375	2008-04-16 08:18:00	2008-04-14 08:34:00	194.756258	1 day, 23:44:00
40	Fergie	2007	9.638079	2007-04-06 08:15:00	2007-04-05 09:35:00	212.037732	22:40:00
41	Joaquin	2008	24.415733	2008-05-15 11:14:00	2008-05-16 12:14:00	610.393316	1 day, 1:00:00
42	Joaquin	2007	15.891224	2007-10-03 13:37:00	2007-10-04 12:27:00	349.606922	22:50:00
43	Katya	2006	6.730497	2006-05-11 14:08:00	2006-05-12 12:47:00	148.070929	22:39:00
44	Kola	2008	3.060592	2008-04-16 10:49:00	2008-04-14 15:28:00	131.605471	1 day, 19:21:00
45	Kola	2007	1.497744	2007-06-15 14:38:00	2007-06-20 11:06:00	173.738309	4 days, 20:28:00
46	Scuter	2008	1.759339	2008-10-16 13:28:00	2008-10-21 09:36:00	204.083316	4 days, 20:08:00

Record: 1 Show: All Selected Records (0 out of 46 Selected) Options

Figure 6-10: Tortoise by Name and Year Table

By analyzing the results in the tables created in ArcMap, the fastest tortoise, Mortimer, achieved 26.7 meters per hour, and the slowest tortoise, Scuter, only achieved 1.8 meters per hour at his fastest speed. The mean speed for all of the tortoises was 13.5 meters per hour. The tortoise by name and year table was created for analysis and was not used to support any of the other scenarios. The values in the table were used to calculate the average speed per year for the tortoises. The tortoise average speed per year was calculated within Microsoft Excel to output a graph (Figure 6-11). This graph shows that the slowest average speed over the four years was 2007. This graph could be compared to the results of other analysis, such as weather, to determine if there is a correlation. It was documented in JOTR that "2007 was a year of extremely low annual plant productivity which is reflected in reduced home ranges" (Joshua Tree National Park, 2008). Was the slow speed in 2007 due to lack of energy, lack of moisture, or possibly no food in the surrounding area? The introduced variable of tortoise average speed/year could be studied further in relation to these other factors.

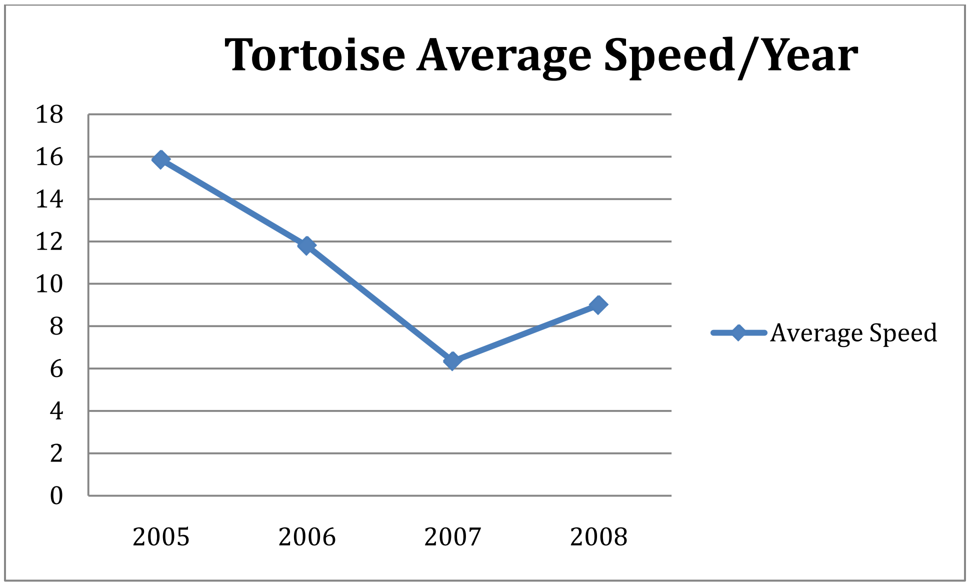


Figure 6-11: Tortoise Average Speed/Year

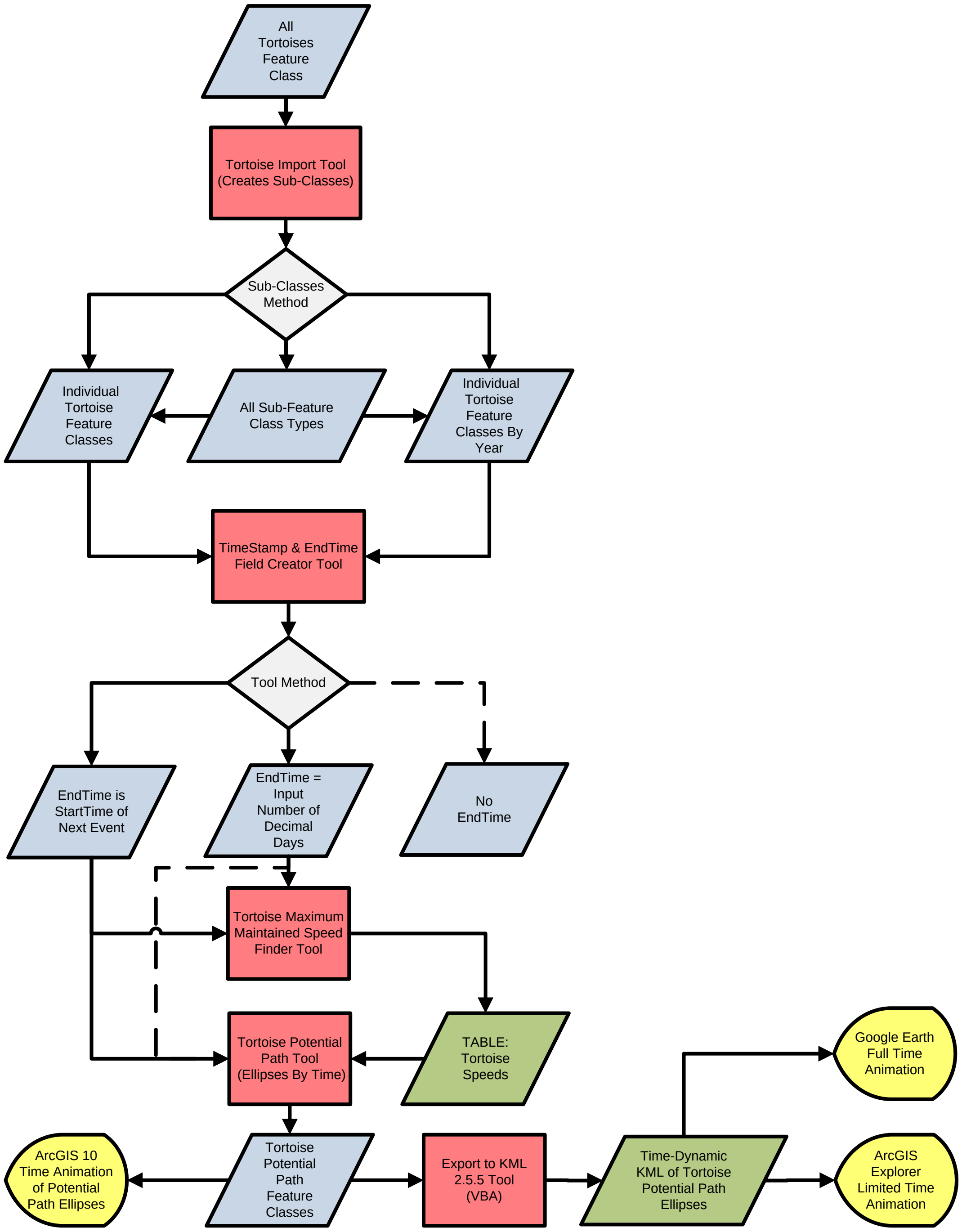
6.2.4 What area could a tortoise cover between positions?

The recorded positions of the desert tortoises are at most recorded daily, unless the tortoise has been relocated. Because of reasons such as inclement weather, minimal workforce, or the inability to locate a tortoise on any given day, there may be gaps anywhere from one day to about two weeks between positions. Therefore, the potential extent of ground covered by the tortoise between recorded positions is unknown. However, the potential path areas are helpful to determine the tortoise habitat range and can reveal the relationship between the tortoise trajectories and roads. Therefore, the *Tortoise Potential Path Tool* provides a means to create ellipses between consecutive positions that indicate the possible ground covered.

6.2.4.1 Process

To find out the tortoise’s potentially travelled area, many of the custom tools were used (Figure 6-12). Except for the *Tortoise Potential Path Tool*, all of the tools have been described in previous use-cases. This process uses the tortoise speed table or a user-entered speed. If the speed entered by the user is smaller than the speed calculated in the table, the potential distance travelled between two consecutive recorded times can be less than the actual distance. In those cases, ellipses will be generated between the positions with a one-meter semi-minor axis.

There are two output types to choose from when using the tool. The default output is for unlimited potential path ellipses. This is found by using the maximum speed of the tortoise and the time difference between consecutive positions. However, this approach can result in unrealistically large ellipses when there are large time lapses between the positions. For example, a tortoise may not move much during two weeks, while the potential ellipse for that time frame is very large. To account for this issue, a limited potential path ellipse option was added. This requires another two parameters: a maximum amount of time between positions and a minimum distance in meters that the tortoise has to move. For example, using the default parameter values of hours equals 36 and distance equals 21 meters, if the time gap is larger than 36 hours and the tortoise has moved less than 21 meters, then the maximum speed will simply be multiplied by one day or 24 hours. This minimizes the size of the ellipses when applicable. If this option is chosen, the parameter values are adjustable.



All Tortoises Feature Class
Tortoise Import Tool (Creates Sub-Classes)
Sub-Classes Method
Individual Tortoise Feature Classes
All Sub-Feature Class Types
Individual Tortoise Feature Classes By Year
TimeStamp & EndTime Field Creator Tool
Tool Method
EndTime is StartTime of Next Event
EndTime = Input Number of Decimal Days
No EndTime
Tortoise Maximum Maintained Speed Finder Tool
Tortoise Potential Path Tool (Ellipses By Time)
TABLE: Tortoise Speeds
Google Earth Full Time Animation
ArcGIS 10 Time Animation of Potential Path Ellipses
Tortoise Potential Path Feature Classes
Export to KML 2.5.5 Tool (VBA)
Time-Dynamic KML of Tortoise Potential Path Ellipses
ArcGIS Explorer Limited Time Animation

Figure 6-12: Flow Diagram, Tortoise Potential Path

6.2.4.2 Results

The results of the tool can be seen in ArcMap (Figure 6-13). The yellow ellipse represents the potential path through which the tortoise will be moving. The gray ellipses represent the other potential path ellipses.

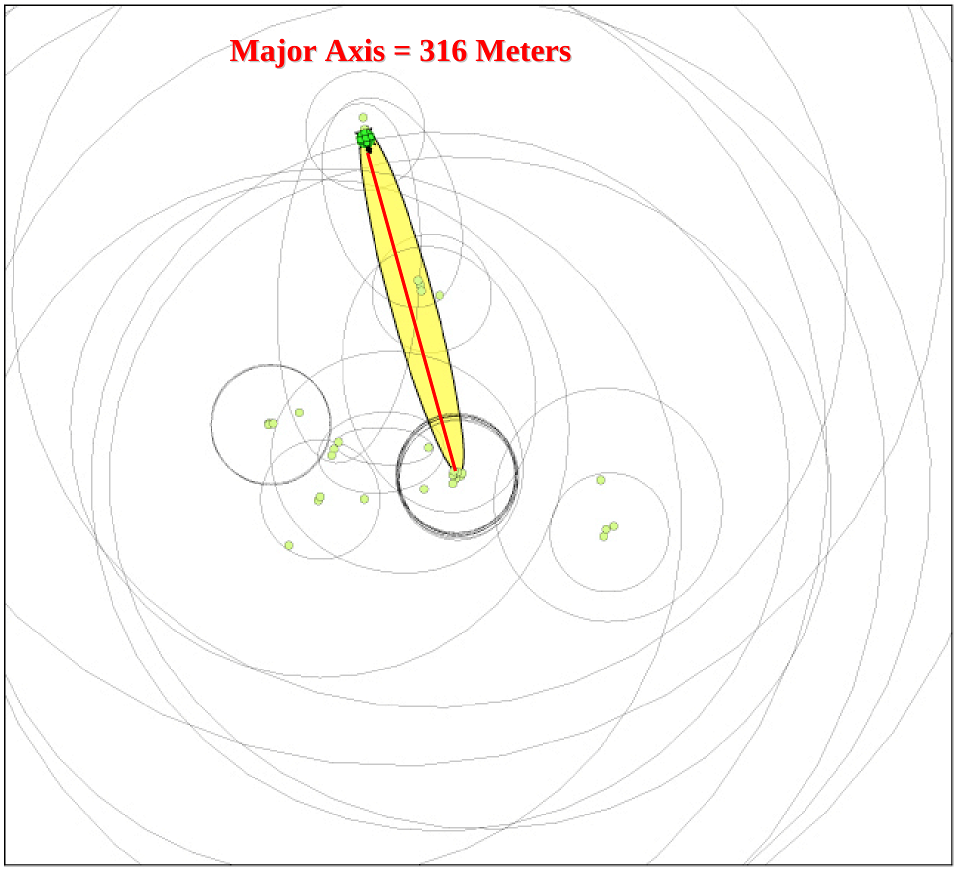


Figure 6-13: Tortoise Potential Paths in ArcMap

The resulting feature classes were also exported to KML for viewing in Google Earth (Figure 6-14). The feature classes were exported twice, one with 60% opacity magenta fill and both TimeStamp and EndTime attributes exported, and the other one with 60% opacity yellow fill and only the TimeStamp attribute exported. When these two feature classes are animated in Google Earth simultaneously, the current ellipse will appear to leave behind footprints of the previous ellipse. This allows the user to see the total possible area that has been covered by previous ellipses.

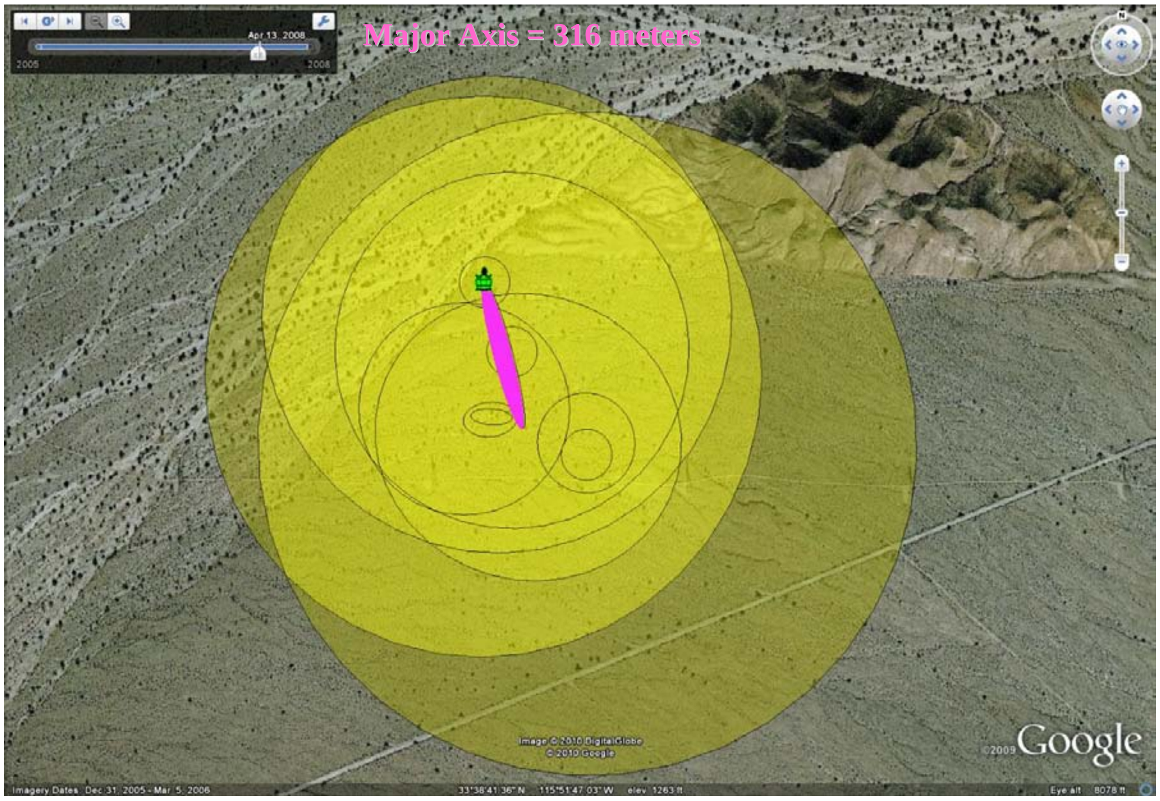


Figure 6-14: Tortoise Potential Paths in Google Earth

When portraying the ellipses based on the tortoise speed table, the ellipses overlapped much of the road buffers. The limited potential path ellipses covered much less road than the unlimited potential path ellipses. The unlimited potential path ellipses covered 59% of JOTR's total area, while the limited potential path ellipses covered only 28% of JOTR's total area. Therefore, if the user went by the unlimited buffer, it would indicate that the tortoises could cover 59% of JOTR, which is unrealistic.

When the limited potential path areas are all merged together or merged by individual tortoises, an estimation of the habitat range of all tortoises or individual tortoises is obtained. With that information, further studies can be conducted to examine the interactions between different tortoises and, most critical, possible interaction with roads. Because of the capability and findings, a new tool was added to the deliverables using ModelBuilder, as described in the following section.

6.2.5 How to prioritize where new curbing should be implemented

JOTR has a need to determine where new tortoise trot curbing needs to be implemented and which sections of road are high priority. The *Priority Trot Needs Tool* was created to address this need. By using the road buffers created by the *Road and Trot Buffer Tool*, and the tortoise potential path ellipses built by the *Tortoise Potential Path Tool*, this tool determines tortoise trot curbing needs and prioritizes them in three categories depending on whether they are covered by limited ellipses, unlimited ellipses, or not covered by any ellipses. This process involved only one model, which was described in Section 5.8. The areas could be calculated differently by the client, based upon their using of different parameters for the limited potential path ellipses.

6.2.5.1 Results

The results of the tool yield three zones depicting the Primary, Secondary, and Tertiary road segments in consideration for future tortoise trots (Figure 6-15). JOTR can use the output of the tool to determine their priority curbing needs (in red), secondary curbing needs (in yellow), and tertiary curbing needs (in green). Using other limiting parameters for tortoise potential path limits, JOTR has the capability to perform the same process with the existing data or any new data that they collect.

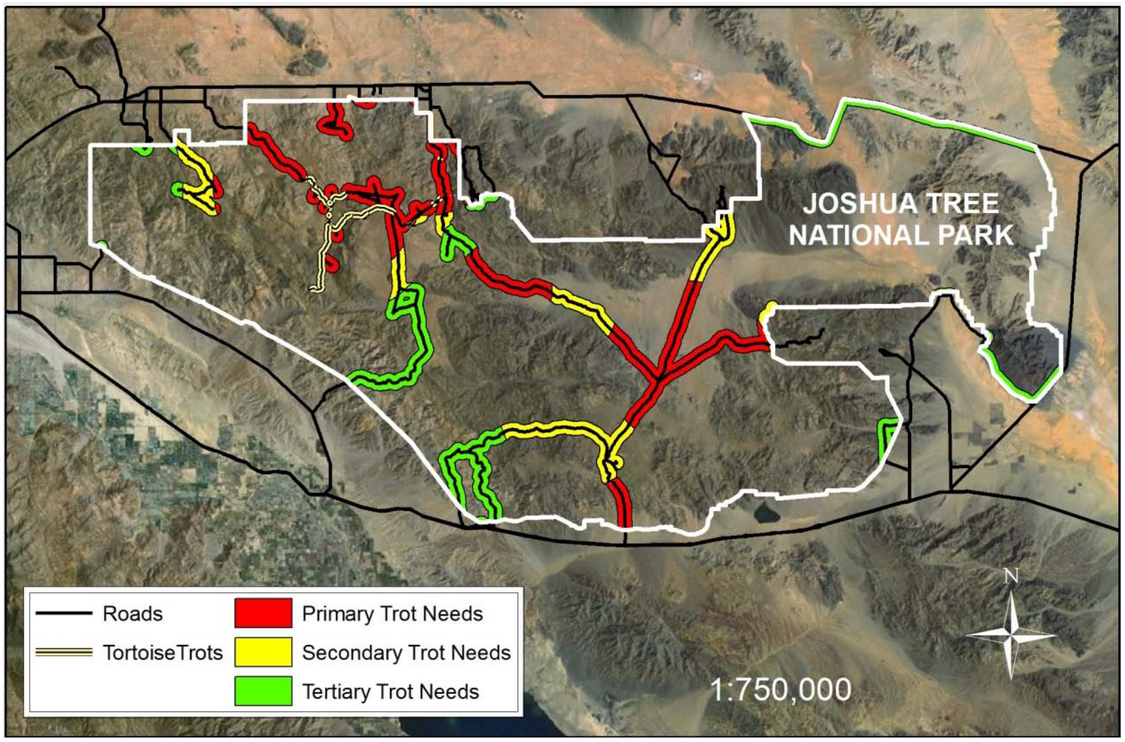


Figure 6-15: Tortoise Trot Needs by Priority

6.2.6 How to locate a lost tortoise

In tracking the tortoises, sometimes they may not be found where they are expected. This can be due to the time elapsed from the last recorded position, the tortoise moving to a further location, or the transmitter malfunctioning. In such situations, the park staff needs an estimation of potential areas that the lost tortoise could visit. To address this issue, the *Lost Tortoise Tool* was built to locate the lost tortoise by considering its maximum achieved speed and the time lapsed.

6.2.6.1 Process

The workflow of finding a lost tortoise is summarized in Figure 6-16. To obtain the result for illustration, a fictitious UTM last known position was entered for Tex the tortoise, as well as the last known position TimeStamp. For the simulation, the Tortoise Speed Table was used, rather than a user-entered speed. The *Lost Tortoise Tool* outputs a feature class for the last known position, called Lost_Tex_LastKnownPos, and a feature class for the search area, called Lost_Tex_Buffer.

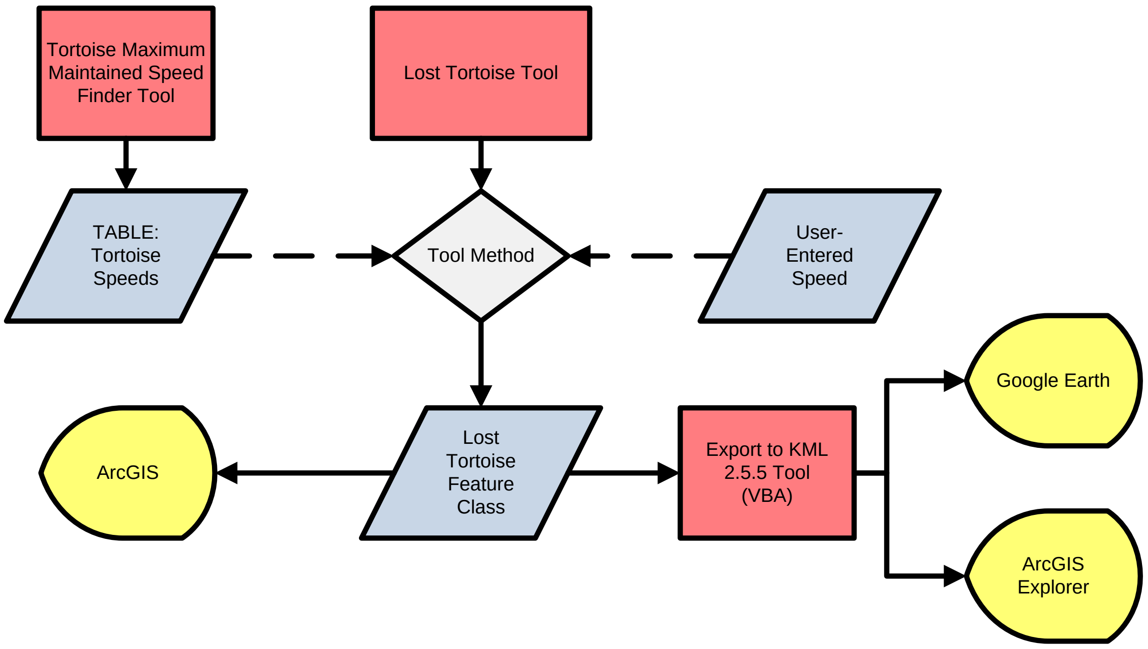
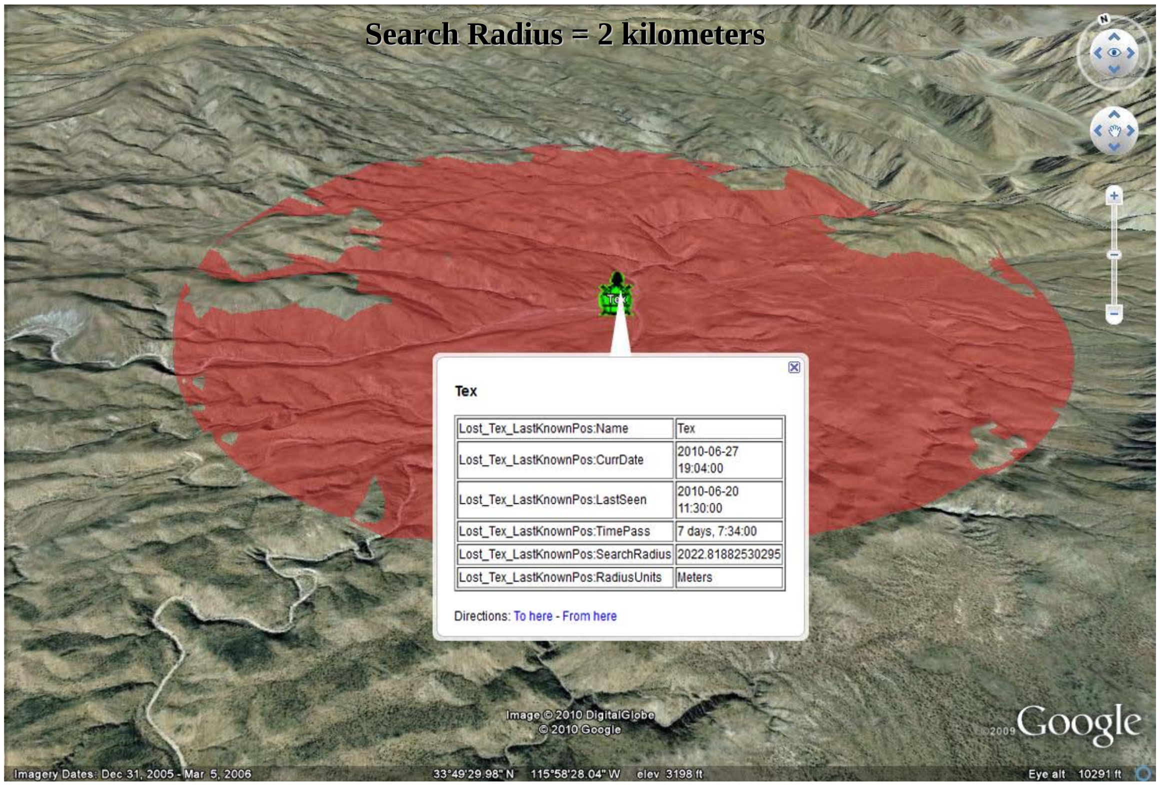

Tortoise Maximum Maintained Speed Finder Tool
Lost Tortoise Tool
TABLE: Tortoise Speeds
Tool Method
User-Entered Speed
Google Earth
ArcGIS
Lost Tortoise Feature Class
Export to KML 2.5.5 Tool (VBA)
ArcGIS Explorer

Figure 6-16: Flow Diagram, Lost Tortoise Tool

6.2.6.2 Results

The last recorded position of the tortoise and the search area are shown in Google Earth (Figure 6-17). The KML symbol for the search area was changed to have an elevation of three meters relative to the ground measured at the center position, rather than clamped to ground. This demonstrates the capability to set a maximum height that the tortoise is able to climb. By changing this value with terrain turned on in Google Earth, the terrain over three meters from the last known position masks out the search area. The attributes can also be exported as part of the *Export to KML Tool*.


Search Radius = 2 kilometers
Tex
Image © 2010 DigitalGlobe
© 2010 Google
Google
Imagery Dates: Dec 31, 2005 - Mar 5, 2006
33°49'29.98" N 115°58'28.04" W elev 3198 ft
Eye alt 10291 ft

Tex

Lost_Tex_LastKnownPos:Name	Tex
Lost_Tex_LastKnownPos:CurrDate	2010-06-27 19:04:00
Lost_Tex_LastKnownPos:LastSeen	2010-06-20 11:30:00
Lost_Tex_LastKnownPos:TimePass	7 days, 7:34:00
Lost_Tex_LastKnownPos:SearchRadius	2022.81882530295
Lost_Tex_LastKnownPos:RadiusUnits	Meters

Directions: To here - From here

Figure 6-17: Lost Tortoise Simulation and Portrayal in Google Earth

These results could be used to narrow the search area for a missing tortoise and provide the metadata related to the tortoise to help locate it.

This tool could be useful not only for locating a lost tortoise, but also for locating other creatures whose last know position is known and the speed can be estimated. This KML format could also easily be shared with members of a search party.

6.2.7 Does a changing weather condition affect tortoise movement

Several articles mention that lack of vegetation or water affects tortoise movement, as tortoises need water and the plants they eat also require water. Indirectly, the temperature has an impact on the tortoise, as it influences the abundance or lack of annual vegetation, as well as evaporation or freezing of water. To examine the relations between the weather changes and the movement of the tortoises, animating both precipitation and temperature amongst the dynamic tortoise movement could provide a starting point. The tortoise animation has already been achieved through the other tools. To animate weather data through time, the *Stationary Event from Table Tool* was created. This tool uses textual precipitation and temperature data, and converts it to animated stationary events.

6.2.7.1 Process

The *Stationary Event from Table Tool* requires use of the *TimeStamp and EndTime Field Creator Tool*. This use-case requires many steps (Figure 6-18).

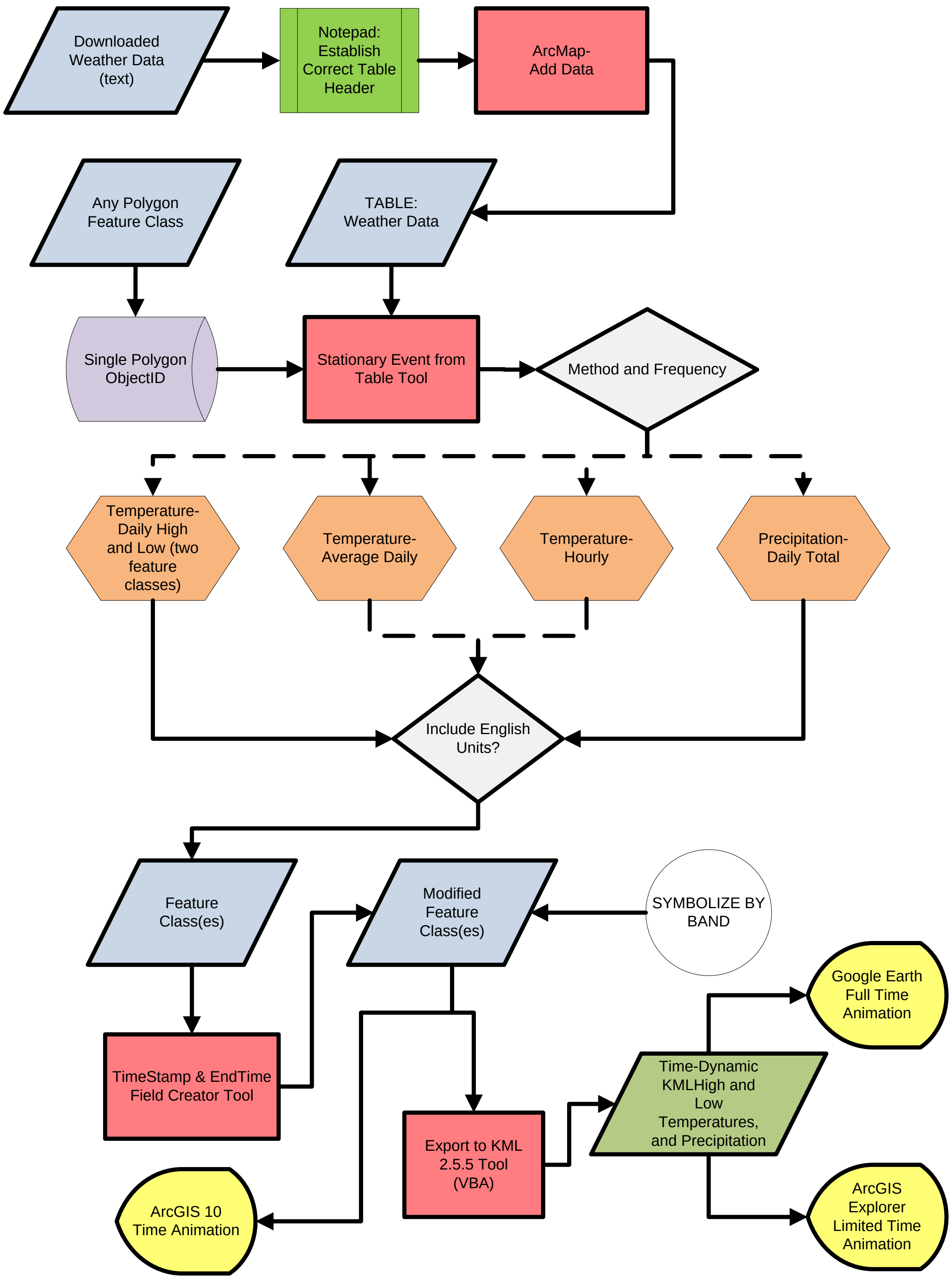

Downloaded Weather Data (text)
Notepad: Establish Correct Table Header
ArcMap- Add Data
Any Polygon Feature Class
TABLE: Weather Data
Single Polygon ObjectID
Stationary Event from Table Tool
Method and Frequency
Temperature- Daily High and Low (two feature classes)
Temperature- Average Daily
Temperature- Hourly
Precipitation- Daily Total
Include English Units?
Feature Class(es)
Modified Feature Class(es)
SYMBOLIZE BY BAND
Google Earth Full Time Animation
TimeStamp & EndTime Field Creator Tool
Time-Dynamic KMLHigh and Low Temperatures, and Precipitation
Export to KML 2.5.5 Tool (VBA)
ArcGIS 10 Time Animation
ArcGIS Explorer Limited Time Animation

Figure 6-18: Flow Diagram, Dynamic Weather Data

The weather data needed to be downloaded first and imported to the geodatabase, as described in Section 4-5. For the project, the high and low temperatures and the precipitation features needed to be present when viewing each tortoise. This called for creating the three feature classes 18 times, once for each tortoise. By zooming into the area of a tortoise, the central tortoise point was selected. The ObjectID for that feature was noted from the attribute table. The feature class and ObjectID are required for the *Stationary Event from Table Tool*.

The *Stationary Event from Table Tool* has five methods to use. For this project, the Temperature-Daily High and Low method was used for the temperature data, and the Precipitation-Daily Total was used for the precipitation data. The Temperature-Daily High and Low method creates both a high and a low temperature feature class.

Once these feature classes were created, they were processed with the *TimeStamp and EndTime Field Creator Tool* using the EndTime is Start of Next Day(s) method and an EndTime Days parameter of one day. Within ArcMap, the feature classes were each symbolized by equal interval color bands. These new feature classes were then exported to KML for Google Earth animation. As part of the export, the appropriate temperature field or precipitation amount field was set as the label. The TimeStamp and EndTime attributes were specified in the tool for export. Both the color and label were animated in Google Earth. To separate the temperature indicators from the precipitation indicator, the high temperature KML export was offset +18 meters on the y-axis, and the low temperature KML export was offset -18 meters.

6.2.7.2 Results

The feature classes that were exported to KML format can be viewed dynamically in Google Earth. By initiating the animation or by manually adjusting the time slider, the three feature classes change color and label value. The high and low temperatures are both color banded from blue to red, representing the temperature. Centered in each symbol is either the high or the low temperature for that day. The precipitation feature is represented by a color band from white to dark blue, representing the total daily precipitation. This feature is only visible on days when there is precipitation. The layout of the features has the feature high temperature on top, followed by precipitation and low temperature (Figure 6-19). By animating the tortoises simultaneously with the weather, one can look for visual relation between tortoise movement and the weather.

This tool would be more useful for animal movement that is recorded at regular intervals without interruption. In altering the time for several of the tortoises, it appears that a tortoise is mostly stationary during rainy or very cold days. However, this is because the weather conditions may be prohibiting the rangers from acquiring positions for those days.

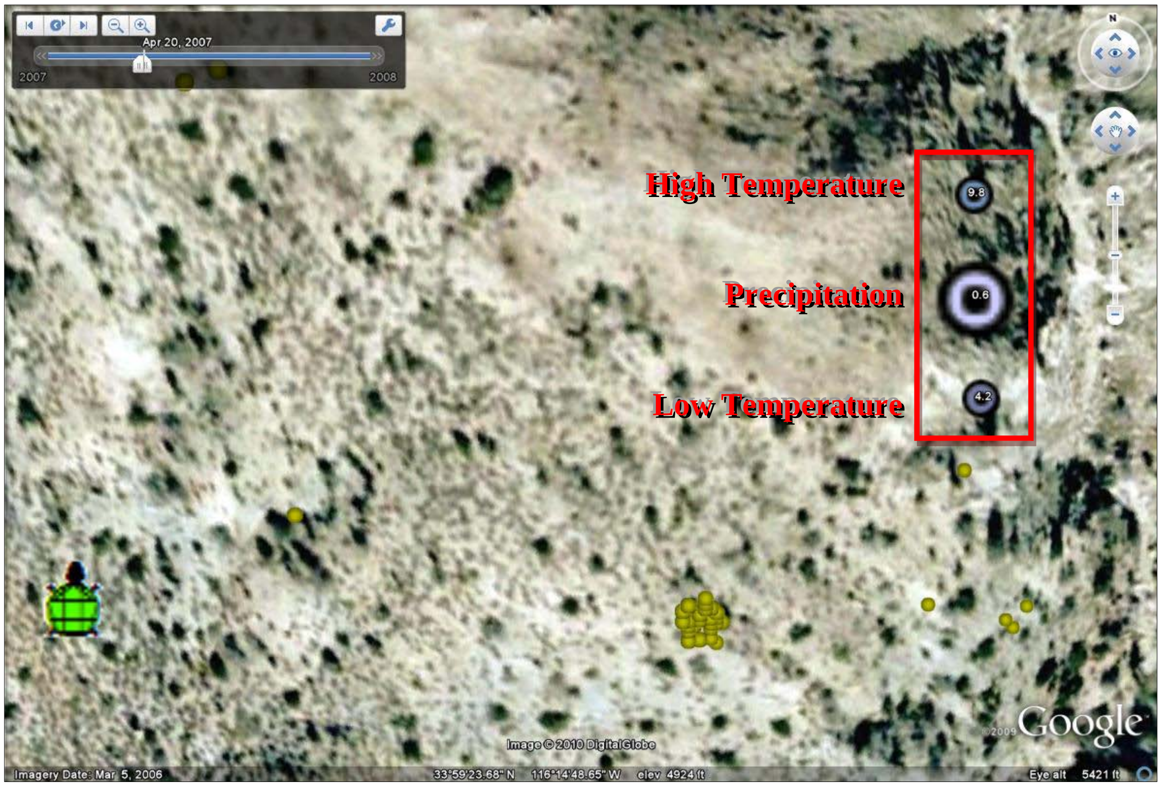


Figure 6-19: Google Earth View of Dynamic Weather Data and Tortoise Positions

6.2.8 Animation of Hawth's Tools results

In addition to the tortoise telemetry data, the client also provided data including Minimum Convex Polygons (MCP), 95 Percent Volume Contours (95PVC), and Kernel Density polygons for each tortoise. These data were created with Hawth's Tools that is a free to download extension for ArcMap. The goals of this project did not include rebuilding the functionality of Hawth's Tools, nor did it require further analyses of these datasets. However, since the data were provided with temporal attributes, it is worth improving the visualization of these data layers with the tools built in the project.

6.2.8.1 Process

The provided data covered the time frame of the study; however, the schema of the provided feature classes needed to be modified for the project. Once the schema was modified, the *TimeStamp and EndTime Field Creator Tool* was run on the feature classes. The feature classes were then exported to KML for animation in Google Earth (Figure 6-20).

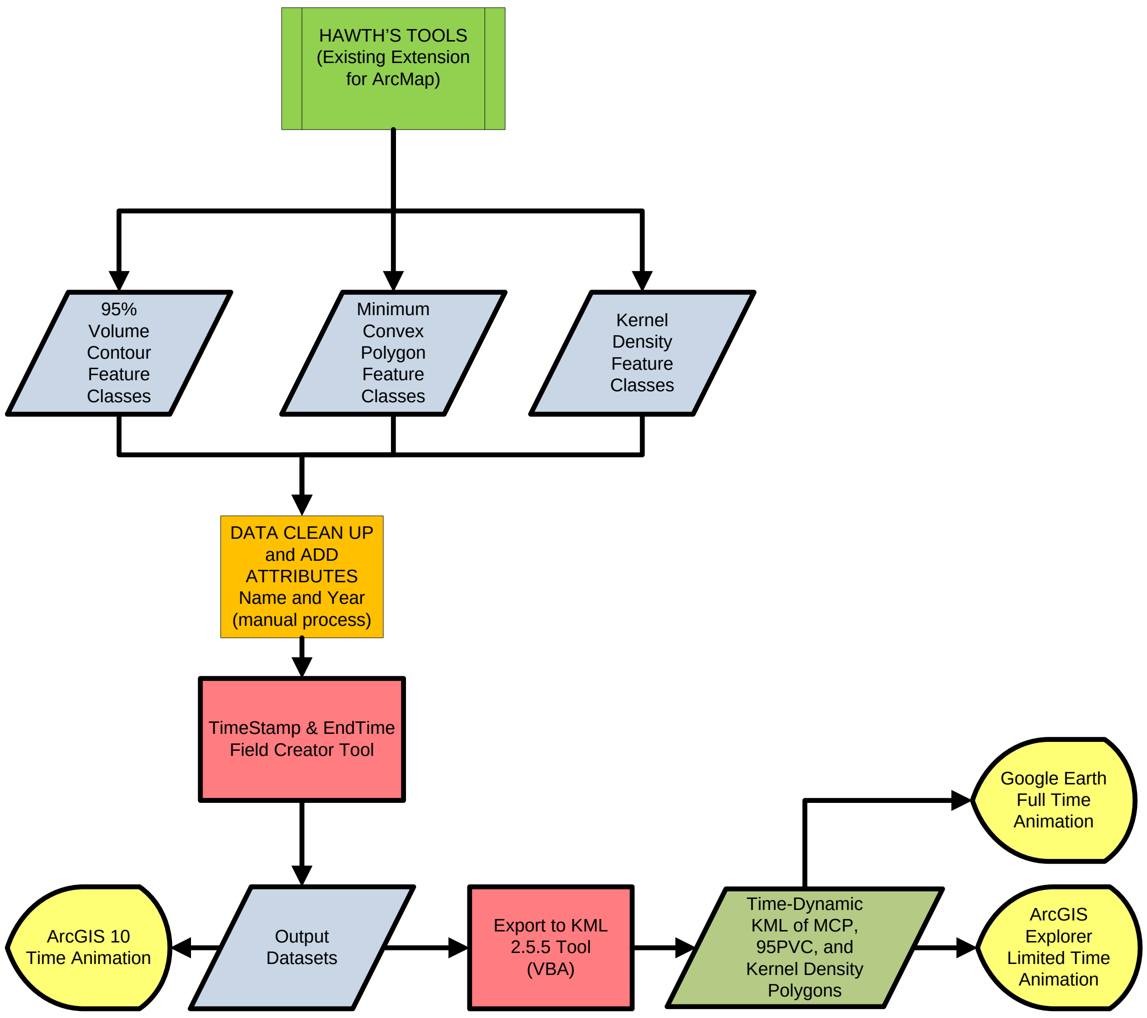

HAWTH'S TOOLS (Existing Extension for ArcMap)
95% Volume Contour Feature Classes
Minimum Convex Polygon Feature Classes
Kernel Density Feature Classes
DATA CLEAN UP and ADD ATTRIBUTES Name and Year (manual process)
TimeStamp & EndTime Field Creator Tool
ArcGIS 10 Time Animation
Output Datasets
Export to KML 2.5.5 Tool (VBA)
Time-Dynamic KML of MCP, 95PVC, and Kernel Density Polygons
Google Earth Full Time Animation
ArcGIS Explorer Limited Time Animation

Figure 6-20: Flow Diagram, Animating Hawth's Tools Results

The original data had the tortoise name, year, or season as part of the file name; however, these values did not exist in the fields of the data. Attribute fields needed to be created and populated with these values. Each individual feature class had to be modified.

The MCP feature classes required creating the name, TimeStamp, and EndTime attributes. The name attribute was populated with the tortoise's name; the time fields had to be manually populated with timestamps matching the season.

The PVC feature classes were simpler, as they covered the entire time of the study. The Kernel Density feature classes were by name and year. These feature classes needed the name field, as well as the TimeStamp and EndTime fields manually populated.

Once modified, all of the MCP feature classes were merged into a single MCP feature class. The 95PVC and Kernel Density feature classes were handed the same way. This created three new single feature classes. The old feature classes were no longer needed.

6.2.8.2 Analysis

The single MCP_All_Tortoises feature class was exported into 18 feature classes by tortoise, using the *Tortoise Import Tool*. These new feature classes were then exported from ArcMap into KML format for animation within Google Earth (Figure 6-21). The pink area in the figure shows the Minimum Convex Polygon for Elizabeth the tortoise.

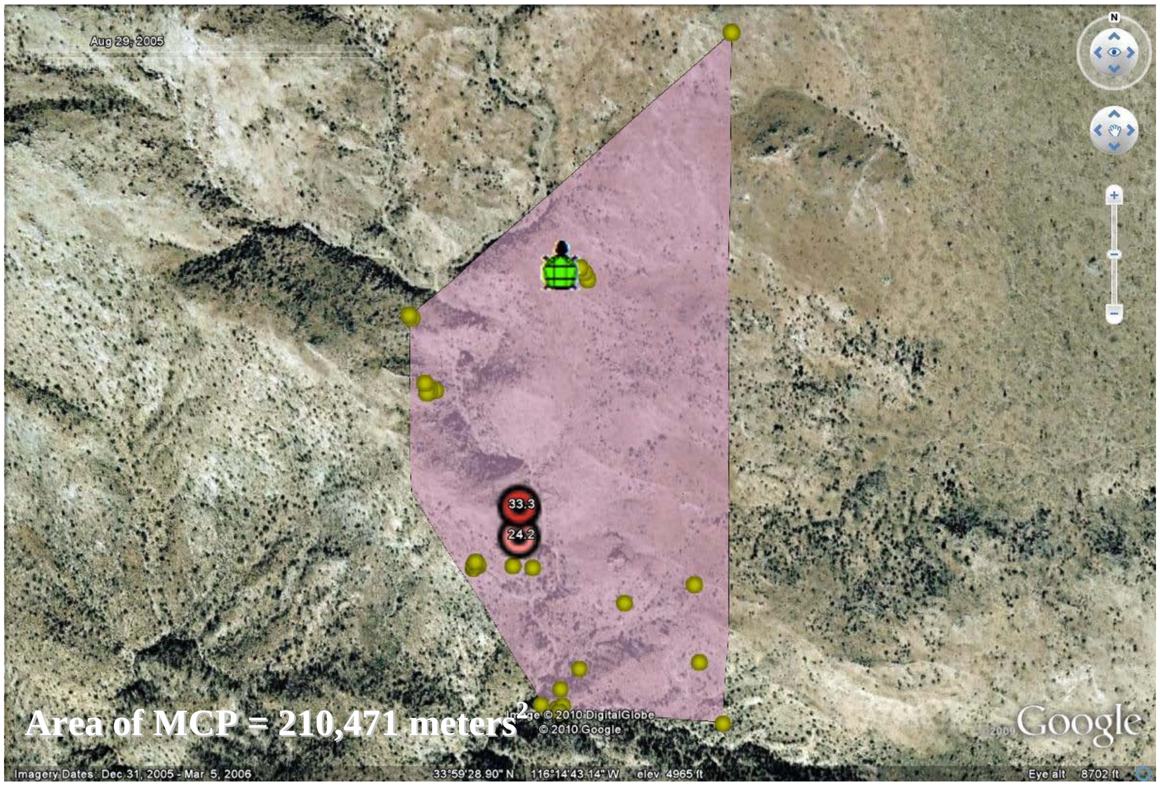

Aug 29, 2005
33.3
24.2
Area of MCP = 210,471 meters²
Image © 2010 DigitalGlobe
© 2010 Google
Google
Imagery Dates: Dec 31, 2005 - Mar 5, 2006
33°59'28.90" N 116°14'43.14" W elev 4965 ft
Eye alt 8702 ft

Figure 6-21: Google Earth View of a Dynamic MCP

The single PVC_All_Tortoises feature class was exported into 18 feature classes by tortoise, using the *Tortoise Import Tool*. These feature classes are not dynamic, as they cover the entire study period. These new feature classes were then easily exported from ArcMap into KML format into Google Earth (Figure 6-22). The pink area in the figure depicts the 95 Percent Volume Contour for Elizabeth the tortoise.

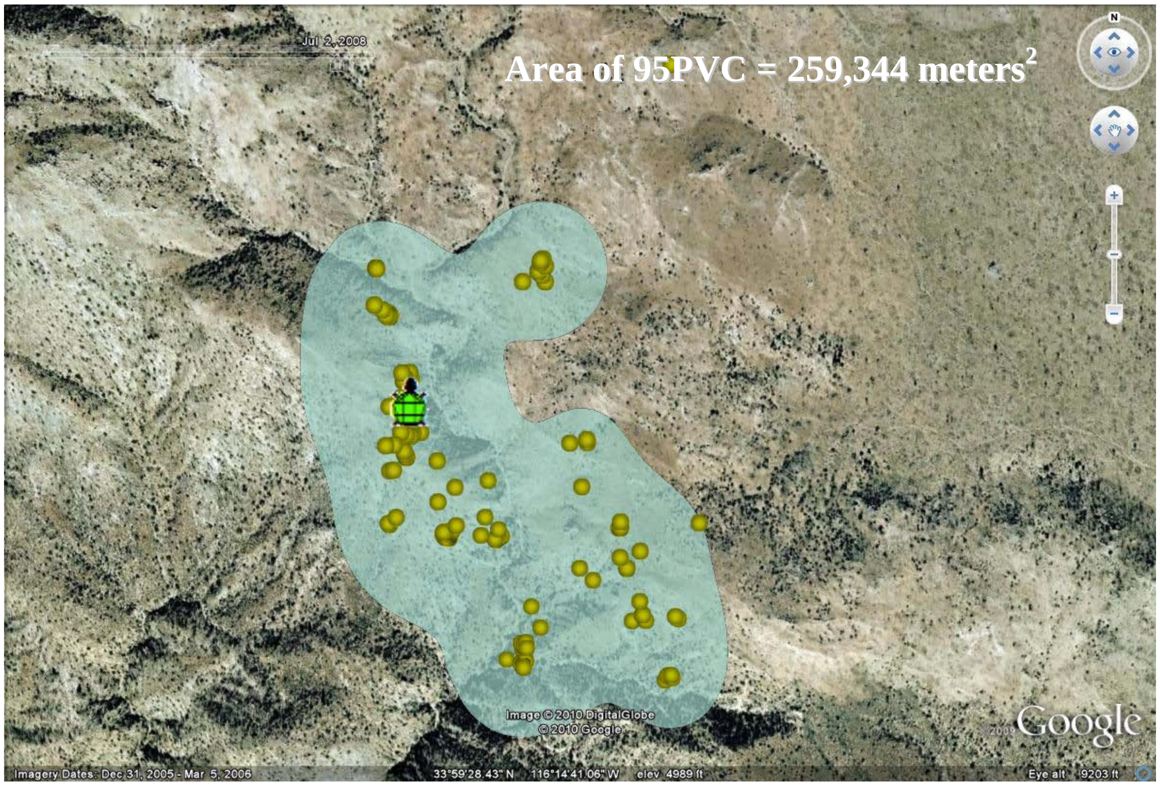

Jul 2, 2008
Area of 95PVC = 259,344 meters²
Image © 2010 DigitalGlobe
© 2010 Google
Google
Imagery Dates: Dec 31, 2005 - Mar 5, 2006
33°59'28.43" N 116°14'41.06" W elev 4989 ft
Eye alt 9203 ft

Figure 6-22: Google Earth View of a Dynamic 95 Percent Volume Contour

The single Kernel Density feature class was exported into 18 feature classes by tortoise, using the *Tortoise Import Tool*. These feature classes are not dynamic as they cover the entire study period. These new feature classes were then easily exported from ArcMap into KML format into Google Earth (Figure 6-23). The light blue area in the figure depicts the kernel density polygon for Elizabeth the tortoise in 2008.

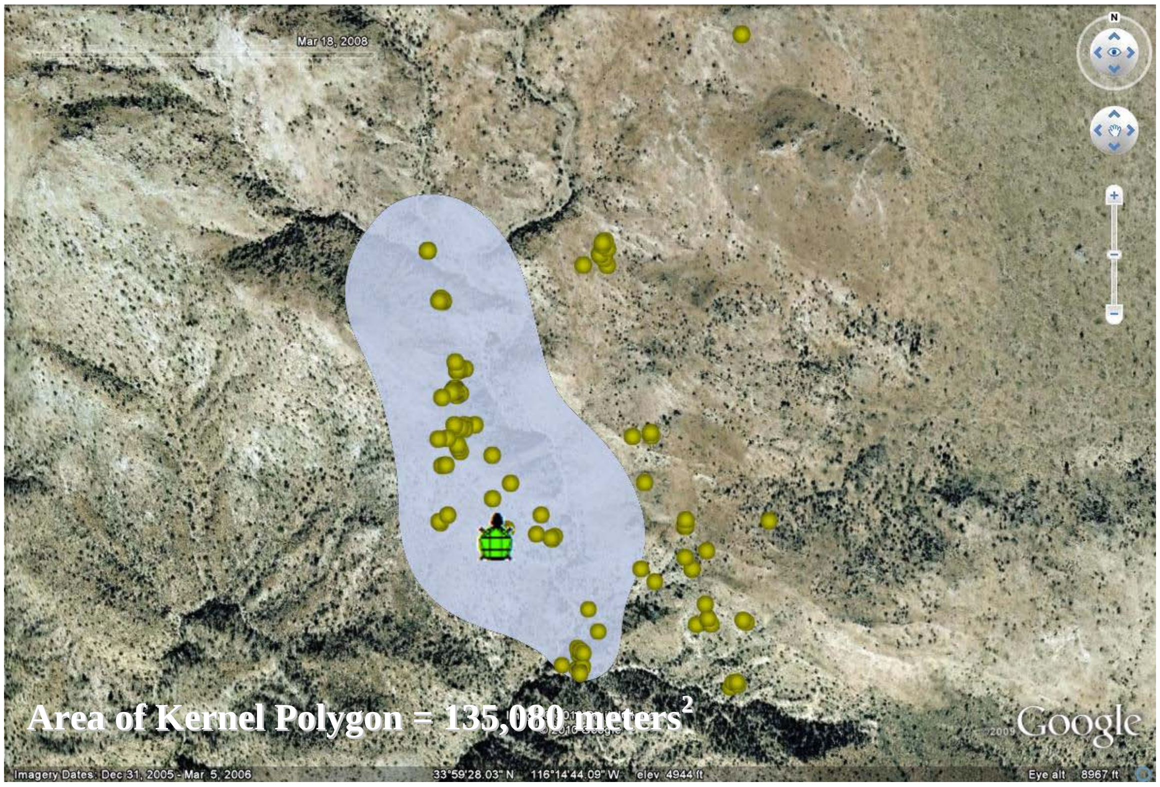

Mar 18, 2008
Area of Kernel Polygon = 135,080 meters²
Google
Imagery Dates: Dec 31, 2005 - Mar 5, 2006
33°59'28.03" N 116°14'44.09" W elev 4944 ft
Eye alt 8967 ft

Figure 6-23: Google Earth View of a Kernel Density Polygon

6.3 Issues

All of the processes for the use-cases were performed many times. This was required for documenting the processes and acted as a test plan. This involved start to finish processing of all the data. As a result, many issues were encountered.

Many coding glitches were found when different feature classes were processed. These errors were fixed. Formula modifications also were made, as errors were discovered in the results. Additionally, interface efficiencies discovered though later tools were added to the earlier built tools. The Python scripts for all of the tools are included in Appendix A. Additionally, the Toolbox Help html files are shown in Appendix B.

6.4 Process Times

When running all of the processes in the use-cases detailed in this chapter, the process times were recorded. The results suggest that all of the custom tools could be run on the tortoise data in about two hours. This signifies efficiencies in the tools. The other steps of the processes that are not tool-based add additional time that has not been measured; however, the goal of building efficient automated tools was achieved (Figure 6-24). Additionally, process output windows are included in Appendix D.

Tool Process Times (seconds)

Use Case:	Tool: Import Tool	TimeStamp Tool	Stationary Event	Lost Tortoise	Road & Trot	Speed Finder	Potential Path	Process Time: (minutes)
Roads & Trot Buffers	🟥	10	🟥	🟥	10	🟥	🟥	0.33
Tortoise Movement	49	157	🟥	🟥	🟥	🟥	🟥	3.43
Tortoise Speed - Name	🟥	*157	🟥	🟥	🟥	216	🟥	3.60
-By Name and Year	🟥	*157	🟥	🟥	🟥	226	🟥	3.77
Potential Path -Limited	*49	*157	🟥	🟥	🟥	*216	863	14.38
-Unlimited	*49	*157	🟥	🟥	🟥	*216	877	14.62
Lost Tortoise	🟥	🟥	🟥	18	🟥	*216	🟥	0.30
Weather -High Temps	🟥	230	4440	🟥	🟥	🟥	🟥	77.83
-Low Temps	🟥	210	🟥	🟥	🟥	🟥	🟥	3.50
-Precipitation	🟥	227	227	🟥	🟥	🟥	🟥	7.57
Tortoise Trot Needs	🟥	55	🟥	🟥	**	**	**	0.92
							Total Time:	130.25

***Previously Run Process**

****Uses Existing Feature Class**

Figure 6-24: Processing Times of Custom Tools with Use-Cases

6.5 Summary

This chapter addressed tool requirements and installation procedures, the processes for the use-cases involving the tools, results, issues encountered, and process times. The eight custom-built tools can be run either individually, or in conjunction with each other to support many use-cases. Such use-cases include road and tortoise trot buffering, tortoise movement visualization, tortoise speed calculations, tortoise potential path depiction, lost tortoise search zones, prioritization of curbing needs, temperature, and precipitation visualization. Additionally, the tools were used to modify Hawth's Tools produced MCP, 95PVC, and kernel density polygons. All of the created or modified feature classes that resulted from the use-cases were exported into KML format. This format allowed the features to be visualized in Google Earth or ArcGIS Explorer.

Chapter 7 – **Conclusions and Future Work**

The eight tools that were built for ArcGIS, along with the existing ArcGIS tools Export to KML and Military Analyst, were used for manipulation of the provided data to produce many new feature classes, tables, and KML files for the project. These tools were built to be living tools, meaning that they can be used on newly collected data for new datasets and analysis. In addition, the original project data could be used with the tools and different desired parameters to produce different results. The results in the project achieved the goals of providing a means to dynamically visualize tortoise movement and weather data. The client also wanted to analyze curbing and tortoise trots to see if they were adequate or if new curbing was needed. The Trot Needs ModelBuilder tied several of the other tools' results together to provide such a prioritization for new curbing.

Because of the extensive amount of work involved in creating the tools for this project, the lack of certain collected data, as well as ideas generated by completing this project, there are several areas for future work to expand on this project. Since the tools can be used for other species monitoring, as well as for other time-based dynamic visualizations, a generic version of the toolbox will be published for other analysts' use following the project. Since ArcGIS 10 and ArcGIS Explorer have time and animation integrated within their application, the tools offer great utility for the users of the new version.

Concerning studying different factors that may influence the desert tortoise, it could benefit JOTR to collect a useful vegetation layer depicting existence of annual vegetation types eaten by tortoises, attributed with the estimated start and end of the plant life cycle. The collected data could be animated using the existing tools and added to the dynamic model for study.

Two additional tools were considered during the project that could also enhance the tortoise toolbox. The tortoise database includes an attribute describing the status of the tortoise for each recorded position. Many of the records state that the tortoise is in their burrow. It is envisioned having a tool that parses out records that contain the word burrow, and making these features be the locations of known burrows. Additionally, the tool could analyze densities of the multiple positions of the recorded tortoises to statistically calculate potential burrow locations. For a second additional tool, it may be possible to have a tool that analyzes the created ellipses for all of the tortoises and find intersections in time to indicate possible tortoise interactions. This may potentially be used to identify mating habits, etc.

The tools created for this project would be useful for the study of other animals, as well. It is recommended to share the tools and deliverables throughout the NPS. It was realized that the animations for tortoises are lacking due to the nature of the recorded positions. There are many gaps in time with the tortoises because rangers have to locate and record positions. Weather also creates gaps in recorded data. The utilization of GPS on the tracked animals would enhance the animations immensely and minimize the predictions involved in studying the tortoises.

With this project, the custom-built toolbox, and the general version of the toolbox to be built, dynamic visualization of tortoises, other animals, and their environment are possible. It is envisioned that the tools will continue to be used and developed to further help with the protection of species, and used for other time-based needs in ArcGIS.

Works Cited

Beyer, H. (n.d.). Hawth's Analysis Tools for ArcGIS [ArcMap Extension Computer Program]. Retrieved April 20, 2010, from Spatial Ecology.com: http://www.spatialecology.com/htools/overview.php

Bissonette, J. A., Sherburne, S. S., & Ramsey, R. D. (1994). Analysing telemetry data with a GIS-based vector structure. *International Journal of Geographical Information Science , 8* (6), 533-543.

Boarman, W. I., Sazaki, M., & Jennings, W. B. (1997). The Effect of Roads, Barrier Fences, and Culverts on Desert Tortoise Populations in California, USA. *Conservation, Restoration, and Management of Tortoises and Turtles - An International Conference* (pp. 54-58). New York Turtle and Tortoise Society.

Cole, W. P. (1977). *Using the UTM Grid System to Record Historic Sites.* Retrieved July 10, 2010, from National Park Service, National Register Publications: http://www.nps.gov/history/nr/publications/bulletins/nrb28/

Davidson, C. (2009). *Joshua Tree National Park - Desert Tortoise*. Retrieved September 30, 2009, from National Park Service: http://www.nps.gov/jotr/naturescience/tortoise.htm

Duda, J. J., & Krzysik, A. J. (1998). *Radiotelemetry Study of a Desert Tortoise Population.* US Army Corps of Engineers.

Joshua Tree National Park. (2008). *2008 Annual Report for the Monitoring Program to Assess the Effects of Curbing on Tortoise Movement and Survival.* Resource Management Division, Wildlife Branch.

Kernohan, B. J., Millspaugh, J. J., Jenks, J. A., & Naugle, D. E. (1998). Use of an adaptive kernel home-range estimator in a GIS environment to calculate habitat use. *Journal of Environmental Management* (53), 83-86.

Martin, K. (2010). Export to KML (2.5.5) [ArcMap Extension Computer Program].

Peterson, C. C. (1996, Sep.). Ecological Energetics of the Desert Tortoise (Gopherus Agassizii): Effects of Rainfall and Drought. *Ecology , 77* (6), pp. 1831-1844.

Riedle, J. D., Bolen, D. K., & Averill-Murray, R. C. (2002). *Desert Tortoise Habitat Use and Home Range Size on the Florence Military Reservation 2002 Progress Report.* Arizona Game and Fish Department, Nongame Branch, Wildlife Management Division.

Appendix A. Python Code

Python Code 1: Tortoise Import Tool

```
# ---------------------------------------------------------------------------
# tortimport.py
# Created on: Feb 27 2010 12:44:03 PM
#
# Built By David Turnbull (NGA), University of Redlands MS GIS for
# Joshua Tree National Park as part of his Master's Project
# Updated: 5/18/2010
# ---------------------------------------------------------------------------

# Import system modules
import sys, string, os, arcgisscripting
from datetime import date, datetime, timedelta, time

# Create the Geoprocessor object
gp = arcgisscripting.create()
gp.overwriteoutput =1

# Load required toolboxes...
gp.AddToolbox("C:/Program Files
(x86)/ArcGIS/ArcToolbox/Toolboxes/Conversion Tools.tbx")

# Local variables...
gp.addmessage("Setting Local Variables...")
infeatureClass = gp.GetParameterAsText(0)
workspacex = gp.GetParameterAsText(1)
methName = gp.GetParameterAsText(2)
tortfield = gp.GetParameterAsText(3)
datefield = gp.GetParameterAsText(4)
tag = gp.GetParameterAsText(5)

#adds tag + underscore to filenames in end if tag is populated
if tag != "":
    tag = tag+"_"

gp.AddMessage("")
gp.Workspace = workspacex

#Selectable Methods...
FCBT = "Create Feature Classes By Tortoise"
FCBTY = "Create Feature Classes By Toroise and Year"
FCABY = "Create All Tortoises By Year"
FCBTTY = "Create All Three Types of Feature Classes Above"

#This Creates the Tortoise List...
gp.AddMessage("Tortoises:")
startSearch = gp.UpdateCursor(infeatureClass)###changed to Update
tortlist = list()
searchRow = startSearch.Next()

while searchRow:
    tort = searchRow.GetValue(tortfield)
    ###Added this section to remove spaces from Names
    tort = tort.replace(" ","")
    searchRow.SetValue(tortfield, tort)
    startSearch.UpdateRow(searchRow)

    if tort not in tortlist:
        tortlist.append(tort)
        gp.AddMessage(tort) #prints tortoise name
    searchRow = startSearch.Next()
del startSearch

#Feature Classes By tortoises...
if methName == FCBT or methName == FCBTTY:
    for tort in tortlist:
        gp.FeatureClassToFeatureClass_conversion(infeatureClass,
workspacex, tag+tort, '"'+tortfield+'"'+" = "+"'"+tort+"'")

elif (methName == FCBTY and datefield != "") or (methName == FCBTTY and
datefield != ""):
    gp.AddField(infeatureClass,"YearX","Text")
    for tort in tortlist:
        yrSearch = gp.UpdateCursor(infeatureClass, '"'+tortfield+'"'+"
                                  = "+"'"+tort+"'")
        yrlist = list()
        yr = yrSearch.Next()

        while yr:
            thisdate = datetime.strptime(yr.getValue(datefield),"%Y-%m-
                                        %d %H:%M:%S")
            thisyr = thisdate.year
            if thisyr not in yrlist:
                yrlist.append(thisyr)
                gp.AddMessage(thisyr) #prints tortoise year
            yr.SetValue("YearX", thisyr)
            yrSearch.UpdateRow(yr)
            yr = yrSearch.Next()

        for y in yrlist:
            tyear = str(y)
            gp.FeatureClassToFeatureClass_conversion(infeatureClass,
                        workspacex, tag+tort+"_"+tyear,
                        '"'+tortfield+'"'+" = "+"'"+tort+"'"+" and
                        "+'"YearX" = '+"'"+tyear+"'")
    del yrSearch

elif (methName == FCABY and datefield != "") or (methName == FCBTTY and
                                                     datefield != ""):
    gp.AddField(infeatureClass,"YearX","Text")
    yrlist2 = list()
    yrSearch2 = gp.UpdateCursor(infeatureClass)
    yr2 = yrSearch2.Next()

    while yr2:
        thisdate2 = datetime.strptime(yr2.getValue(datefield),"%Y-%m-%d
                                     %H:%M:%S")
        thisyr2 = thisdate2.year
        if thisyr2 not in yrlist2:
            yrlist2.append(thisyr2)
        yr2.SetValue("YearX", thisyr2)
        yrSearch2.UpdateRow(yr2)
        yr2 = yrSearch2.Next()

    for y2 in yrlist2:
        tyear2 = str(y2)
        gp.FeatureClassToFeatureClass_conversion(infeatureClass,
workspacex, tag+"AllTortoises_"+tyear2, '"YearX" = '+"'"+tyear2+"'")
    del yrSearch2

else:
    gp.AddMessage("Try Again")

gp.AddMessage("------------------------")
```

Python Code 2: TimeStamp and EndTime Field Creator Tool

```
# ---------------------------------------------------------------------
# timestamp.py
# Created on: Wed Mar 18, 2010
#
# Built By David Turnbull (NGA), University of Redlands MS GIS for
# Joshua Tree National Park as part of his Master's Project
# 4/22/2010
# ---------------------------------------------------------------------

# Import system modules
import sys, string, os, arcgisscripting, time, datetime, math
from datetime import date, datetime, timedelta
from operator import itemgetter


# Create the Geoprocessor object
gp = arcgisscripting.create()
gp.overwriteoutput =1

# Load required toolboxes...
gp.AddToolbox("C:/Program Files (x86)/ArcGIS/ArcToolbox/Toolboxes/Data
             Management Tools.tbx")

# Local variables...
gp.addmessage("Setting Local Variables...")
infeatureClass = gp.GetParameterAsText(0)
insort = gp.GetParameterAsText(1)
infield = gp.GetParameterAsText(2)
intime = gp.GetParameterAsText(3)
endMethod = gp.GetParameterAsText(4)
endDays = gp.GetParameterAsText(5)

#This sets variable to selection for determining endTime
nFeature = "EndTime is StartTime of Next Occuring Feature"
nDay = "EndTime is Start of Next Day(s)"
noEnd = "No EndTime Calculation"

# Adding the TimeStamp or TimeStamp_ field and EndTime field
gp.addmessage("Adding the TimeStamp Field...")
gp.AddField(infeatureClass,"TimeStamp","Text") ###was Date
if endMethod != noEnd:
    gp.addmessage("Adding the EndTime field")
    gp.AddField(infeatureClass,"EndTime","Text") ###was Date

gp.workspace= infeatureClass

#Creating a sort list. If populated, a multilist, if empty, a single
list
#This is necessary for the eventtime starting at next.
if insort != "":
    startSearch = gp.SearchCursor(infeatureClass)
    gp.AddMessage("Tortoises:")
    tortlist = list()
    searchRow = startSearch.Next()
    while searchRow:
        tort = searchRow.GetValue(insort)
        if tort not in tortlist:
            tortlist.append(tort)
            gp.AddMessage(tort) #prints tortoise name
        searchRow = startSearch.Next()
    del startSearch
elif insort == "": #I did it this way so there will be a list either
way.
    tortlist = list()
    tortlist.append("SingleSort")
    gp.AddMessage("Sort not specified, treating all records as part of
one event class.")

#Adding the TimeStamp field
if infield != "":
    gp.AddMessage("")
    gp.AddMessage("Feature Class: "+infeatureClass)
    gp.AddMessage("Converting "+infield+" to TimeStamp Field...")
    gp.AddMessage("")

    desc= gp.Describe(infeatureClass)
    fields = desc.Fields
    field = fields.next()
    while field:
        if field.Name == "TimeStamp":
            fieldn ="TimeStamp"
            break
        elif field.Name == "TimeStamp_":
            fieldn = "TimeStamp_"
            break
        field = fields.next()

    #Popoulate TimeStamp
    for tort in tortlist:
        if insort == "":
            TimeRows= gp.UpdateCursor(infeatureClass)
        elif insort != "":
            TimeRows= gp.UpdateCursor(infeatureClass, '"'+insort+'"'+"
                                      = "+"'"+tort+"'")

                #Creating an empty list
        tlist = tort+"_timelst"
        tlist = list()

        TR = TimeRows.Next()
        while TR:
            dayx = datetime.strptime(TR.GetValue(infield), "%m/%d/%Y")
            xdayx = dayx
            if intime != "":
                timex = TR.GetValue(intime)
                desc2 = gp.Describe(infeatureClass)
                fields2 = desc.Fields
                fields = fields2.next()
                while fields:
                    if fields.Name == intime:
```

```
                ft = fields.Type
                gp.AddMessage("")
            fields = fields2.next()
        if ft == "Double":
            strtime = str(timex)
            ntime = datetime.strptime(strtime, "%H.%M")
            newtime = datetime.time(ntime)
            daytimex = str(datetime.combine(xdayx, newtime))
        elif ft == "Date":
            otime = datetime.strptime(timex, "%I:%M:%S %p")
            timex = datetime.time(otime)
            daytimex = str(datetime.combine(xdayx,timex))
    elif intime == "":
        daytimex = str(dayx)

    TR.SetValue(fieldn,daytimex)
    TimeRows.UpdateRow(TR)
    newtime = TR.GetValue(fieldn)
    gp.AddMessage(newtime)

    if newtime not in tlist:
        tlist.append(newtime)
    TR = TimeRows.Next()

if endMethod == nFeature:
    resultx = sorted(tlist, reverse=False)
    n = len(resultx)
    gp.AddMessage(n)

    gp.AddMessage("Sorting List Next...")
    #Check to see order of new list--
    gp.AddMessage("")
    gp.AddMessage("This is the sorted list")
    i=0
    for i in range(n):
        gp.AddMessage(resultx[i])
    gp.AddMessage("End of sorted list")
    gp.AddMessage("")

    gp.AddMessage("Populating EndTime with the date of the
                 start time of the next event.")

    if insort == "":
        FinishTime= gp.UpdateCursor(infeatureClass)
    elif insort != "":
        FinishTime= gp.UpdateCursor(infeatureClass,
                            '"'+insort+'"'+" = "+"'"+tort+"'")

    L1 = FinishTime.Next()
    while L1:
        start = L1.GetValue(fieldn)
        gp.AddMessage(start)

        i=0

        if start in resultx:
            i = resultx.index(start) + 1
```

```
                if i < n:
                    end = resultx[i]
                    L1.SetValue("EndTime",end)
                    FinishTime.UpdateRow(L1)
            L1 = FinishTime.Next()

    if endMethod == nDay: #This only shows a feature for one day
        if endDays == 1:
            gp.AddMessage("Populating EndTime with a date "+endDays+"
                          Day later.")
        else:
            gp.AddMessage("Populating EndTime with a date "+endDays+"
                          Days later.")

        FinishTime= gp.UpdateCursor(infeatureClass)

        L1 = FinishTime.Next()
        while L1:
            if intime == "":
                start = L1.GetValue(fieldn)
                end1 = datetime.strptime(start,"%Y-%m-%d %H:%M:%S")
                numDays = float(endDays) #Can do decimal days even
                                         #though no initial time
                d = timedelta(days=numDays)
                end = end1 + d
                d2 = datetime.strftime(end, "%Y-%m-%d %H:%M:%S")

                L1.SetValue("EndTime", d2)
                FinishTime.UpdateRow(L1)
            elif intime != "":
                start = L1.GetValue(fieldn)
                end1 = datetime.strptime(start,"%Y-%m-%d %H:%M:%S")
                numDays =  float(endDays)
                d = timedelta(days=numDays)
                end = end1 + d
                d2 = datetime.strftime(end, "%Y-%m-%d %H:%M:%S")

                L1.SetValue("EndTime", d2)
                FinishTime.UpdateRow(L1)
            L1 = FinishTime.Next()

gp.AddMessage("____________")
```

Python Code 3: Stationary Event from Table tool

```
# ---------------------------------------------------------------------------
# poly2tableNew2.py
# Created on: Fri Jan 08 2010
#
# Built By David Turnbull (NGA), University of Redlands MS GIS for
# Joshua Tree National Park as part of his Master's Project
# Updated: 5/18/2010
# ---------------------------------------------------------------------------

#Import system modules
import sys, string, os, arcgisscripting, copy
from operator import itemgetter
from datetime import date, datetime

#Create the Geoprocessor object
gp = arcgisscripting.create(9.3)
gp.overwriteoutput =1

#Local variables...
gp.addmessage("Setting Local Variables...")
mainFeatureClass = gp.GetParameterAsText(0)
inTable = gp.GetParameterAsText(1)
inRow = gp.GetParameterAsText(2)
procType = gp.GetParameterAsText(3)
dateField = gp.GetParameterAsText(4)
tempField = gp.GetParameterAsText(5)
convert = gp.GetParameterAsText(6)
workspace = gp.GetParameterAsText(7)
tag = gp.GetParameterAsText(8)

if tag != "":
    tag = tag+"_"
else:
    tag = tag

gp.Workspace = workspace
inFeatureClass = "inFeatureClass"

#These are the choices for procType: Coding them for ease
THL = "Temperature-Daily High and Low"
TAV = "Temperature-Daily Average"
THO = "Temperature-Hourly- matches table records (Large dataset \
      possible)"
PRE = "Precipitation-Daily Total"
OTH = "Other- matches table records"

#Temporary feature class that will be deleted in end
gp.AddMessage("")
gp.AddMessage("Creating Temporary Dataset...")
try:
    gp.FeatureClassToFeatureClass_conversion(mainFeatureClass,
                              workspace, inFeatureClass, "", "")
except:
    print "An error occurred"
    print gp.GetMessages()

#Give Feature Counts - For All Cases!
gp.AddMessage("")
gp.AddMessage("Input Feature Class: "+mainFeatureClass)
gp.AddMessage("Table: "+inTable)
gp.AddMessage("Object id: "+inRow)
gp.AddMessage("")

#Check to see if fields already were created
processed = "N"
drsc = gp.describe(inTable)
fn = drsc.Fields
for fns in fn:
    if fns.Name == "TCrossRef":
        processed = "Y"
        gp.AddMessage("Table already processed...")

if processed == "Y":
    gp.deletefield(inTable, "TCrossRef")

del drsc

#THL or TAV processes...
if procType == THL or procType == TAV:

    if processed == "N":
        #adding fields
        gp.addmessage("Adding the HighTemp and Low Temp Field...")
        gp.AddField(inTable,"HighTemp","Text")
        gp.AddField(inTable,"LowTemp","Text")
        gp.AddField(inTable,"TempAve","Double")

        dateList = list()
        mmaxRows =gp.SearchCursor(inTable)
        mmaxRow = mmaxRows.Next()
        while mmaxRow:
            day = mmaxRow.GetValue(dateField)
            tx = mmaxRow.GetValue(tempField)
            if day not in dateList and (tx != -999):
                dateList.append(day)
            mmaxRow = mmaxRows.Next()
        del mmaxRow
        del mmaxRows

        gp.addmessage("Setting Max and Min Temperature...")
        for item in dateList:
            tmpList = list()
            searchRows = gp.SearchCursor(inTable, '"'+dateField+'"'" =
                                         date '"+item+"'")
            searchRow= searchRows.Next()
            total = 0
            tcount = 0
            while searchRow:
                this = searchRow.GetValue(tempField)
                if (this != -999):
                    tmpList.append(this)
                    total = total + this
                    tcount = tcount + 1
                searchRow = searchRows.Next()
            mintemp = min(tmpList) #sets min, max and ave temps
            maxtemp = max(tmpList)
            aveTemp = total/tcount
            del tmpList

            updateRows = gp.UpdateCursor(inTable, '"'+dateField+'"'" =
                                        date '"+item+"'")
            updateRow= updateRows.Next()
            updatedHi = 0
            updatedLo = 0
            while updateRow:

                tempCheck = updateRow.GetValue(tempField)
                if (tempCheck == mintemp and updatedLo == 0):
                    updateRow.SetValue("LowTemp","True")
                    updateRow.SetValue("TempAve",aveTemp)
                    updateRows.UpdateRow(updateRow)
                    updatedLo = updatedLo + 1
                if (tempCheck == maxtemp and updatedHi == 0):
                    updateRow.SetValue("HighTemp","True")
                    updateRow.SetValue("TempAve", aveTemp)
                    updateRows.UpdateRow(updateRow)
                    updatedHi = updatedHi +1
                updateRow= updateRows.Next()

        del searchRow # Clean up
        del searchRows
        del updateRow # clean up
        del updateRows
        del dateList

##Precipitation
elif procType == PRE:
    if processed == "N":
        #adding fields
        gp.addmessage("Adding the Daily Precipitation Field...")
        gp.AddField(inTable,"DailyPreMM","Double")
        gp.AddField(inTable,"Use","Text")

        dateList = list()
        mmaxRows =gp.SearchCursor(inTable, '"'+tempField+'"'+" > 0")
        mmaxRow = mmaxRows.Next()
        while mmaxRow:
            day = mmaxRow.GetValue(dateField)
            tx = mmaxRow.GetValue(tempField)
            if day not in dateList:
                dateList.append(day)
            mmaxRow = mmaxRows.Next()
        del mmaxRow # Clean up
        del mmaxRows

        gp.addmessage("Calculating Daily Cumulative Precipitation...")
        for item in dateList:
            tmpList = list()
            searchRows = gp.UpdateCursor(inTable, '"'+dateField+'"'" =
                        date '"+item+"' and "+'"'+tempField+'"'+" > 0")
            searchRow = searchRows.Next()
            total = 0
            tcount = 0
            while searchRow:
                this = searchRow.GetValue(tempField)
                if tcount == 0:
                    searchRow.SetValue("Use","Y")
                    # This puts a Y in the Updated field for first
                    # record of query
                    searchRows.UpdateRow(searchRow)
                if (this != -999):
                    tmpList.append(this)
                    total = total + this
                    tcount = tcount + 1
                searchRow = searchRows.Next()
            totPre = total
            del tmpList
            del searchRow # Clean up
            del searchRows

            updateRows = gp.UpdateCursor(inTable, '"'+dateField+'"'" =
                              date '"+item+"' and "+'"Use" = '+"'Y'")
            updateRow= updateRows.Next()

            while updateRow:
                updateRow.SetValue("DailyPreMM",totPre)
                updateRows.UpdateRow(updateRow)
                updateRow= updateRows.Next()

        del updateRow # Clean up
        del updateRows
        del dateList

#This returns the number of records in the table to be used later
if procType == THL or procType == TAV:
    thlCount = 0
    thr =gp.SearchCursor(inTable, '"HighTemp" = '+"'True' or
                        "+'"LowTemp" = '+"'True'")
    thlrow = thr.Next()
    while thlrow:
        thlCount = thlCount + 1
        thlrow = thr.Next()
    resultx = thlCount
    del thlrow # Clean up
    del thr
elif procType == PRE:
    thlCount =0
    thr =gp.SearchCursor(inTable, '"DailyPreMM" >= 0')
    thlrow = thr.Next()
    while thlrow:
        thlCount = thlCount + 1
        thlrow = thr.Next()
    resultx = thlCount
    del thlrow # Clean up
    del thr
```

```
else:
    resultx = gp.GetCount_management(inTable)
stringresult = str(resultx)
gp.AddMessage("The table "+inTable+" has "+ stringresult +" records;")
gp.AddMessage("they will be put into the feature class:
"+inFeatureClass+".")
gp.AddMessage("")

#This adds a CrossRef index field to the two datasets for later field
#join
gp.AddMessage("Adding the CrossRef fields...")
        #In Feature Class Check and add
fieldExists = "False"
flds = gp.listfields(inFeatureClass)
for field1 in flds:
    gp.AddMessage(field1)
    if field1.Name == "CrossRef":
        fieldExists = "True"
        gp.AddMessage("CrossRef Already Exists.")
if fieldExists == "False":
    gp.AddField(inFeatureClass,"CrossRef","double")
    gp.AddMessage("CrossRef was added.")
        #In Table Class Check and add

fieldExists2 = 0
flds2 = gp.listfields(inTable)
for field2 in flds2:
    if field2.Name == "TCrossRef":
        fieldExists2 = 1
        gp.AddMessage("TCrossRef Already Exists.")
if fieldExists2 == 0:
    gp.AddField(inTable,"TCrossRef","double")
    gp.AddMessage("TCrossRef was added.")

#This populates the CrossRef of the inTable
if procType == THL or procType == TAV:    #### THL and TAV
    tabrows =gp.UpdateCursor(inTable, '"HighTemp" = '+"'True' or
                                          "+'"LowTemp" = '+"'True'")
    tabrow = tabrows.Next()
    tcross = 1
    while tabrow:
        tabrow.SetValue("TCrossRef", tcross)
        tabrows.UpdateRow(tabrow)
        tcross = tcross+1
        tabrow = tabrows.Next()
    del tabrow # Clean up
    del tabrows
elif procType == PRE: #Precipitation
    tabrows =gp.UpdateCursor(inTable, '"DailyPreMM" >= 0')
    tabrow = tabrows.Next()
    tcross = 1
    while tabrow:
        tabrow.SetValue("TCrossRef", tcross)
        tabrows.UpdateRow(tabrow)
        tcross = tcross+1
        tabrow = tabrows.Next()
    del tabrow # Clean up
    del tabrows
else:               ##### Everything Else!!!!
    xtabrows =gp.UpdateCursor(inTable)
    xtabrow = xtabrows.Next()
    tcross = 1
    while xtabrow:
        xtabrow.SetValue("TCrossRef", tcross)
        xtabrows.UpdateRow(xtabrow)
        tcross = tcross+1
        xtabrow = xtabrows.Next()
    del xtabrow # Clean up
    del xtabrows

gp.AddMessage("")
#Get object in in Feature class to copy
rows =gp.UpdateCursor(inFeatureClass)
row = rows.Next()
while row:
    test = str(row.GetValue("ObjectID"))
    if test == inRow:
        gp.AddMessage("OBJECT FOUND!")
        #Updates the Cross Ref for just this one object
        row.SetValue("CrossRef", 1)
        rows.UpdateRow(row)
        #End of Cross Ref updating
        geom = row.shape #existing row
        desc = gp.Describe(inFeatureClass)
        fields = desc.Fields #existing fields
        x = 0
        myvars ={}
        myvars2 ={}
        for field in fields:
            if (field.Name != "OBJECTID" and field.Name != "Shape" and
                  field.Name != "Date_" and field.Name != "Time" and
                  field.Name !="Id" and field.Name !="Area" and
                  field.Name != "TimeStamp" and field.Name != "EndTime"
                  and field.Name !="Shape_Length" and field.Name !=
                  "Shape_Area" and field.Name != "CrossRef" and
                  field.Name != "TCrossRef"):
                x = x +1
                num = str(x)
                myvars["attr_"+num] = field.Name
                myvars2["val_"+num] = row.GetValue(field.Name)
    row = rows.Next()
del row # Clean up
del rows

gp.AddMessage("Creating Duplicate Features, Please wait...")
gp.AddMessage("")
cprows= gp.InsertCursor(inFeatureClass)
count = 1
i = 1
cross = 2 #cross reference 1 already exists
while count <= (resultx -1): #1 row in class already exists
    cprow = cprows.NewRow()
    while i <= x:
        newi = str(i)
        new_attr = myvars["attr_"+newi]
        new_value = myvars2["val_"+newi]
        cprow.SetValue(new_attr, new_value)
        i = i +1
    cprow.SetValue("CrossRef", cross)
    cprow.shape = geom
    cprows.InsertRow(cprow)
    cross = cross+1

    count = count +1
del cprow # Clean up
del cprows
gp.AddMessage("Attaching the table attributes to features...")
gp.joinfield(inFeatureClass, "CrossRef", inTable, "TCrossRef")

##Adding Optional Fahrenheit Fields
#True = "true"
#False = "false"

if convert == "true":
    gp.AddMessage("Converting units from metric to new English
Field...")
if convert == "true" and (procType == THL or procType == TAV):
    gp.AddField(inFeatureClass,"LowTFahr","Double")
    gp.AddField(inFeatureClass,"HighTFahr","Double")
    gp.AddField(inFeatureClass,"TempAveFahr","Double")

    unitSearch = gp.UpdateCursor(inFeatureClass)
    unit = unitSearch.Next()
    while unit:
        getlo = "False"
        gethi = "False"
        getlo = unit.GetValue("LowTemp")
        if getlo == "True":
            cLow = float(unit.GetValue(tempField))
            fLow = int(((cLow*9.0)/5)+32)
            unit.SetValue("LowTFahr", fLow)
            unitSearch.UpdateRow(unit)
        else:
            gethi = unit.GetValue("HighTemp")
            if gethi == "True":
                cHigh = float(unit.GetValue(tempField))
                fHigh = int(((cHigh*9.0)/5)+32)
                unit.SetValue("HighTFahr", fHigh)
                unitSearch.UpdateRow(unit)
        tCheck = unit.GetValue("TCrossRef")
        if tCheck > 0:
            cAve = float(unit.GetValue("TempAve"))
            fAve = int(((cAve*9.0)/5) + 32.0)
            unit.SetValue("TempAveFahr", fAve)
            unitSearch.UpdateRow(unit)
        unit = unitSearch.Next()
    del unit
    del unitSearch

elif convert == "true" and procType == PRE:
    gp.AddField(inFeatureClass,"DailyPreInch","Double")
    unitSearch2 = gp.UpdateCursor(inFeatureClass)
    unit2 = unitSearch2.Next()
    while unit2:
        t2Check = unit2.GetValue("TCrossRef")
        if t2Check > 0:
            mmPre = unit2.GetValue("DailyPreMM")
            gp.AddMessage(str(mmPre))
            inPre = round((mmPre/25.4), 1)
            unit2.SetValue("DailyPreInch", inPre)
            unitSearch2.UpdateRow(unit2)
        unit2 = unitSearch2.Next()
    del unit2
    del unitSearch2


#Creating new feature classes and deleting the temporary ones
if procType == THL:
        gp.AddMessage("Creating High and Low Temperature Feature
                      Classes...")

        # Process: HighTemperatures...
        highnewset = workspace+"\\"+tag+"HighTemps"
        gp.Select_analysis(inFeatureClass, highnewset, "\"HighTemp\" =
                                                              'True'")

        # Process: Low Temperatures...
        lownewset = workspace+"\\"+tag+"LowTemps"
        gp.Select_analysis(inFeatureClass, lownewset, "\"LowTemp\" =
                                                              'True'")

        #Deleting the temporary dataset
        gp.AddMessage("Deleting the temporary dataset...")
        gp.Delete_management(inFeatureClass)

elif procType == TAV:
        gp.AddMessage("Creating Average Temperature Feature
                                                     Classes...")
        # Process: AverageTemperatures...
        tavnewset = workspace+"\\"+tag+"AveTemp"
        gp.Select_analysis(inFeatureClass, tavnewset, "\"TempAve\" >= -
                                                                    30")

        #Deleting the temporary dataset
        gp.AddMessage("Deleting the temporary dataset...")
        gp.Delete_management(inFeatureClass)

elif procType == PRE:
        gp.AddMessage("Creating Daily Precipitation Features...")
        # Process: Daily Precipitation...
        newset = workspace+"\\"+tag+"DailyPre"
        gp.Select_analysis(inFeatureClass, newset, "\"Use\" >= 'Y'")

        #Deleting the temporary dataset
        gp.AddMessage("Deleting the temporary dataset...")
        gp.Delete_management(inFeatureClass)

elif procType == THO:
        gp.AddMessage("Creating Hourly Features...")
        # Process: Hourly...
        newset = workspace+"\\"+tag+"Hourly"
        gp.Select_analysis(inFeatureClass, newset)

        #Deleting the temporary dataset
        gp.AddMessage("Deleting the temporary dataset...")
        gp.Delete_management(inFeatureClass)

elif procType == OTH:
        gp.AddMessage("Creating New Time Features...")
        # Process: Other...
        newset = workspace+"\\"+tag+"TimeFeatures"
        gp.Select_analysis(inFeatureClass, newset)

        #Deleting the temporary dataset
        gp.AddMessage("Deleting the temporary dataset...")
        gp.Delete_management(inFeatureClass)
gp.AddMessage("Done!")
gp.AddMessage("____________")
```

Python Code 4: Road and Trot Buffer Tool

```
# ---------------------------------------------------------------------
# buffertool.py
# Created on: Sat Dec 12 2009 12:27:20 PM
#
# Built By David Turnbull (NGA), University of Redlands MS GIS for
# Joshua Tree National Park as part of his Master's Project
# 05/17/2010
# --------------------------------------------------------------------

# Import system modules
import sys, string, os, arcgisscripting

# Create the Geoprocessor object
gp = arcgisscripting.create()

gp.overwriteoutput = 1

# Load required toolboxes...
gp.AddToolbox("C:/Program Files
(x86)/ArcGIS/ArcToolbox/Toolboxes/Analysis Tools.tbx")

# Local variables...
gp.addmessage("Setting Local Variables...")
Public_Road = gp.GetParameterAsText(0)
First =gp.GetParameterAsText(1)
Buffer1 = gp.GetParameterAsText(2)
Second = gp.GetParameterAsText(3)
Buffer2 = gp.GetParameterAsText(4)
Trot = gp.GetParameterAsText(5)
TrotLayer = gp.GetParameterAsText(6)
TrotSize = gp.GetParameterAsText(7)
workspace = gp.GetParameterAsText(8)
tag = gp.GetParameterAsText(9)

#This had to be added due to a bug in 9.3 whereby the booleans are
lower case
True = "true"
False = "false"

gp.Workspace = workspace

#Tag handling
if tag == "":
    tag = ""
else:
    tag = tag+"_"

# Process: Buffer (2)...
if (Second == True and Public_Road != "" and Buffer2 != ""):
    gp.addmessage("")
    gp.addmessage("Creating Far Buffer of "+ Buffer2+ " ...")
    gp.addmessage("Source Feature Class: "+ Public_Road)
    roadfar = tag+"Roads_Buffer_Far"
    gp.addmessage("Output Feature Class: "+ workspace+": "+roadfar)
    gp.Buffer_analysis(Public_Road, roadfar, Buffer2, "FULL", "ROUND",
                                                        "ALL", "")
    gp.addmessage("You may have to change layer order to view both!")

# Process: Buffer (1)...
if (First == True and Public_Road != "" and Buffer1 != ""):
    gp.addmessage("")
    gp.addmessage("Creating Close Proximatey Buffer of "+ Buffer1+ "
                                                              ...")
    gp.addmessage("Source Feature Class: "+ Public_Road)
    roadnear = tag+"Roads_Buffer_Near"
    gp.addmessage("Output Feature Class: "+ workspace+": "+roadnear)
    gp.Buffer_analysis(Public_Road, roadnear, Buffer1, "FULL", "ROUND",
                                                        "ALL", "")

# Process: Trot Buffer...
if (Trot == True and TrotLayer != "" and TrotSize != ""):
    gp.addmessage("")
    gp.addmessage("Creating Trot Buffer of "+ TrotSize+ " ...")
    gp.addmessage("Source Feature Class: "+ TrotLayer)
    trotbuff = tag+"TortoiseTrot_Buffer"
    gp.addmessage("Output Feature Class: "+ workspace+": "+trotbuff)
    timecheck = ("TimeStamp" or "TimeStamp_")
    gp.Buffer_analysis(TrotLayer, trotbuff, TrotSize, "FULL", "ROUND",
                                            "LIST", timecheck)

else:
    gp.AddMessage("The selected buffers were created")

gp.AddMessage("____________")
```

Python Code 5: Tortoise Maximum Speed Finder Tool

```
# ---------------------------------------------------------------------------
# tortspeed2.py
# Created on: Thurs Feb 11, 2010
#
# Built By David Turnbull (NGA), University of Redlands MS GIS for
# Joshua Tree National Park as part of his Master's Project
# 5/17/2010
# ---------------------------------------------------------------------------

# Import system modules
import sys, string, os, arcgisscripting, time, datetime, math
from datetime import date, datetime, timedelta, time
from operator import itemgetter

# Create the Geoprocessor object
gp = arcgisscripting.create()
gp.overwriteoutput =1

# Load required toolboxes...
gp.AddToolbox("C:/Program Files (x86)/ArcGIS/ArcToolbox/Toolboxes/Data
                                                Management Tools.tbx")

# Local variables...
gp.AddMessage("Setting Local Variables...")
infeatureClass = gp.GetParameterAsText(0)
infield = gp.GetParameterAsText(1)
inMethod = gp.GetParameterAsText(2)
inDate = gp.GetParameterAsText(3)
infolder = gp.GetParameterAsText(4)

gp.Workspace = infolder

#This sets variable to selection for determining method
methName = "Tortoise By Name"
methYear = "Tortoise By Name and Year"

if inMethod == methName:
    gp.CreateTable(infolder, "TortSpeeds")
    loc = infolder+"\TortSpeeds"
    gp.AddField(loc,"Name","Text")
    gp.AddField(loc,"MaxSpeed","Double")
    gp.AddField(loc,"StartTime","Text")
    gp.AddField(loc,"EndTime","Text")
    gp.AddField(loc,"Distance","Double")
    gp.AddField(loc,"TimeElapse","Text")

    tortlist = list()
    startSearch = gp.SearchCursor(infeatureClass, "","",infield)
    searchRow = startSearch.Next()
    while searchRow:
        tort = searchRow.GetValue(infield)
        if tort not in tortlist:
            tortlist.append(tort)
            gp.AddMessage(tort) #prints tortoise name
        searchRow = startSearch.Next()
    for tort in tortlist:
        gp.AddMessage("")
        gp.AddMessage(tort)
        gp.AddMessage("-----------")
        tortSearch = gp.SearchCursor(infeatureClass, '"'+infield+'"'+"
                                                      = "+"'"+tort+"'")
        ttimeDiff = 0
        tdistDiff = 0
        ttimeHourDiff = 0
        tspeed = 0
        maxspeed = 0
        tortRow = tortSearch.Next()

        while tortRow:
            dat = datetime.strptime(tortRow.getValue(inDate),"%Y-%m-%d
                                                            %H:%M:%S")
            geom = tortRow.shape
            ex = geom.GetPart()
            origx = ex.x
            origy = ex.y
            ##These two lines would list the position if added back
            ##gp.AddMessage(ex.x)
            ##gp.AddMessage(ex.y)

            #This looks through all remaining positions of the tortoise
            #to get max speed
            original = tortRow.GetValue("ObjectID")
            tortSearch2 = gp.SearchCursor(infeatureClass,
                                    '"'+infield+'"'+" = "+"'"+tort+"'")
            tortRow2 = tortSearch2.Next()
            while tortRow2:
                ##These two rows were to check to ensure the correct
                ##tortoise was being called  torx=
                ##tortRow2.GetValue(infield)
                ##gp.AddMessage(torx)
                ttimeDiff = 0
                dat2 = datetime.strptime(tortRow2.getValue(inDate),"%Y-
                                                      %m-%d %H:%M:%S")
                ttimeDiff = abs(dat2 - dat) #absolute value
                geom2 = tortRow2.shape
                newex = geom2.GetPart()
                newx = newex.x
                newy = newex.y
                tdistDiff = math.sqrt((origx-newx)**2 + (origy-
                                                            newy)**2)
                del newex
                #convert to only hours
                ###Note- timedelta has time in seconds only
                ttimeHourDiff= ttimeDiff.days*24 +
                                                ttimeDiff.seconds/3600
                if ttimeHourDiff > 12:
                        #Was >0, but am changing to 12 due to drastic
                        #movements by rangers within a few hours.
                    tspeed = tdistDiff/ttimeHourDiff
                if maxspeed < tspeed and ttimeHourDiff > 12: #Added
                        #this for same reason as above
                    maxspeed = tspeed
                    maxdate = dat2
                    startday = dat
                    finaldist = tdistDiff
                    finaltime = ttimeDiff
                    finalday = dat2
                tortRow2 = tortSearch2.Next()
            del tortSearch2
            tortRow = tortSearch.Next()
            tspeed = str(maxspeed)
            sdays = str(startday)
            fdays = str(finalday)
            fdist = str(finaldist)
            ftime= str(finaltime)

        #This creates a row in the new table
        newrs = gp.InsertCursor(loc)
        newr = newrs.NewRow()
        newr.SetValue("Name",tort)
        newr.SetValue("MaxSpeed",maxspeed)
        newr.SetValue("StartTime",sdays)
        newr.SetValue("EndTime",fdays)
        newr.SetValue("Distance",finaldist)
        newr.SetValue("TimeElapse",ftime)
        newrs.InsertRow(newr)

        #This prints the info to the output screen
        gp.AddMessage(tort+" has a max speed of: "+tspeed+" meters per
                        hour, beginning on "+sdays+" and ending on
                        "+fdays+" total distance= "+fdist+" total time=
                        "+ftime)

elif inMethod == methYear:
    gp.CreateTable(infolder, "TortSpeedByYr")
    loc = infolder+"\TortSpeedByYr"
    gp.AddField(loc,"Name","Text")
    gp.AddField(loc,"Year","Long")
    gp.AddField(loc,"MaxSpeed","Double")
    gp.AddField(loc,"StartTime","Text")
    gp.AddField(loc,"EndTime","Text")
    gp.AddField(loc,"Distance","Double")
    gp.AddField(loc,"TimeElapse","Text")

    tortlist = list()
    startSearch = gp.SearchCursor(infeatureClass, "","",infield)
    searchRow = startSearch.Next()
    while searchRow:
            tort = searchRow.GetValue(infield)
            if tort not in tortlist:
                tortlist.append(tort)
                gp.AddMessage(tort) #prints tortoise name
            searchRow = startSearch.Next()
    for tort in tortlist:
        gp.AddMessage("")
        gp.AddMessage(tort)
        gp.AddMessage("------------")
        yearlist = list()
```

```
tortSearcher= gp.SearchCursor(infeatureClass, '"'+infield+'"'+"
                              = "+"'"+tort+"'")
tortSearch1 = tortSearcher.Next()
while tortSearch1:
    yrx = datetime.strptime(tortSearch1.GetValue(inDate),"%Y-
                        %m-%d %H:%M:%S")
    yr = yrx.year
    if yr not in yearlist:
        yearlist.append(yr)
    tortSearch1 = tortSearcher.Next()
for tyear in yearlist:
    tortSearch = gp.SearchCursor(infeatureClass,
                        '"'+infield+'"'+" = "+"'"+tort+"'")
    ttimeDiff = 0
    tdistDiff = 0
    ttimeHourDiff = 0
    tspeed = 0
    maxspeed = 0
    tortRow = tortSearch.Next()

    while tortRow:
        yrsx = datetime.strptime(tortRow.getValue(inDate),"%Y-
                             %m-%d %H:%M:%S")
        yrs = yrsx.year
        if tyear == yrs:
            dat = datetime.strptime(tortRow.getValue(inDate),
                                  "%Y-%m-%d %H:%M:%S")
            geom = tortRow.shape
            ex = geom.GetPart()
            origx = ex.x
            origy = ex.y
            ##These two lines would list the position if added
            ##back-
            ##gp.AddMessage(ex.x)
            ##gp.AddMessage(ex.y)

            #This looks through all remaining positions of the
            ##tortoise to get max speed
            original = tortRow.GetValue("ObjectID")
            tortSearch2 = gp.SearchCursor(infeatureClass,
                        '"'+infield+'"'+" = "+"'"+tort+"'")
            tortRow2 = tortSearch2.Next()
            while tortRow2:
                ##These two rows were to check to ensure the
                ##correct tortoise was being called torx=
                ##tortRow2.GetValue(infield)
                ##gp.AddMessage(torx)
                ttimeDiff = 0
                dat2 =
                datetime.strptime(tortRow2.getValue(inDate)
                                 ,"%Y-%m-%d %H:%M:%S")
                ttimeDiff = abs(dat2 - dat) #absolute value
                geom2 = tortRow2.shape
                newex = geom2.GetPart()
                newx = newex.x
                newy = newex.y
                tdistDiff = math.sqrt((origx-newx)**2 + (origy-
                                                            newy)**2)
                        del newex
                        #convert to only hours
                        ###Note- timedelta has time in seconds only
                        ttimeHourDiff= ttimeDiff.days*24 +
                                          ttimeDiff.seconds/3600
                        if ttimeHourDiff > 12:
                            ##Was >0, but am changing to 12 due to
                            ##drastic movements
                            ##by rangers within a few hours.
                            tspeed = tdistDiff/ttimeHourDiff
                        if maxspeed < tspeed and ttimeHourDiff > 12:
                        #Added this for same reason as above
                            maxspeed = tspeed
                            maxdate = dat2
                            startdate = dat
                            finaldist = tdistDiff
                            finaltime = ttimeDiff
                            finalday = dat2
                        tortRow2 = tortSearch2.Next()
                    del tortSearch2
                tortRow = tortSearch.Next()
                tspeed = str(maxspeed)
                sdays = str(startdate)
                fdays = str(finalday)
                fdist = str(finaldist)
                ftime= str(finaltime)

            #This creates a row in the new table
            newrs = gp.InsertCursor(loc)
            newr = newrs.NewRow()
            newr.SetValue("Name",tort)
            newr.SetValue("Year",tyear)
            newr.SetValue("MaxSpeed",maxspeed)
            newr.SetValue("StartTime",sdays)
            newr.SetValue("EndTime",fdays)
            newr.SetValue("Distance",finaldist)
            newr.SetValue("TimeElapse",ftime)
            newrs.InsertRow(newr)

            #This prints the info to the output screen
            gp.AddMessage(tort+" in year: "+str(tyear)+" had a max
                        speed of: "+tspeed+" meters per hour, beginning
                        on "+sdays+" and ending on "+fdays+" total
                        distance= "+fdist+" total time= "+ftime)
            gp.AddMessage("")
gp.AddMessage("____________")
```

Python Code 6: Tortoise Potential Path Tool

```
# ---------------------------------------------------------------------
# potentpathtool4.py
# Created on: Mon May 17 2010
#
# Built By David Turnbull (NGA), University of Redlands MS GIS for
# Joshua Tree National Park as part of his Master's Project
# Modified: 6/12/2010
# ---------------------------------------------------------------------

# Import system modules
import sys, string, os, arcgisscripting, math
from datetime import date, datetime, timedelta
from operator import itemgetter

# Create the Geoprocessor object
gp = arcgisscripting.create()
gp.overwriteoutput =1

#This is added to see if ArcGIS 10 or later is being used. The ellipse
too
#function differs after ArcGIS 9.3
installD = gp.GetInstallInfo("desktop")
for key in installD.keys():
    if key == "Version":
        gp.AddMessage("ArcGIS Version: "+installD[key])
        thisVersion = float(installD[key])

# Load required toolboxes...
gp.AddToolbox("C:/Program Files
            (x86)/ArcGIS/ArcToolbox/Toolboxes/Analysis Tools.tbx")
# Local variables...
gp.addmessage("Setting Local Variables...")
infeatureClass = gp.GetParameterAsText(0)
inName = gp.GetParameterAsText(1)
inDate = gp.GetParameterAsText(2)
opt1table = gp.GetParameterAsText(3)
opt2speed = gp.GetParameterAsText(4)
outworkspace = gp.GetParameterAsText(5)
potpathlim = gp.GetParameterAsText(6)
hrlim = gp.GetParameterAsText(7)
distlim = gp.GetParameterAsText(8)
wgsutm = gp.GetParameterAsText(9)
symangle = gp.GetParameterAsText(10)

dlim = float(distlim)
hlim = int(hrlim)

#Gets added to end of feature class name unless changed
if potpathlim == "true":
    tag = "_LtdH"+str(hrlim)+"D"+str(distlim)
else:
    tag = "_Unlimited"

gp.Workspace = outworkspace
```

```
#Create SymAngle Field
if symangle == "true":
    gp.AddField(infeatureClass,"SymAngle","Short")

#Fetch Datum, UTM zone and central meridian of dataset
desc = gp.describe(infeatureClass)
sr = desc.SpatialReference
projn = sr.Name
values = projn.split("_")
datm = values[0] + "_" + values[1]
zone = values[4]
lenzone = len(zone)
znum = zone[0:lenzone-1]
zstr = zone[lenzone-1]
zonecomma = str(znum)+","+zstr+","
cm = sr.CentralMeridian
gp.AddMessage("Source Datum: "+datm)
gp.AddMessage("Source UTM Zone: "+str(zone)+
"("+str(znum)+","+str(zstr)+")")
gp.AddMessage("Central Meridian: "+str(cm))
if (zstr != "N" and zstr != "S"):
    gp.AddMessage("The source data must be in UTM!!!")
    gp.AddMessage("Your results will be incorrect.")

#Begin...
tortlist = list()
startSearch = gp.SearchCursor(infeatureClass, "","",inName)
gp.AddMessage("")
searchRow = startSearch.Next()
while searchRow:
    tort = searchRow.GetValue(inName)
    if tort not in tortlist:
        tortlist.append(tort)
        gp.AddMessage(tort) #prints tortoise name
    searchRow = startSearch.Next()
gp.AddMessage("-----------")
for tort in tortlist:
    #get tortoise speed from table
    if opt1table != "":
        spdSrch = gp.SearchCursor(opt1table, '"Name"'+" =
                                  "+"'"+tort+"'")
        gp.AddMessage(tort)
        spdRow = spdSrch.Next()
        while spdRow:
            spd = spdRow.GetValue("MaxSpeed")
            spdRow = spdSrch.Next()
    elif opt2speed != "":
        spd = float(opt2speed)
    else:
        gp.AddError("Please Select an Option")

    #Make individual Tortoise Table for each tortoise
    gp.CreateTable(outworkspace, tort+"_ellipse_table"+tag)
    loc = outworkspace+"/"+tort+"_ellipse_table"+tag
    gp.AddField(loc,"Name","Text")
    gp.AddField(loc,"CenterXY","Text")
    gp.AddField(loc,"CenterX","Text") ## This field and the next will
                                      ## be needed for ArcGIS 10
    gp.AddField(loc,"CenterY","Text") ## "
    gp.AddField(loc,"SemiMinorAxis","Double")
    gp.AddField(loc,"SemiMajorAxis","Double")
    gp.AddField(loc,"Rotation","Double")
    gp.AddField(loc,"TimeStamp","Text")
    gp.AddField(loc,"EndTime","Text")


    #search tortoise info...
    TimeRows= gp.SearchCursor(infeatureClass, '"'+inName+'"'+" =
                              "+"'"+tort+"'")

                #Creating an empty list
    gp.AddMessage("-----------")
    dlist = list()
    TR = TimeRows.Next()
    while TR:
        time = TR.GetValue(inDate)
        if time not in dlist:
            dlist.append(time)
        TR = TimeRows.Next()

    resultx = sorted(dlist, reverse=False)
    n = len(resultx)
    for i in range(n):
        gp.AddMessage(str(resultx[i]))
        i = i + 1

    for i in range(n-1): #no action for last feature
        datew = resultx[i]
        datex = datetime.strptime(datew, "%Y-%m-%d %H:%M:%S")
        gp.AddMessage("-----------")

        ##DR = DateRows.Next()
        ##while DR:
        datey = datetime.strptime(resultx[i+1], "%Y-%m-%d %H:%M:%S")
        gp.AddMessage(str(datex)+" to "+str(datey))
        ###TimeDelta between two positions  VARIABLE 1
        datediff = abs(datey - datex)
        gp.AddMessage("Time Difference = "+str(datediff))
        ###Potential Distance between two positions VARIABLE 2
        ddatediff = datediff.days*24
        dtimediff = datediff.seconds*1.00/3600.00
        totdatediff = ddatediff+dtimediff
        gp.AddMessage("Time Difference in Hours = "+ str(totdatediff))
        potdist = spd * totdatediff
        gp.AddMessage("Tortoise Max Speed: "+ str(spd))
        gp.AddMessage("Potential Distance = "+str(potdist))
        semimajorAxis = potdist/2 ##############Semi-Major Axis
        gp.AddMessage("Semi-Major Axis: "+str(semimajorAxis)+ "
                      meters")
        DateRows = gp.SearchCursor(infeatureClass, '"'+inName+'"'+" =
                              "+"'"+tort+"'"+" and "+'"'+inDate+'"'+" =
                              "+"'"+str(datex)+"'")
```

```
#Get x and y here of i
DR = DateRows.Next()
while DR:
    geom = DR.shape
    ex = geom.GetPart()
    origx = ex.x
    origy = ex.y

    NewDateRows = gp.SearchCursor(infeatureClass,
                '"'+inName+'"'+" = "+"'"+tort+"'"+" and
                "+'"'+inDate+'"'+" = "+"'"+str(datey)+"'")
    NDR = NewDateRows.Next()
    while NDR:
        ngeom = NDR.shape
        nex = ngeom.GetPart()
        norigx = nex.x
        norigy = nex.y
        NDR = NewDateRows.Next()
    ###Semi-Minor Axis Calculation
    ##SIDE 1: ea
    tdistDiff = math.sqrt((origx-norigx)**2 + (origy-
                      norigy)**2)
    gp.AddMessage("z distance = "+str(tdistDiff))
    ea = tdistDiff/2

    if (potpathlim == "true" and hrlim != "" and distlim !=
                                                      ""):
        if ((tdistDiff < dlim) and (totdatediff > hlim)):
            ##This ensures that recorded positions are larger
            ##than x hrs apart.
            gp.AddMessage("Potential Distance has been limited
                      to 1 day for large date gaps >
                      "+str(dlim)+" hours and movement
                      less than "+str(distlim)+"
                      meters.")
            potdist = spd * 24
            gp.AddMessage("Limited Potential Distance =
                      "+str(potdist))
            semimajorAxis = potdist/2 ###############Limited
                              #######Semi-Major Axis
            gp.AddMessage("Limited Semi-Major Axis:
                    "+str(semimajorAxis)+ " meters")

    #Pythagorean ###############Semi-Minor Axis
    ma = ((2*semimajorAxis)-(2*ea))
    if ma <= 0.0:
        semiminorAxis = 1.0
    else:
        semiminorAxis = math.sqrt((semimajorAxis**2)-(ea**2))
    gp.AddMessage("Semi-Minor Axis: "+str(semiminorAxis)+ "
                 meters")

    ###Center Point #######################CENTER POINT
    midX = int((origx + norigx)/2)
    midY = int((origy + norigy)/2)
    gp.AddMessage("Start Point at: x= "+str(origx)+" y=
                 "+str(origy))
```

```
gp.AddMessage("End Point at: x= "+str(norigx)+" y=
                  "+str(norigy))
gp.AddMessage("Center at: x= "+str(midX)+" y= "+str(midY))

###Angle between the two points: ROTATION ANGLE, in degrees
###from positive X axis
pi = math.pi
if (norigy-origy) != 0.00: #Have to account for the div/0

    theta = math.atan((norigx-origx)/(norigy-origy))
                        # angle in radians of major axis
elif ((norigy-origy) == 0.00) and ((norigx-origx) >= 0):
    rotAngle = 180 #rotation angle of minor axis
elif ((norigy-origy) == 0.00) and ((norigx-origx) < 0):
    rotAngle = 360
#Degrees and minor
if ((norigx-origx) >= 0.00) and ((norigy-origy) > 0.00):
    rotAngle = theta*180/pi + 90 # angle in degrees of
               minor axis (0 + 90 for major to minor)
elif ((norigx-origx) >= 0.00) and ((norigy-origy) < 0.00):
    rotAngle = theta*180/pi + 270
elif ((norigx-origx) < 0.00) and ((norigy-origy) < 0.00):
    rotAngle = theta*180/pi + 270
elif ((norigx-origx) < 0.00) and ((norigy-origy) > 0.00):
    rotAngle = theta*180/pi + 450 #Have to add 360 here for
                  #logical symbol angle later when I
                  #subtract 90 or for ArcGIS 10
gp.AddMessage(str(theta))

if thisVersion != 9.3: #ArcGIS 10 measures from Yaxis to
                       #Major Axis; whereas Military
                       #Analyst 9.3 is to minor axis
    rotAngle = rotAngle - 90
gp.AddMessage("angle = "+str(rotAngle))

###Populate original tortoise file with symangle for symbol
###direction
if symangle == "true":
    AngleRows = gp.UpdateCursor(infeatureClass,
      '"'+inName+'"'+" = "+"'"+tort+"'"+" and
      "+'"'+inDate+'"'+" = "+"'"+str(datex)+"'")
    AR = AngleRows.Next()
    while AR:
        AR.SetValue("SymAngle", round(rotAngle-90, 0))
        AngleRows.UpdateRow(AR)
        AR = AngleRows.Next()

###Insert rows into table
nrow = gp.InsertCursor(tort+"_ellipse_table"+tag)
nrows = nrow.NewRow()
nrows.SetValue("Name", tort)
nrows.SetValue("CenterXY",
                  zonecomma+str(midX)+","+str(midY))
nrows.SetValue("CenterX", str(midX))## This field and the
                        ##next will be useful for ArcGIS 10
nrows.SetValue("CenterY", str(midY)) ## "
nrows.SetValue("SemiMinorAxis", semiminorAxis)
```

```
        nrows.SetValue("SemiMajorAxis", semimajorAxis)
        nrows.SetValue("Rotation", rotAngle)
        nrows.SetValue("TimeStamp", str(datex))
        nrows.SetValue("EndTime", str(datey))
        gp.AddMessage("")
        nrow.InsertRow(nrows)
        del nrow
        DR = DateRows.Next()

    i = i + 1

#Build Ellipse Feature Class for this tortoise
if thisVersion == 9.3:
    inEllipseTable = tort+"_ellipse_table"+tag
    EllipseFCWGS = tort+"_PotPath_WGS"+tag
    EllipseFCUTM = tort+"_PotPath_UTM"+tag
    gp.AddMessage("Creating Ellipses")
    gp.TableToEllipse(inEllipseTable, EllipseFCWGS, "UTM",
                "CenterXY", "", "SemiMinorAxis",
                "SemiMajorAxis", "Meters", "Semi", "Rotation",
                "Degrees")
    gp.AddMessage("-----------")
    if wgsutm != "true":
        gp.Project_management(EllipseFCWGS, EllipseFCUTM,
              "PROJCS['"+datm+"_UTM_Zone_"+zone+"',GEOGCS['GCS_Nort
              h_American_1983',DATUM['D_North_American_1983',SPHERO
              ID['GRS_1980',6378137.0,298.257222101]],PRIMEM['Green
              wich',0.0],UNIT['Degree',0.0174532925199433]],PROJECT
              ION['Transverse_Mercator'],PARAMETER['False_Easting',
              500000.0],PARAMETER['False_Northing',0.0],PARAMETER['
              Central_Meridian',"+str(cm)+"],PARAMETER['Scale_Facto
              r',0.9996],PARAMETER['Latitude_Of_Origin',0.0],UNIT['
              Meter',1.0]]", "NAD_1983_To_WGS_1984_5",
              "GEOGCS['GCS_WGS_1984',DATUM['D_WGS_1984',SPHEROID['W
              GS_1984',6378137.0,298.257223563]],PRIMEM['Greenwich'
              ,0.0],UNIT['Degree',0.0174532925199433]]")

        #Deleting the WGS feature class and table, they are not
        #needed, comment them if desired
        gp.Delete_management(inEllipseTable)
        gp.Delete_management(EllipseFCWGS)
    else:
        gp.Delete_management(inEllipseTable)

if thisVersion != 9.3:
  arcpy.TableToEllipse_management(tort+"_ellipse_table"+tag,
  tort+"_PotPath_UTM"+tag, "CenterX", "CenterY", "SemiMajorAxis",
  "SemiMinorAxis", "METERS", "Rotation", "DEGREES", "OBJECTID",
  "PROJCS['"+datm+"_UTM_Zone_"+zone+"',GEOGCS['GCS_North_American_1
  983',DATUM['D_North_American_1983',SPHEROID['GRS_1980',6378137.0,
  298.257222101]],PRIMEM['Greenwich',0.0],UNIT['Degree',0.017453292
  5199433]],PROJECTION['Transverse_Mercator'],PARAMETER['False_East
  ing',500000.0],PARAMETER['False_Northing',0.0],PARAMETER['Central
  _Meridian',"+str(cm)+"],PARAMETER['Scale_Factor',0.9996],PARAMETE
  R['Latitude_Of_Origin',0.0],UNIT['Meter',1.0]];-5120900 -9998100
  10000;-100000 10000;-100000
  10000;0.001;0.001;0.001;IsHighPrecision")
```

```
    del dlist

gp.AddMessage("____________")
```

Python Code 7: Lost Tortoise Tool

```
# ---------------------------------------------------------------------------
# losttortoise.py
# Created on: Mon May 17 2010
#
# Built By David Turnbull (NGA), University of Redlands MS GIS for
# Joshua Tree National Park as part of his Master's Project
# Updated: 07/27/2010
# ---------------------------------------------------------------------------

# Import system modules
import sys, string, os, arcgisscripting
from datetime import date, datetime, timedelta, time

# Create the Geoprocessor object
gp = arcgisscripting.create()
gp.overwriteoutput =1

# Load required toolboxes...
gp.AddToolbox("C:/Program Files
(x86)/ArcGIS/ArcToolbox/Toolboxes/Analysis Tools.tbx")

# Local variables...
gp.addmessage("Setting Local Variables...")
tortname = gp.GetParameterAsText(0)
torttable = gp.GetParameterAsText(1)
tortspeed2 =gp.GetParameterAsText(2)
spatial = gp.GetParameterAsText(3)
xPos = gp.GetParameterAsText(4)
yPos = gp.GetParameterAsText(5)
date1 = gp.GetParameterAsText(6)
date2 = gp.GetParameterAsText(7)
clipArea = gp.GetparameterAsText(8)
eraseArea = gp.GetparameterAsText(9)
workspace = gp.GetParameterAsText(10)

gp.workspace = workspace

# Process: Buffer...
if (torttable != "" and tortspeed2 != ""):
    gp.AddMessage("Either Enter Option1 OR Option2")

elif (torttable != "" or tortspeed2 != ""):
    gp.addmessage("")
    dat1 = datetime.strptime(date1,"%m/%d/%Y %I:%M:%S %p")
    dat2 = datetime.strptime(date2,"%m/%d/%Y %I:%M:%S %p")

    timediff = (dat2 - dat1)
    gp.addmessage("")

    #Get tortoise speed from table
    if (torttable != ""):
        gp.AddMessage(tortname)
        gp.addmessage("Source Table: "+ torttable)
        gp.addmessage("Output Feature Class: Lost_"+ tortname+"_Buffer
                              in "+workspace)
        spdSrch = gp.SearchCursor(torttable, '"Name"'+" =
                              "+"'"+tortname+"'")
        gp.addmessage(str(timediff))
        spdRow = spdSrch.Next()
        while spdRow:
            tortspeed1 = spdRow.GetValue("MaxSpeed")
            spdRow = spdSrch.Next()
        gp.addmessage("Creating Tortoise Location Buffer Based on Table
                              Speed: "+str(tortspeed1)+" ...")
    elif(tortspeed2 != ""):
        tortspeed1 = float(tortspeed2)
        gp.addmessage("Creating Tortoise Location Buffer Based on Input
                              Speed: "+str(tortspeed1)+" ...")

    thours = timediff.days * 24.0
    tortdist1 = tortspeed1 * thours

    #Create a feature class here
    gp.AddMessage("Creating Tortoise Possible Location Buffer...")
    fc = gp.CreateFeatureClass(workspace,
                              "Lost_"+tortname+"_LastKnownPos",
                              "POINT", "", "","", spatial)
    gp.AddField(fc, "Name", "Text")
    gp.AddField(fc, "CurrDate", "Text")
    gp.AddField(fc, "LastSeen", "Text")
    gp.AddField(fc, "TimePass", "Text")
    gp.AddField(fc, "MaxSpeed", "Double")
    gp.AddField(fc, "SearchRadius", "Double")
    gp.AddField(fc, "RadiusUnits", "Text")

   #Make a point feature here
    newpt = gp.InsertCursor(fc)
    newrow = newpt.NewRow()
    pnt = gp.CreateObject("Point")
    newrow.SetValue("Name", tortname)
    newrow.SetValue("CurrDate", str(dat2))
    newrow.SetValue("LastSeen", str(dat1))
    newrow.SetValue("TimePass", str(timediff))
    newrow.SetValue("MaxSpeed", tortspeed1)
    newrow.SetValue("SearchRadius", tortdist1)
    newrow.SetValue("RadiusUnits", "Meters")
    pnt.x = xPos
    pnt.y = yPos
    newrow.shape = pnt
    newpt.InsertRow(newrow)

    del newpt
    del newrow

    #Creating Buffer
    outx = "Lost_"+tortname+"_Buffer"
    gp.AddMessage("")
    gp.AddMessage("Creating Output Feature Classes...")
    gp.AddMessage("Creating Circular Buffer: "+outx)
    gp.Buffer_analysis(fc, outx, tortdist1, "", "", "NONE")
```

```
    if clipArea != "":
        outx = "Lost_"+tortname+"_Buffer_Clipped" #This way due to
                                                  #optional erase
        gp.AddMessage("Clipping Circular Buffer to new feature class:
                              "+outx)
        gp.Clip_analysis("Lost_"+tortname+"_Buffer", clipArea, outx,
                              "")

    if eraseArea != "":
        gp.AddMessage("Erasing Portion of Buffer to new feature class:
                              "+outx+"_Erase")
        gp.Erase_analysis(outx, eraseArea, outx+"_Erase","")
else:
    gp.addmessage("")
    gp.addmessage("Nothing Was Selected or necessary fields not
                              populated!")
    gp.addmessage("")

gp.AddMessage("____________")
```

Appendix B. Toolbox Help Documents

Tortoise Import Tool

This tool was built by David Turnbull, NGA, as part of his thesis at the University of Redlands (2010). This tool was designed for his client: Joshua Tree National Park, but can be used for manipulating any dynamic data to prepare it for dynamic visualizations in ArcGIS Explorer and Google Earth.

Command line syntax

TortImport <Select_Feature_Class_> <Output_Workspace_> <Create Feature Classes By Tortoise | Create Feature Classes By Toroise and Year | Create All Tortoises By Year | Create All Three Types of Feature Classes Above> <Select_Name_Sort_Field_> {Select_Date_Sort_Field_} {Add_Tag_To_File_Name_}

Parameters

Expression	**Explanation**
<Select_Feature_Class_>	Select a feature class that you would like to create multiple sub-feature classes from.
<Output_Workspace_>	Select the Geodatabase or folder location for the feature classes to be created in.
<Create Feature Classes By Tortoise | Create Feature Classes By Toroise and Year | Create All Tortoises By Year | Create All Three Types of Feature Classes Above>	Options: • Create Feature Classes By Tortoise-This will create a single feature class containing all positions for each tortoise. • Create Feature Classes By Toroise and Year- This will create multiple feature classes by year for each tortoise. • Create All Tortoises By Year- This will create a feature class for each year containing all tortoise positions for that year. • Create All Three Types of Feature Classes Above- This will create all of the feature classes described above.
<Select_Name_Sort_Field_>	Select the field for the Tortoise Name.

{Select_Date_Sort_Field_}	Select the Date Field.
{Add_Tag_To_File_Name_}	Optional: Add a tag name that gets appended to the beginning of the output file name.

Scripting syntax

TortImport (Select_Feature_Class_, Output_Workspace_, Select_Method_, Select_Name_Sort_Field_, Select_Date_Sort_Field_, Add_Tag_To_File_Name_)

Parameters

Expression	**Explanation**
Select Feature Class: (Required)	Select a feature class that you would like to create multiple sub-feature classes from.
Output Workspace: (Required)	Select the Geodatabase or folder location for the feature classes to be created in.
Select Method: (Required)	Options: • Create Feature Classes By Tortoise-This will create a single feature class containing all positions for each tortoise. • Create Feature Classes By Toroise and Year- This will create multiple feature classes by year for each tortoise. • Create All Tortoises By Year- This will create a feature class for each year containing all tortoise positions for that year. • Create All Three Types of Feature Classes Above- This will create all of the feature classes described above.
Select Name Sort Field: (Required)	Select the field for the Tortoise Name.
Select Date Sort Field: (Optional)	Select the Date Field.
Add Tag To File Name? (Optional)	Optional: Add a tag name that gets appended to the beginning of the output file name.

TimeStamp and EndTime Field Creator

This tool was built by David Turnbull, NGA, as part of his thesis at the University of Redlands (2010). This tool was designed for his client: Joshua Tree National Park, but can be used for manipulating any dynamic data to prepare it for dynamic visualizations in ArcGIS Explorer and Google Earth.

This tool is used to add a TimeStamp attribute to a feature class as well as copy existing time information from another field to the new attribute. In some cases, it will create an attribute called TimeStamp. This tool also has the option to automatically create a TimeEnd attribute by either adding a number of days to the Start Time or by populating the EndTime with the Start Time of the next event's Start Time.

Command line syntax

TimeStampFieldCreator <Select_Feature_Class_> {Select_Event__Tortoise_Name__} <Select_Field_with_Date_> {Select_Field_with_Time_} <EndTime is StartTime of Next Occuring Feature | EndTime is Start of Next Day(s) | No EndTime Calculation> {TimeEnd_Days__decimal_days_ok__}

Parameters

Expression	Explanation
<Select_Feature_Class_>	Select a feature class that has a DATE attribute.
{Select_Event__Tortoise_Name__}	Enter the field to sub-sort by if feature class contains many event categories. In this case, select the field that contains the tortoise name.
<Select_Field_with_Date_>	Select the FIELD from the Input Feature Class that has a DATE attribute.
{Select_Field_with_Time_}	OPTIONAL- Select the field that has a Time in it. The field must be either a Date Field or a Double Field.
<EndTime is StartTime of Next Occurring Feature | EndTime is Start of Next Day(s) | No EndTime Calculation>	Options: • EndTime is StartTime of Next Occurring Feature- This will populate EndTime with the next event date for your particular tortoise. This is beneficial when wanting to view in Google Earth and have the tortoise remain on screen in between events. • EndTime is Start of Next Day(s)- This will populate

	the EndTime with a date that is the number of days specified in the last optional field. The default value is 1 day. This value can be in decimal days as well. The decimal days will be converted to time. • No EndTime Calculation- This will not produce a TimeEnd Attribute. If one already exists within the data, it will remain and be unaffected.
{TimeEnd_Days__decimal_days_ok__}	This is only needed for the second method. Enter the number of days. This value can be in decimal days as well- The decimal days will be converted to time.

Command Line Example

Scripting syntax

TimeStampFieldCreator (Select_Feature_Class_, Select_Event__Tortoise_Name__, Select_Field_with_Date_, Select_Field_with_Time_, TimeEnd_Method_, TimeEnd_Days__decimal_days_ok__)

Parameters

Expression	**Explanation**
Select Feature Class: (Required)	Select a feature class that has a DATE attribute.
Select Event (Tortoise Name): (Optional)	Enter the field to sub-sort by if feature class contains many event categories. In this case, select the field that contains the tortoise name.
Select Field with Date: (Required)	Select the FIELD from the Input Feature Class that has a DATE attribute.
Select Field with Time: (Optional)	OPTIONAL- Select the field that has a Time in it. The field must be either a Date Field or a Double Field.
TimeEnd Method: (Required)	Options: • EndTime is StartTime of Next Occurring Feature- This will populate EndTime with the next event date for your particular tortoise. This is beneficial when wanting to view in Google Earth and have the tortoise remain on screen in between events. • EndTime is Start of Next Day(s)- This will populate the EndTime with a date that is the number of days specified in the last optional field. The default value is 1 day. This value can be in decimal days as well. The decimal days will be converted to time. • No EndTime Calculation- This will not produce a TimeEnd Attribute. If one already exists within the data, it will remain and be unaffected.

TimeEnd Days (decimal days ok): (Optional)	This is only needed for the second method. Enter the number of days. This value can be in decimal days as well- The decimal days will be converted to time.

Stationary Event from Table

This tool was built by David Turnbull, NGA, as part of his thesis at the University of Redlands (2010). This tool was designed for his client: Joshua Tree National Park, but can be used for manipulating any dynamic data to prepare it for dynamic visualizations in ArcGIS Explorer and Google Earth.

Usage Tips

Need: feature class with at least one geometry (any type) & a non-geographic table that contains a date or datetime field.

Command line syntax

AreaPoly2Weather22 <Input_Feature_Class_> <Input_Table_> <Input_Object_id_> <Temperature-Daily High and Low | Temperature-Daily Average | Temperature-Hourly- matches table records (Large dataset possible) | Precipitation-Daily Total | Other- matches table records> {Table_Field_for_Date____Required_for_Average_or_Hi_Low_Calculations_} {Table_Field_for_Aver_Hi_Low___Required_for_Average_or_Hi_Low_Calculations_} {Include_English_Units_in_Output_Feature_Class__} <Work_Space_> {Enter_Optional_Output_File_Tag_}

Parameters

Expression	**Explanation**
<Input_Feature_Class_>	Select the Feature Class that you will be grabbing the single geometry from to copy it for multiple dates or datetimes.
<Input_Table_>	Select the non-geographic table of data that has a date or datetime field. These records will be copied to the duplicated geometries of the feature from the input feature class.
<Input_Object_id_>	Please enter the numeric Object ID of the feature from the Input Feature Class that you want to duplicate.
<Temperature-Daily High and Low | Temperature-Daily Average | Temperature-Hourly- matches table records (Large dataset possible) | Precipitation-Daily Total | Other- matches table records>	Options: • Temperature High and Low: This will produce two feature classes, a feature class, which has a single record for each day with the High Temperature of that day, and a feature class, which has a single record for each day with the Low Temperature of that day.

	• Average Daily Temperature: This will produce a feature class that has a single record for each day with the Average Temperature of that day. • Hourly temperature: This basically is the same as the "Other" option. If the data from the table are by the hour, it merely creates a feature for each row of the table. The number of objects in the output feature class will equal the number of records in the table. • Daily Precipitation: This will produce a feature class that has a single record for each day with the Total Precipitation of that day. • Other: This option is here to allow utilization for any non-geographic table that has a date attribute and there is a need to tie the records to a geometry. It merely creates a feature for each row of the table. The number of objects in the output feature class will equal the number of records in the table.
{Table_Field_for_Date____Required_for_Average_or_Hi_Low_Calculations_}	Select the field from the table that has the date or datetime attribute.
{Table_Field_for_Aver_Hi_Low___Required_for_Average_or_Hi_Low_Calculations_}	Select the field from the table that has the desired value to get the Hi/Low, Daily Average, or Daily Total from. This also can be used for the "Other" option. Any field that is numeric can be summarized.
{Include_English_Units_in_Output_Feature_Class__}	This option will add a field to the feature class that contains the values in English units, in addition to the metric field.
<Work_Space_>	Please select the Geodatabase or workspace folder that the outputs will be put into. Note that the outputs may not be automatically loaded into your view.
{Enter_Optional_Output_File_Tag_}	Optional: Add a tag name that gets appended to the beginning of the output file name.

Command Line Example

Scripting syntax

AreaPoly2Weather22 (Input_Feature_Class_, Input_Table_, Input_Object_id_, Data_Type_of_Table_and_Frequency_Requested_, Table_Field_for_Date____Required_for_Average_or_Hi_Low_Calculations_, Table_Field_for_Aver_Hi_Low___Required_for_Average_or_Hi_Low_Calculations_, Include_English_Units_in_Output_Feature_Class__, Work_Space_, Enter_Optional_Output_File_Tag_)

Parameters

Expression	Explanation

Input Feature Class: (Required)	Select the Feature Class that you will be grabbing the single geometry from to copy it for multiple dates or datetimes.
Input Table: (Required)	Select the non-geographic table of data that has a date or datetime field. These records will be copied to the duplicated geometries of the feature from the input feature class.
Input Object id: (Required)	Please enter the numeric Object ID of the feature from the Input Feature Class that you want to duplicate.
Data Type of Table and Frequency Requested? (Required)	Options: • Temperature High and Low: This will produce two feature classes, a feature class, which has a single record for each day with the High Temperature of that day, and a feature class, which has a single record for each day with the Low Temperature of that day. • Average Daily Temperature: This will produce a feature class that has a single record for each day with the Average Temperature of that day. • Hourly temperature: This basically is the same as the "Other" option. If the data from the table are by the hour, it merely creates a feature for each row of the table. The number of objects in the output feature class will equal the number of records in the table. • Daily Precipitation: This will produce a feature class that has a single record for each day with the Total Precipitation of that day. • Other: This option is here to allow utilization for any non-geographic table that has a date attribute and there is a need to tie the records to a geometry. It merely creates a feature for each row of the table. The number of objects in the output feature class will equal the number of records in the table.
Table Field for Date: (Required for Average or Hi/Low Calculations) (Optional)	Select the field from the table that has the date or datetime attribute.
Table Field for Aver/Hi Low: (Required for Average or Hi/Low Calculations) (Optional)	Select the field from the table that has the desired value to get either the Hi/Low, Daily Average or Daily Total from. This also can be used for the "Other" option. Any field that is numeric can be summarized.

Include English Units in Output Feature Class?: (Optional)	This option will add a field to the feature class that contains the values in English units, in addition to the metric field.
Work Space: (Required)	Please select the Geodatabase or workspace folder that the outputs will be put into. Note that the outputs may not be automatically loaded into your view.
Enter Optional Output File Tag: (Optional)	Optional: Add a tag name that gets appended to the beginning of the output file name.

Road and Trot Buffer Tool

This tool was built by David Turnbull, NGA, as part of his thesis at the University of Redlands (2010). This tool was designed for his client: Joshua Tree National Park, but can be used for manipulating any dynamic data to prepare it for dynamic visualizations in ArcGIS Explorer and Google Earth.

This tool is used to calculate one or two BUFFERS from the roads and one buffer for the Tortoise Trots. The default values are 0.8 kilometers and 3.5 kilometers for the roads and 0.5 kilometers for the trots. The default values used for roads reflect the values in the study by Boarman, Sazaki, and Jennings (1997). The study suggests that the areas within 0.8km of the road may be more beneficial to study as well as a secondary buffer of up to 3.5km from the road should be considered. This is a result of tortoises being hit, but travelling a distance before the effects of being hit by a vehicle cause them to die. -- Boarman, W. I., Sazaki, M., & Jennings, W. B. (1997). The Effect of Roads, Barrier Fences, and Culverts on Desert Tortoise Populations in California, USA. Conservation, Restoration, and Management of Tortoises and Turtles - An International Conference (pp. 54-58). New York Turtle and Tortoise Society.

Command line syntax

BufferTool {Select_Road_Feature_Class} {Build_Close_Buffer_} <Select_Buffer_Tolerance_1__close_proximity_> {Build_Far_Buffer_} <Select_Buffer_Tolerance_2__farther_> <Build_Trot_Buffer_> {Tortoise_Trot_data} {Enter_Trot_Buffer_Size} <Output_Workspace> {Enter_a_filename_tag_if_desired_}

Parameters

Expression	**Explanation**
{Select_Road_Feature_Class}	Select the Road feature class that you would like to create buffers from
{Build_Close_Buffer_}	Check this box if you want the tool to build this buffer
<Select_Buffer_Tolerance_1__close_proximity_>	Buffer #1 tolerance. Default is 0.8 Km.
{Build_Far_Buffer_}	Check this box if you want the tool to build this buffer
<Select_Buffer_Tolerance_2__farther_>	Buffer #2 tolerance. Default is 3.5 Km.
<Build_Trot_Buffer_>	Check this box if you want the tool to build this buffer
{Tortoise_Trot_data}	Select the feature class that contains the Tortoise Trots

{Enter_Trot_Buffer_Size}	Enter the Tortoise Trot Buffer Size
<Output_Workspace>	Select the output geodatabase or feature dataset.
{Enter_a_filename_tag_if_desired_}	Optional- Enter a tag name that will be appended to the front of the output file name followed by a "_".

Scripting syntax

BufferTool (Select_Road_Feature_Class, Build_Close_Buffer_, Select_Buffer_Tolerance_1__close_proximity_, Build_Far_Buffer_, Select_Buffer_Tolerance_2__farther_, Build_Trot_Buffer_, Tortoise_Trot_data, Enter_Trot_Buffer_Size, Output_Workspace, Enter_a_filename_tag_if_desired_)

Parameters

Expression	**Explanation**
Select Road Feature Class (Optional)	Select the Road feature class that you would like to create buffers from
Build Close Buffer? (Optional)	Check this box if you want the tool to build this buffer
Select Buffer Tolerance 1 (close proximity) (Required)	Buffer #1 tolerance. Default is 0.8 Km.
Build Far Buffer? (Optional)	Check this box if you want the tool to build this buffer
Select Buffer Tolerance 2 (farther) (Required)	Buffer #2 tolerance. Default is 3.5 Km.
Build Trot Buffer? (Required)	Check this box if you want the tool to build this buffer
Tortoise Trot data (Optional)	Select the feature class that contains the Tortoise Trots
Enter Trot Buffer Size (Optional)	Enter the Tortoise Trot Buffer Size
Output Workspace (Required)	Select the output geodatabase or feature dataset.
Enter a filename tag if desired: (Optional)	Optional- Enter a tag name that will be appended to the front of the output file name followed by a "_".

Tortoise Maximum Speed Finder

This tool was built by David Turnbull, NGA, as part of his thesis at the University of Redlands (2010). This tool was designed for his client: Joshua Tree National Park, but can be used for manipulating any dynamic data to prepare it for dynamic visualizations in ArcGIS Explorer and Google Earth.

Usage Tips

***The Timestamp Creator Tool should be run before running this tool.

Command line syntax

tortspeed <Input_Tortoise_Feature_Class_> <Input_Tortoise_Name_Field_> <Tortoise By Name | Tortoise By Name and Year> <Select_TimeStamp_Field_> <Ouput_location_of_Table_>

Parameters

Expression	**Explanation**
<Input_Tortoise_Feature_Class_>	Select a tortoise positions feature class that contains tortoise positions for one or more tortoises. ***This feature class should have had the TimeStamp Field Creator Tool run on it prior.
<Input_Tortoise_Name_Field_>	Select the Field from the list that contains the Tortoise Names.
<Tortoise By Name | Tortoise By Name and Year>	Options: • Tortoise By Name- Calculates the maximum speed achieved by each particular tortoise over the period covered in the feature class. • Tortoise By Name and Year- Calculates the maximum speed achieved by each particular tortoise for each year.
<Select_TimeStamp_Field_>	Select the TimeStamp field. If it does not exist, the TimeStamp Creator tool needs to be run first.
<Ouput_location_of_Table_>	Select the Geodatabase or Feature Dataset where you want the output table to be created. • Tortoise By Name: Output File will be called

	"TortSpeeds" • Tortoise By Name and Year: Output File will be called "TortSpeedByYr"

Scripting syntax

tortspeed (Input_Tortoise_Feature_Class_, Input_Tortoise_Name_Field_, Method_, Select_TimeStamp_Field_, Ouput_location_of_Table_)

Parameters

Expression	**Explanation**
Input Tortoise Feature Class: (Required)	Select a tortoise positions feature class that contains tortoise positions for one or more tortoises. ***This feature class should have had the TimeStamp Field Creator Tool run on it prior.
Input Tortoise Name Field: (Required)	Select the Field from the list that contains the Tortoise Names.
Method: (Required)	Options: • Tortoise By Name- Calculates the maximum speed achieved by each particular tortoise over the period covered in the feature class. • Tortoise By Name and Year- Calculates the maximum speed achieved by each particular tortoise for each year.
Select TimeStamp Field: (Required)	Select the TimeStamp field. If it does not exist, the TimeStamp Creator tool needs to be run first.
Output location of Table: (Required)	Select the Geodatabase or Feature Dataset where you want the output table to be created. • Tortoise By Name: Output File will be called "TortSpeeds" • Tortoise By Name and Year: Output File will be called "TortSpeedByYr"

Tortoise Potential Path Tool

This tool was built by David Turnbull, NGA, as part of his thesis at the University of Redlands (2010). This tool was designed for his client: Joshua Tree National Park, but can be used for manipulating any dynamic data to prepare it for dynamic visualizations in ArcGIS Explorer and Google Earth.

Command line syntax

PotentialPathTool <Select_Feature_Class> <Select_Animal_Identifier_Field_> <Select_TimeStamp_Field> {Option1__Select_Tortoise_Speed_Table_} {Option2__Input_Tortoise_Speed_} <Output_Workspace_> {Limit_Potential_Distance___Enter_limits_below_} <-------Hour_Limit_> <------Distance_Limit__meters__> {Output_as_WGS84__Not_UTM__} <Populate_Input_Feature_Class_with_Symbol_Angle__>

Parameters

Expression	Explanation
<Select_Feature_Class>	Select the feature class that contains tortoise positions. It can contain many tortoises or an individual tortoise.
<Select_Animal_Identifier_Field_>	Select the field that contains the Animal Identifier.
<Select_TimeStamp_Field>	Select the TimeStamp field. If it does not exist, the TimeStamp Creator tool needs to be run first.
{Option1__Select_Tortoise_Speed_Table_}	Option 1: • Select the tortoise speed table. This table would have been created by the Tortoise Maximum Speed Finder tool. If it has not been run, please run it first to use this option. That tool calculated the maximum achieved speed in meters/hour for each tortoise.
{Option2__Input_Tortoise_Speed_}	Option 2: • Enter the tortoise speed in meters/hour.
<Output_Workspace_>	Select the Geodatabase or Feature Dataset where you want the output table to be created.
{Limit_Potential_Distance___Enter_limits_below_}	When checked and the Hour Limit and Distance Limit populated, this will limit the sizes of the ellipses to a 1 Day potential distance IF the

	consecutive positions are less than the limit distance (meters) and the positions are greater than the Hour Limit (hours) apart.
<-------Hour_Limit_>	The minimum hour differential that consecutive positions must be to be candidates for limited ellipse calculations. • Default is 36 hours (1.5 days).
<------Distance_Limit__meters__>	The maximum distance differential that consecutive positions must be to be candidates for limited ellipse calculations. • Default is 21 meters.
{Output_as_WGS84__Not_UTM__}	Check this box if you want the output to be in WGS84 datum. Unchecked, the results will be in UTM 11N.
<Populate_Input_Feature_Class_with_Symbol_Angle__>	Checked- This will create a SymAngle field in your input feature class and populated it with the objects bearing to the next sequential position. The angle value in the field can then be used in the layer properties to automatically rotate the symbols to these values.

Command Line Example

Scripting syntax

PotentialPathTool (Select_Feature_Class, Select_Animal_Identifier_Field_, Select_TimeStamp_Field, Option1__Select_Tortoise_Speed_Table_, Option2__Input_Tortoise_Speed_, Output_Workspace_, Limit_Potential_Distance___Enter_limits_below_, -------Hour_Limit_, ------Distance_Limit__meters__, Output_as_WGS84__Not_UTM__, Populate_Input_Feature_Class_with_Symbol_Angle__)

Parameters

Expression	**Explanation**
Select Feature Class (Required)	Select the feature class that contains tortoise positions. It can contain many tortoises or an individual tortoise.
Select Animal Identifier Field: (Required)	Select the field that contains the Animal Identifier.
Select TimeStamp Field (Required)	Select the TimeStamp field. If it does not exist, the TimeStamp Creator tool needs to be run first.
Option1: Select Tortoise Speed Table: (Optional)	Option 1: • Select the tortoise speed table. This table would have been created by the Tortoise Maximum Speed Finder tool. If it has not been run, please run it first to use this option. That tool calculated the maximum

	achieved speed in meters/hour for each tortoise.
Option2: Input Tortoise Speed: (Optional)	Option 2: • Enter the tortoise speed in meters/hour.
Output Workspace: (Required)	Select the Geodatabase or Feature Dataset where you want the output table to be created.
Limit Potential Distance? (Enter limits below) (Optional)	When checked and the Hour Limit and Distance Limit populated, this will limit the sizes of the ellipses to a 1 Day potential distance IF the consecutive positions are less than the limit distance (meters) and the positions are greater than the Hour Limit (hours) apart.
-------Hour Limit: (Required)	The minimum hour differential that consecutive positions must be to be candidates for limited ellipse calculations. • Default is 36 hours (1.5 days).
------Distance Limit (meters): (Required)	The maximum distance differential that consecutive positions must be to be candidates for limited ellipse calculations. • Default is 21 meters.
Output as WGS84 (Not UTM)? (Optional)	Check this box if you want the output to be in WGS84 datum. Unchecked, the results will be in UTM 11N.
Populate Input Feature Class with Symbol Angle?: (Required)	Checked- This will create a SymAngle field in your input feature class and populated it with the objects bearing to the next sequential position. The angle value in the field can then be used in the layer properties to automatically rotate the symbols to these values.

Lost Tortoise Tool

This tool was built by David Turnbull, NGA, as part of his thesis at the University of Redlands (2010). This tool was designed for his client: Joshua Tree National Park, but can be used for manipulating any dynamic data to prepare it for dynamic visualizations in ArcGIS Explorer and Google Earth.

Command line syntax

losttortoise <Tortoise_Name_> {Option1__Select_Tortoise_Speed_Table_} {Option2__Estimated_Tortoise_Speed_in_meters_hr_} <Projection_> <Enter_the_last_known_UTM__X_Position__meters__> <Enter_the_last_known_UTM_Y_Position__meters__> <Enter_the_last_known_position_TimeStamp_> <Enter_the_current_TimeStamp_> {Clip_Search_Radius_by_a_Feature_Class_} {Erase_part_of_Search_Radius_by_a_Feature_Class_} {Workspace_}

Parameters

Expression	**Explanation**
<Tortoise_Name_>	Enter the Lost Tortoise's Name. If using the Tortoise Maximum Speed Tool, it must match the spelling and case listed in the tool.
{Option1__Select_Tortoise_Speed_Table_}	Option 1: Select location of Tortoise Maximum Speed table.
{Option2__Estimated_Tortoise_Speed_in_meters_hr_}	Option 2: Enter the tortoise's approximate speed in meters/hour. It is noted that the speeds of the tortoises are likely below 18 meters/hour.
<Projection_>	Please select the preferred projection for the output.
<Enter_the_last_known_UTM__X_Position__meters__>	Enter the UTM longitude in meters.
<Enter_the_last_known_UTM_Y_Position__meters__>	Enter the UTM latitude in meters.
<Enter_the_last_known_position_TimeStamp_>	Enter the last known position's TimeStamp.
<Enter_the_current_TimeStamp_>	Enter the current TimeStamp.

{Clip_Search_Radius_by_a_Feature_Class_}	Optional- select a feature class to clip the search area by.
{Erase_part_of_Search_Radius_by_a_Feature_Class_}	Optional- select a feature class to erase part of the search area.
<Workspace_>	Select the output workspace.

Scripting syntax

losttortoise (Tortoise_Name_, Option1__Select_Tortoise_Speed_Table_,
Option2__Estimated_Tortoise_Speed_in_meters_hr_, Projection_,
Enter_the_last_known_UTM__X_Position__meters__,
Enter_the_last_known_UTM_Y_Position__meters__, Enter_the_last_known_position_TimeStamp_,
Enter_the_current_TimeStamp_, Clip_Search_Radius_by_a_Feature_Class_,
Erase_part_of_Search_Radius_by_a_Feature_Class_, Workspace_)

Parameters

Expression	**Explanation**
Tortoise Name: (Required)	Enter the Lost Tortoise's Name. If using the Tortoise Maximum Speed Tool, it must match the spelling and case listed in the tool.
Option1: Select Tortoise Speed Table: (Optional)	Option 1: Select location of Tortoise Maximum Speed table.
Option2: Estimated Tortoise Speed in meters/hr: (Optional)	Option 2: Enter the tortoise's approximate speed in meters/hour. It is noted that the speeds of the tortoises are likely below 18 meters/hour.
Projection: (Required)	Please select the preferred projection for the output.
Enter the last known UTM X Position (meters): (Required)	Enter the UTM longitude in meters.
Enter the last known UTM Y Position (meters): (Required)	Enter the UTM latitude in meters.
Enter the last known position TimeStamp: (Required)	Enter the last known position's TimeStamp.
Enter the current TimeStamp: (Required)	Enter the current TimeStamp.
Clip Search Radius by a Feature Class? (Optional)	Optional- select a feature class to clip the search area by.
Erase part of Search Radius by a Feature Class? (Optional)	Optional- select a feature class to erase part of the search area.

Workspace: (Required)	Select the output workspace.

Priority Trot Needs Tool

This tool was built by David Turnbull, NGA, as part of his thesis at the University of Redlands (2010). This tool was designed for his client: Joshua Tree National Park.

Command line syntax

```
primarytrots222
<Input__All_Limited_Potential_Path_Areas_;Input__All_Limited_Potential_Path_Areas_...>
<Input_All_Unlimited_Potential_Path_Areas_;Input_All_Unlimited_Potential_Path_Areas_...>
<Select_the_Road_Buffer_Near_Feature_Class_> <Enter_Park_Boundary_Limit_Feature_Class_>
<Primary_Trot_Needs_Feature_Class> <Secondary_Trot_Needs_Feature_Class>
<Tertiary_Trot_Needs_Feature_Class>
```

Parameters

Expression	Explanation
<Input__All_Limited_Potential_Path_Areas_;Input__All_Limited_Potential_Path_Areas_...>	Select all Limited Potential Path polygons created by the Tortoise Potential Path Tool.
<Input_All_Unlimited_Potential_Path_Areas_;Input_All_Unlimited_Potential_Path_Areas_...>	Select all Unlimited Potential Path polygons created by the Tortoise Potential Path Tool.
<Select_the_Road_Buffer_Near_Feature_Class_>	Select the Road Buffer Near feature class created by the Road and Trot Buffer Class.
<Enter_Park_Boundary_Limit_Feature_Class_>	Select the Park Boundary feature class or other area feature class that you want to clip the data with.
<Primary_Trot_Needs_Feature_Class>	Specify the location and name for the new Primary Trot Needs Feature Class.
<Secondary_Trot_Needs_Feature_Class>	Specify the location and name for the new Secondary Trot Needs Feature Class.
<Tertiary_Trot_Needs_Feature_Class>	Specify the location and name for the new Tertiary Trot Needs Feature Class.

Scripting syntax

```
primarytrots222 (Input__All_Limited_Potential_Path_Areas_,
Input_All_Unlimited_Potential_Path_Areas_, Select_the_Road_Buffer_Near_Feature_Class_,
Enter_Park_Boundary_Limit_Feature_Class_, Primary_Trot_Needs_Feature_Class,
Secondary_Trot_Needs_Feature_Class, Tertiary_Trot_Needs_Feature_Class)
```

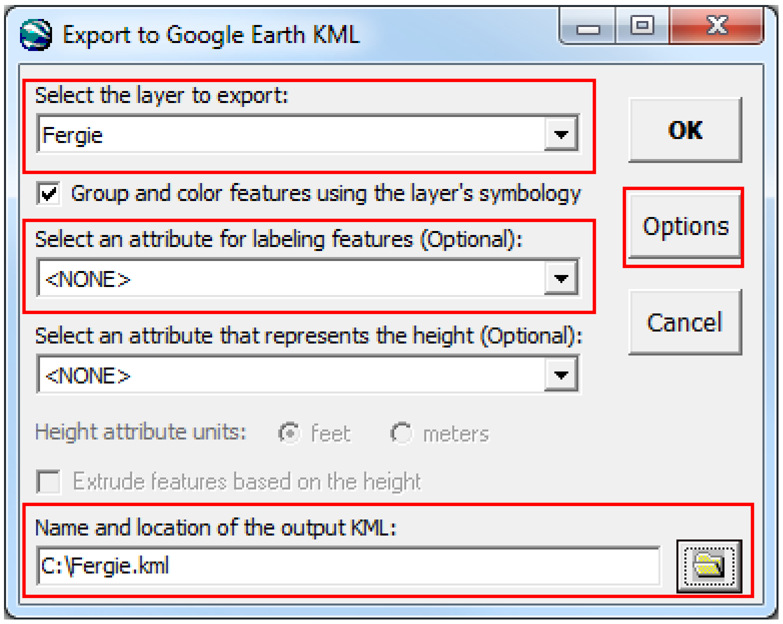
Parameters

Expression	Explanation
Input All Limited Potential Path Areas: (Required)	Select all Limited Potential Path polygons created by the Tortoise Potential Path Tool.
Input All Unlimited Potential Path Areas: (Required)	Select all Unlimited Potential Path polygons created by the Tortoise Potential Path Tool.
Select the Road Buffer Near Feature Class: (Required)	Select the Road Buffer Near feature class created by the Road and Trot Buffer Class.
Enter Park Boundary Limit Feature Class: (Required)	Select the Park Boundary feature class or other area feature class that you want to clip the data with.
Primary Trot Needs Feature Class (Required)	Specify the location and name for the new Primary Trot Needs Feature Class.
Secondary Trot Needs Feature Class (Required)	Specify the location and name for the new Secondary Trot Needs Feature Class.
Tertiary Trot Needs Feature Class (Required)	Specify the location and name for the new Tertiary Trot Needs Feature Class.

Appendix C. Export to KML and Google Earth Use

Export to KML 2.5.5 General Instructions

The Export to KML button will appear on the toolbar if properly installed and at least one feature class is in the Table of Contents in ArcMap. Clicking on it will invoke the Export to KML 2.5.5 tool. The key parameters are outlined in red in Figure C-1. First, you must select the layer to export. This layer must be in your Table of Contents in ArcMap. The second field allows you to choose a field for labeling the features. Once you have set these parameters as well as the output location, the Options Button must be clicked.


Export to Google Earth KML

Select the layer to export: Fergie

[✓] Group and color features using the layer's symbology

Select an attribute for labeling features (Optional): <NONE>

Select an attribute that represents the height (Optional): <NONE>

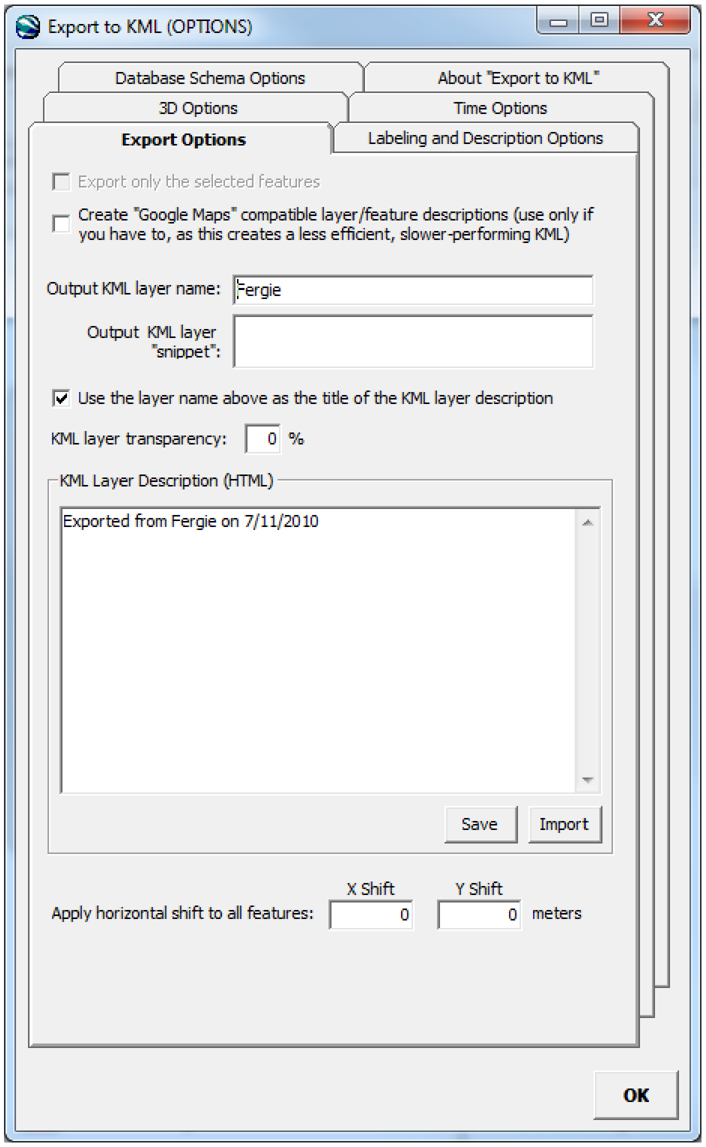
Height attribute units: [✓ feet] meters

[ ] Extrude features based on the height

Name and location of the output KML: C:\Fergie.kml

Figure C-1: Initial Export to KML Window and Parameters

The options window includes three important tabs that contain parameters modified for the production of the KMLs in this project. Those tabs include Export Options, Database Schema Options, and Time Options. The Export Options seen in Figure C-2, allows you to specify the output layer name that appears in the KML, as well as additional metadata. This tab also allows you to apply an X or Y shift in the data. This function was used to produce the images in this document to mask the true positions of the tortoises.



- [ ] Export only the selected features
- [ ] Create "Google Maps" compatible layer/feature descriptions (use only if you have to, as this creates a less efficient, slower-performing KML)

Output KML layer name: Fergie

Output KML layer "snippet":

- [x] Use the layer name above as the title of the KML layer description

KML layer transparency: 0 %

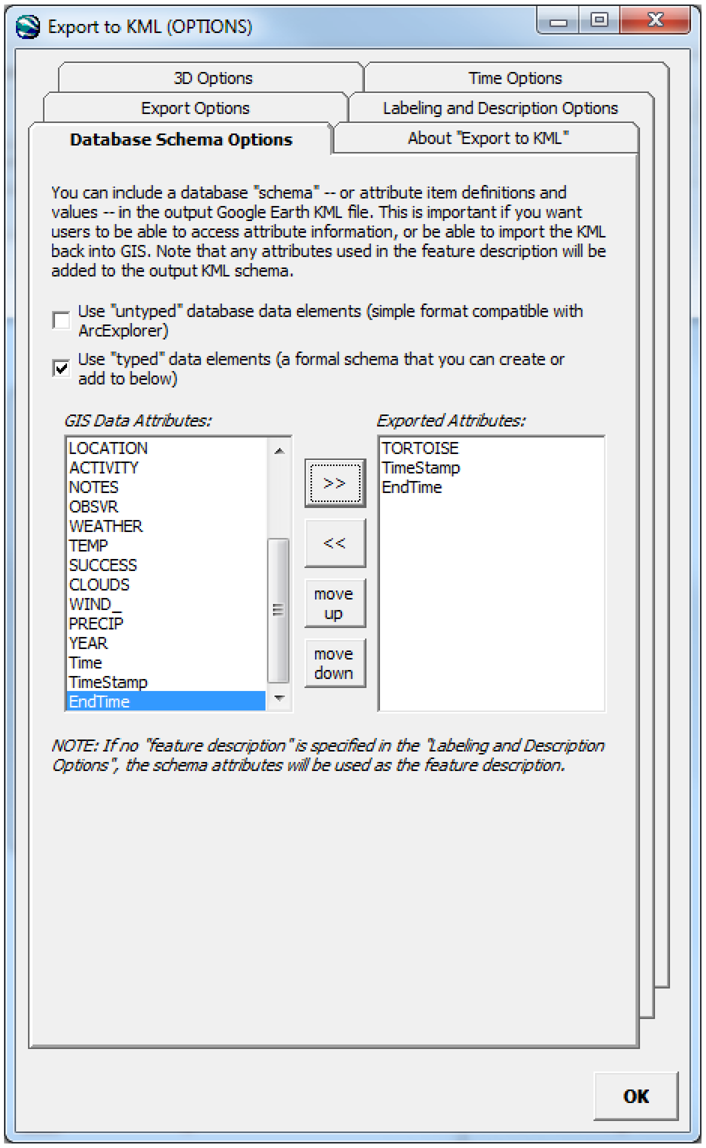
KML Layer Description (HTML)

Exported from Fergie on 7/11/2010

Apply horizontal shift to all features: X Shift 0 Y Shift 0 meters

Figure C-2: Export Options Tab and Parameters

The Database Schema Options tab seen in Figure C-3, allows the user to select fields to be mapped to the KML file. These attributes will be shown in pop-up balloons if the feature is clicked on in Google Earth.


Export to KML (OPTIONS)

You can include a database "schema" -- or attribute item definitions and values -- in the output Google Earth KML file. This is important if you want users to be able to access attribute information, or be able to import the KML back into GIS. Note that any attributes used in the feature description will be added to the output KML schema.

☐ Use "untyped" database data elements (simple format compatible with ArcExplorer)

☑ Use "typed" data elements (a formal schema that you can create or add to below)

GIS Data Attributes:

LOCATION
ACTIVITY
NOTES
OBSVR
WEATHER
TEMP
SUCCESS
CLOUDS
WIND_
PRECIP
YEAR
Time
TimeStamp
EndTime

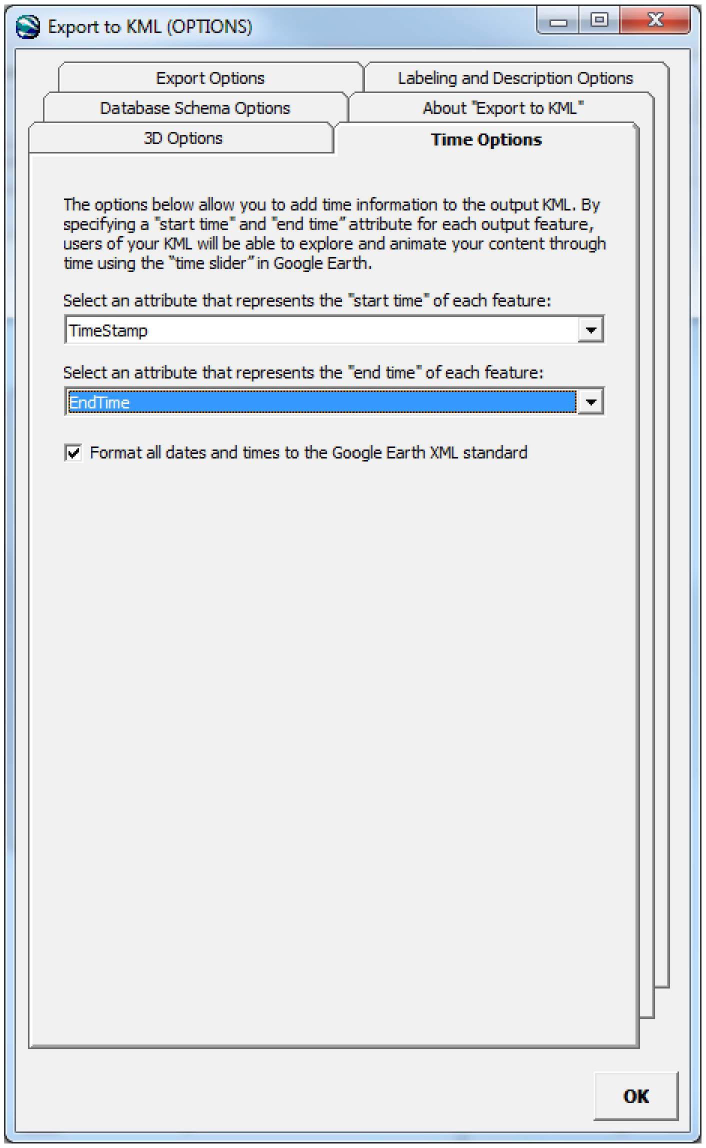
Exported Attributes:

TORTOISE
TimeStamp
EndTime

NOTE: If no "feature description" is specified in the "Labeling and Description Options", the schema attributes will be used as the feature description.

Figure C-3: Database Schema Options Tab and Parameters

The Time Options tab seen in Figure C-4 is where the user selects either the TimeStamp field for features that are meant to stay on once the date occurs, or both the TimeStamp and EndTime field for feature that are temporal.



The options below allow you to add time information to the output KML. By specifying a "start time" and "end time" attribute for each output feature, users of your KML will be able to explore and animate your content through time using the "time slider" in Google Earth.

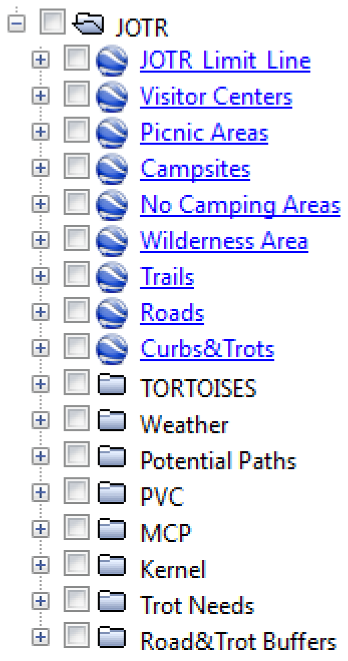
Select an attribute that represents the "start time" of each feature: TimeStamp

Select an attribute that represents the "end time" of each feature: EndTime

[✓] Format all dates and times to the Google Earth XML standard

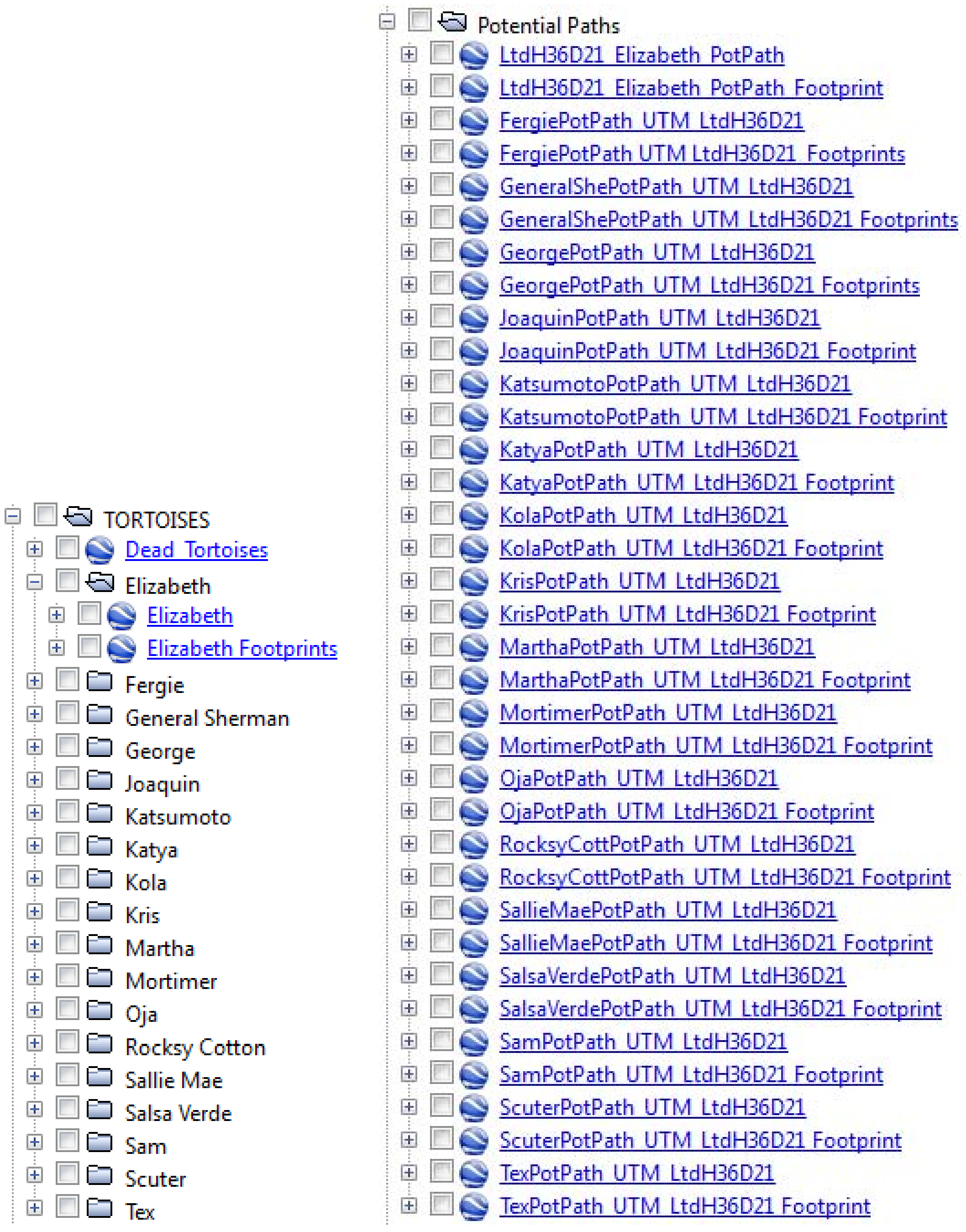
Figure C-4: Time Options Tab and Parameters

JOTR KMZ Folder Structure



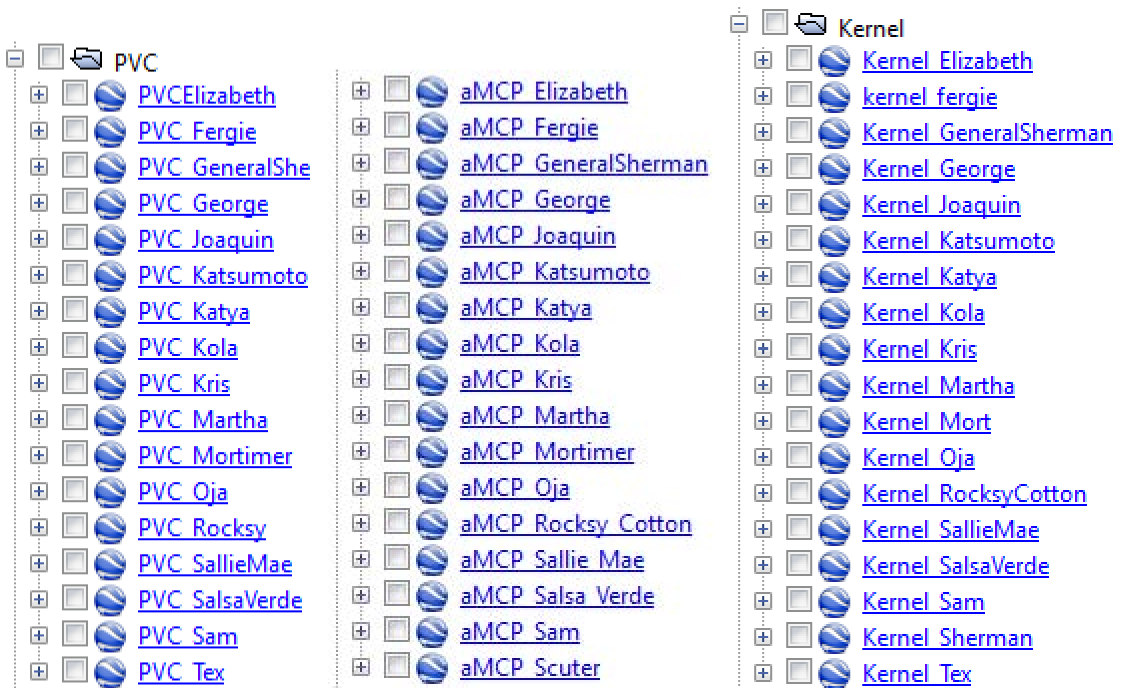
JOTR
- JOTR Limit Line
- Visitor Centers
- Picnic Areas
- Campsites
- No Camping Areas
- Wilderness Area
- Trails
- Roads
- Curbs&Trots
- TORTOISES
- Weather
- Potential Paths
- PVC
- MCP
- Kernel
- Trot Needs
- Road&Trot Buffers

Figure C-5: Collapsed JOTR KMZ Folders


TORTOISES
Dead_Tortoises
Elizabeth
Elizabeth
Elizabeth Footprints
Fergie
General Sherman
George
Joaquin
Katsumoto
Katya
Kola
Kris
Martha
Mortimer
Oja
Rocksy Cotton
Sallie Mae
Salsa Verde
Sam
Scuter
Tex

Potential Paths
LtdH36D21_Elizabeth_PotPath
LtdH36D21_Elizabeth_PotPath_Footprint
FergiePotPath_UTM_LtdH36D21
FergiePotPath UTM LtdH36D21_Footprints
GeneralShePotPath_UTM_LtdH36D21
GeneralShePotPath_UTM_LtdH36D21 Footprints
GeorgePotPath_UTM_LtdH36D21
GeorgePotPath_UTM_LtdH36D21 Footprints
JoaquinPotPath_UTM_LtdH36D21
JoaquinPotPath_UTM_LtdH36D21 Footprint
KatsumotoPotPath_UTM_LtdH36D21
KatsumotoPotPath_UTM_LtdH36D21 Footprint
KatyaPotPath_UTM_LtdH36D21
KatyaPotPath_UTM_LtdH36D21 Footprint
KolaPotPath_UTM_LtdH36D21
KolaPotPath_UTM_LtdH36D21 Footprint
KrisPotPath_UTM_LtdH36D21
KrisPotPath_UTM_LtdH36D21 Footprint
MarthaPotPath_UTM_LtdH36D21
MarthaPotPath_UTM_LtdH36D21 Footprint
MortimerPotPath_UTM_LtdH36D21
MortimerPotPath_UTM_LtdH36D21 Footprint
OjaPotPath_UTM_LtdH36D21
OjaPotPath_UTM_LtdH36D21 Footprint
RocksyCottPotPath_UTM_LtdH36D21
RocksyCottPotPath_UTM_LtdH36D21 Footprint
SallieMaePotPath_UTM_LtdH36D21
SallieMaePotPath_UTM_LtdH36D21 Footprint
SalsaVerdePotPath_UTM_LtdH36D21
SalsaVerdePotPath_UTM_LtdH36D21 Footprint
SamPotPath_UTM_LtdH36D21
SamPotPath_UTM_LtdH36D21 Footprint
ScuterPotPath_UTM_LtdH36D21
ScuterPotPath_UTM_LtdH36D21 Footprint
TexPotPath_UTM_LtdH36D21
TexPotPath_UTM_LtdH36D21 Footprint

Figure C-6: Expanded Tortoise and Potential Path Folders

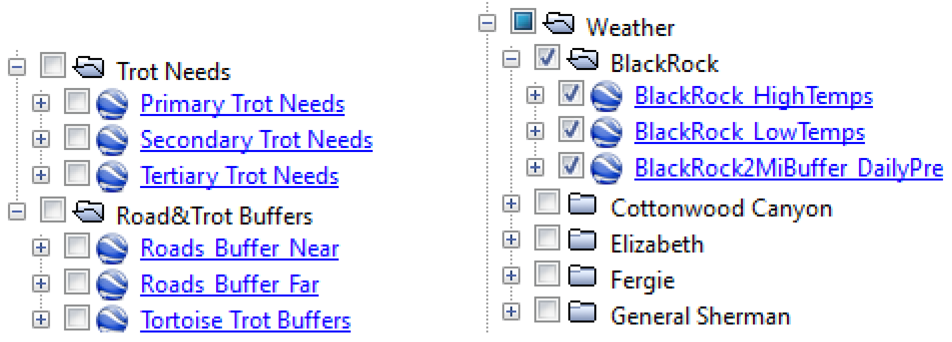


PVC
- PVCElizabeth
- PVC_Fergie
- PVC_GeneralShe
- PVC_George
- PVC_Joaquin
- PVC_Katsumoto
- PVC_Katya
- PVC_Kola
- PVC_Kris
- PVC_Martha
- PVC_Mortimer
- PVC_Oja
- PVC_Rocksy
- PVC_SallieMae
- PVC_SalsaVerde
- PVC_Sam
- PVC_Tex

- aMCP_Elizabeth
- aMCP_Fergie
- aMCP_GeneralSherman
- aMCP_George
- aMCP_Joaquin
- aMCP_Katsumoto
- aMCP_Katya
- aMCP_Kola
- aMCP_Kris
- aMCP_Martha
- aMCP_Mortimer
- aMCP_Oja
- aMCP_Rocksy_Cotton
- aMCP_Sallie_Mae
- aMCP_Salsa_Verde
- aMCP_Sam
- aMCP_Scuter

Kernel
- Kernel_Elizabeth
- kernel_fergie
- Kernel_GeneralSherman
- Kernel_George
- Kernel_Joaquin
- Kernel_Katsumoto
- Kernel_Katya
- Kernel_Kola
- Kernel_Kris
- Kernel_Martha
- Kernel_Mort
- Kernel_Oja
- Kernel_RocksyCotton
- Kernel_SallieMae
- Kernel_SalsaVerde
- Kernel_Sam
- Kernel_Sherman
- Kernel_Tex

Figure C-7: Expanded PVC, MCP, and Kernel Folder



Trot Needs
- Primary Trot Needs
- Secondary Trot Needs
- Tertiary Trot Needs

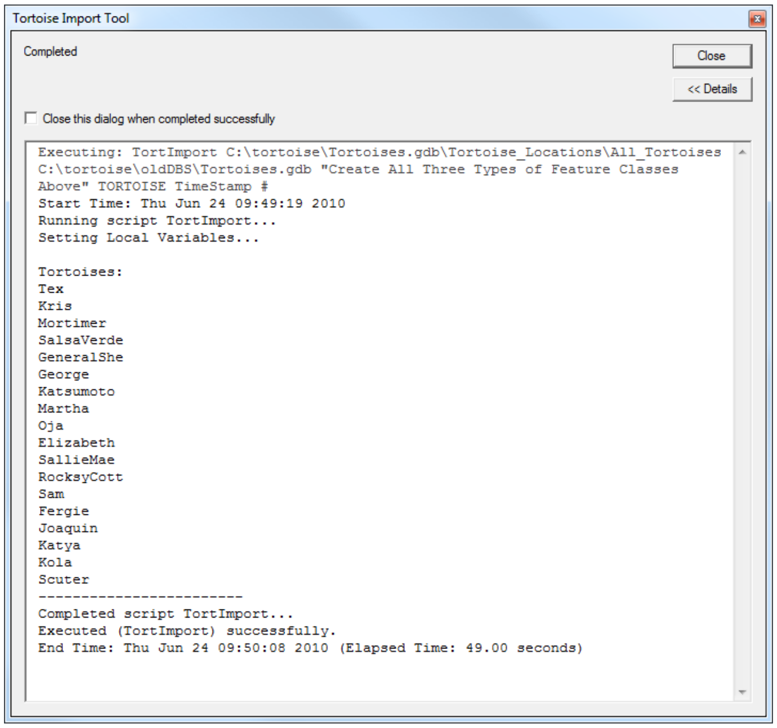
Road&Trot Buffers
- Roads_Buffer_Near
- Roads_Buffer_Far
- Tortoise Trot Buffers

Weather
- BlackRock
  - ✓ BlackRock_HighTemps
  - ✓ BlackRock_LowTemps
  - ✓ BlackRock2MiBuffer_DailyPre
- Cottonwood Canyon
- Elizabeth
- Fergie
- General Sherman

Figure C-8: Expanded Trot Needs, Roads and Trot Buffers, and Weather Folders

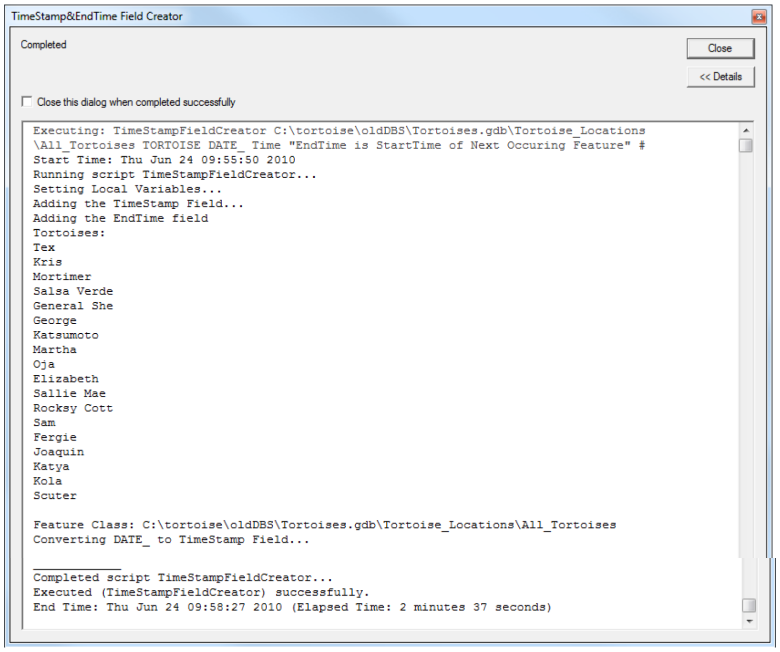
Appendix D. Output Windows

Tortoise Import Tool



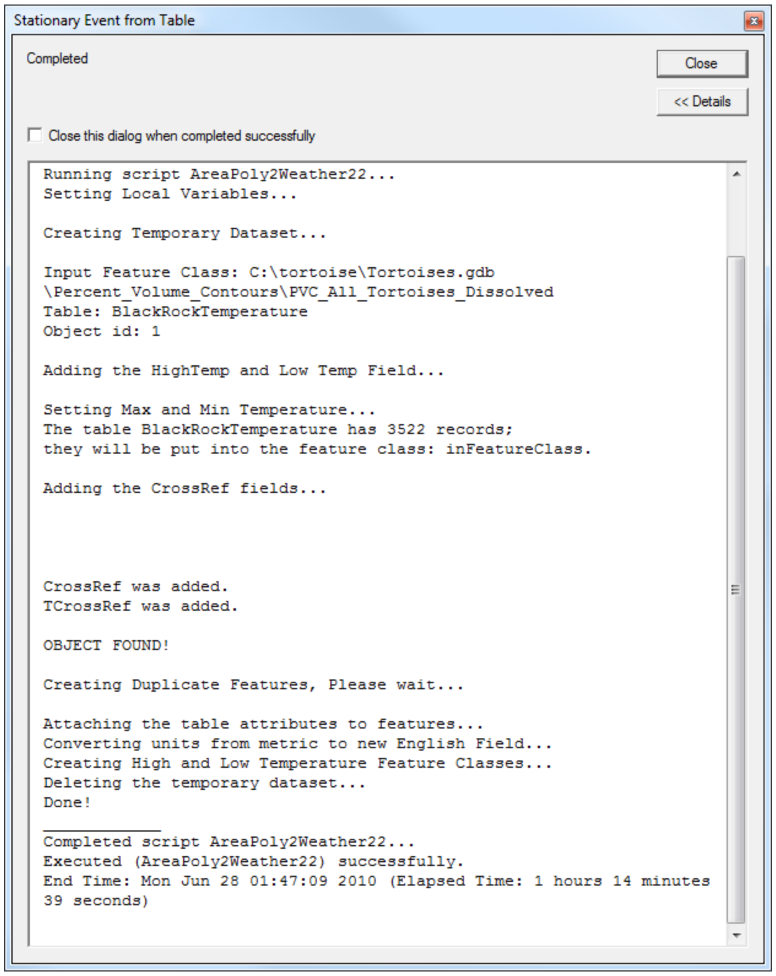
```
Executing: TortImport C:\tortoise\Tortoises.gdb\Tortoise_Locations\All_Tortoises
C:\tortoise\oldDBS\Tortoises.gdb "Create All Three Types of Feature Classes
Above" TORTOISE TimeStamp #
Start Time: Thu Jun 24 09:49:19 2010
Running script TortImport...
Setting Local Variables...

Tortoises:
Tex
Kris
Mortimer
SalsaVerde
GeneralShe
George
Katsumoto
Martha
Oja
Elizabeth
SallieMae
RocksyCott
Sam
Fergie
Joaquin
Katya
Kola
Scuter
------------------------
Completed script TortImport...
Executed (TortImport) successfully.
End Time: Thu Jun 24 09:50:08 2010 (Elapsed Time: 49.00 seconds)
```

TimeStamp and EndTime Field Creator Tool


TimeStamp&EndTime Field Creator
Completed
Close
<< Details
Close this dialog when completed successfully

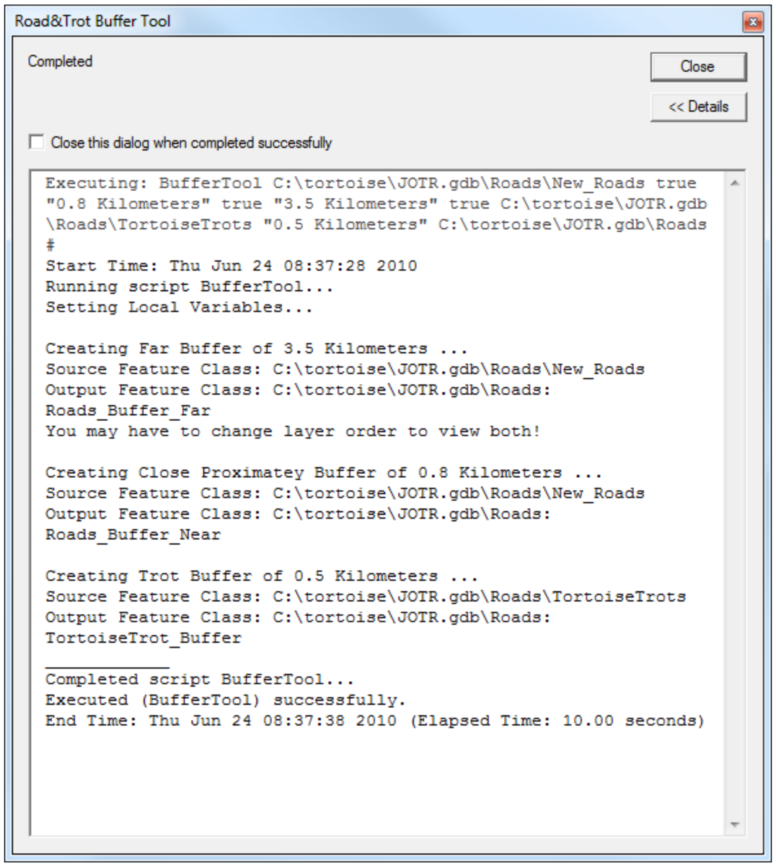
```
Executing: TimeStampFieldCreator C:\tortoise\oldDBS\Tortoises.gdb\Tortoise_Locations
\All_Tortoises TORTOISE DATE_ Time "EndTime is StartTime of Next Occuring Feature" #
Start Time: Thu Jun 24 09:55:50 2010
Running script TimeStampFieldCreator...
Setting Local Variables...
Adding the TimeStamp Field...
Adding the EndTime field
Tortoises:
Tex
Kris
Mortimer
Salsa Verde
General She
George
Katsumoto
Martha
Oja
Elizabeth
Sallie Mae
Rocksy Cott
Sam
Fergie
Joaquin
Katya
Kola
Scuter

Feature Class: C:\tortoise\oldDBS\Tortoises.gdb\Tortoise_Locations\All_Tortoises
Converting DATE_ to TimeStamp Field...

____________
Completed script TimeStampFieldCreator...
Executed (TimeStampFieldCreator) successfully.
End Time: Thu Jun 24 09:58:27 2010 (Elapsed Time: 2 minutes 37 seconds)
```

Stationary Event from Table Tool


Stationary Event from Table
Completed
Close
<< Details
Close this dialog when completed successfully

```
Running script AreaPoly2Weather22...
Setting Local Variables...

Creating Temporary Dataset...

Input Feature Class: C:\tortoise\Tortoises.gdb
\Percent_Volume_Contours\PVC_All_Tortoises_Dissolved
Table: BlackRockTemperature
Object id: 1

Adding the HighTemp and Low Temp Field...

Setting Max and Min Temperature...
The table BlackRockTemperature has 3522 records;
they will be put into the feature class: inFeatureClass.

Adding the CrossRef fields...




CrossRef was added.
TCrossRef was added.

OBJECT FOUND!

Creating Duplicate Features, Please wait...

Attaching the table attributes to features...
Converting units from metric to new English Field...
Creating High and Low Temperature Feature Classes...
Deleting the temporary dataset...
Done!
____________
Completed script AreaPoly2Weather22...
Executed (AreaPoly2Weather22) successfully.
End Time: Mon Jun 28 01:47:09 2010 (Elapsed Time: 1 hours 14 minutes
39 seconds)
```

Road and Trot Buffer Tool


Road&Trot Buffer Tool

Completed

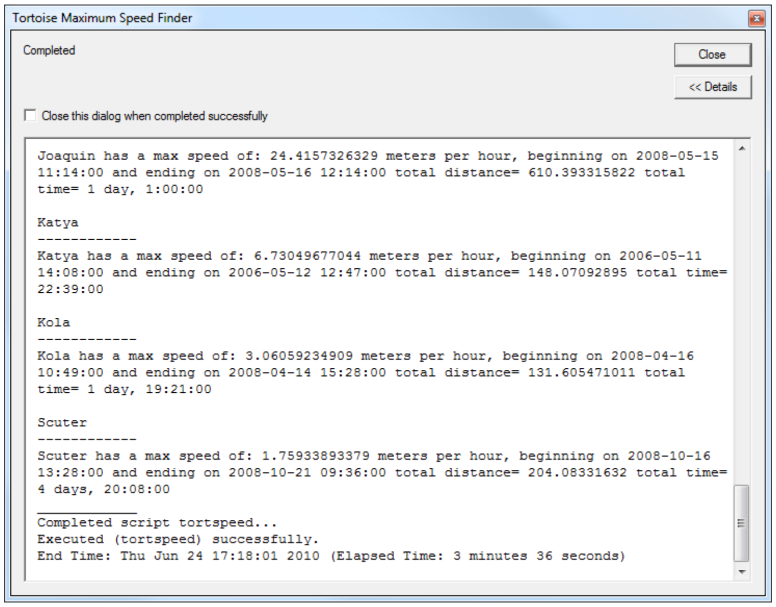
Close

<< Details

Close this dialog when completed successfully

```
Executing: BufferTool C:\tortoise\JOTR.gdb\Roads\New_Roads true
"0.8 Kilometers" true "3.5 Kilometers" true C:\tortoise\JOTR.gdb
\Roads\TortoiseTrots "0.5 Kilometers" C:\tortoise\JOTR.gdb\Roads
#
Start Time: Thu Jun 24 08:37:28 2010
Running script BufferTool...
Setting Local Variables...

Creating Far Buffer of 3.5 Kilometers ...
Source Feature Class: C:\tortoise\JOTR.gdb\Roads\New_Roads
Output Feature Class: C:\tortoise\JOTR.gdb\Roads:
Roads_Buffer_Far
You may have to change layer order to view both!

Creating Close Proximatey Buffer of 0.8 Kilometers ...
Source Feature Class: C:\tortoise\JOTR.gdb\Roads\New_Roads
Output Feature Class: C:\tortoise\JOTR.gdb\Roads:
Roads_Buffer_Near

Creating Trot Buffer of 0.5 Kilometers ...
Source Feature Class: C:\tortoise\JOTR.gdb\Roads\TortoiseTrots
Output Feature Class: C:\tortoise\JOTR.gdb\Roads:
TortoiseTrot_Buffer
____________
Completed script BufferTool...
Executed (BufferTool) successfully.
End Time: Thu Jun 24 08:37:38 2010 (Elapsed Time: 10.00 seconds)
```

Tortoise Maximum Speed Finder Tool


Tortoise Maximum Speed Finder

Completed

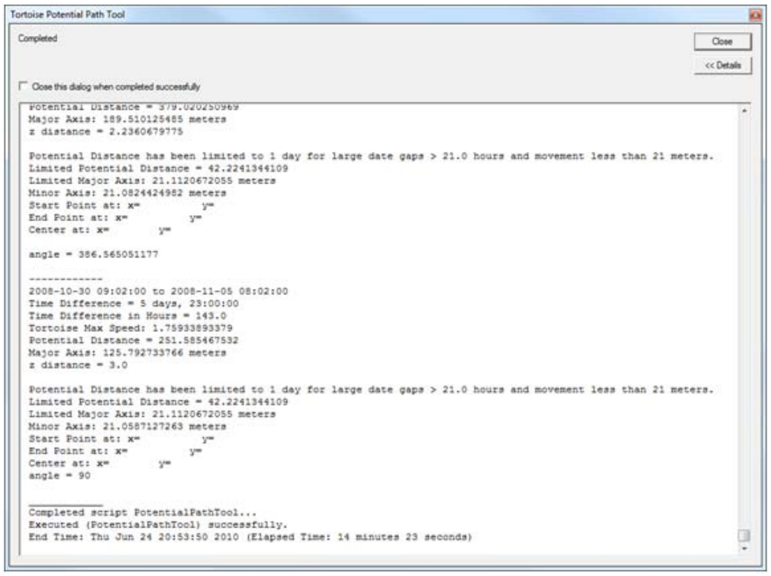
Close

<< Details

Close this dialog when completed successfully

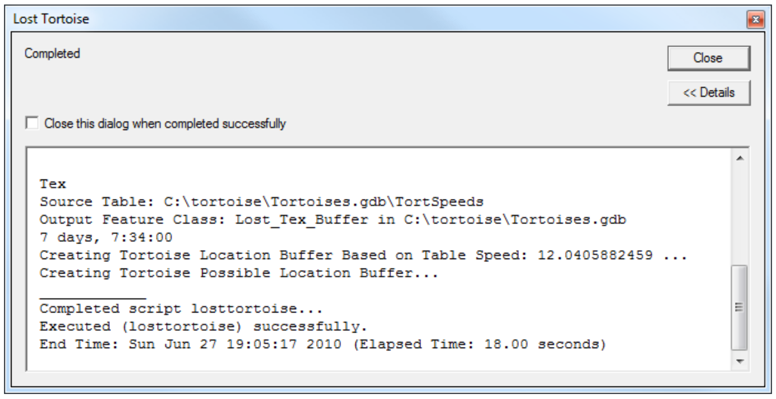
```
Joaquin has a max speed of: 24.4157326329 meters per hour, beginning on 2008-05-15
11:14:00 and ending on 2008-05-16 12:14:00 total distance= 610.393315822 total
time= 1 day, 1:00:00

Katya
------------
Katya has a max speed of: 6.73049677044 meters per hour, beginning on 2006-05-11
14:08:00 and ending on 2006-05-12 12:47:00 total distance= 148.07092895 total time=
22:39:00

Kola
------------
Kola has a max speed of: 3.06059234909 meters per hour, beginning on 2008-04-16
10:49:00 and ending on 2008-04-14 15:28:00 total distance= 131.605471011 total
time= 1 day, 19:21:00

Scuter
------------
Scuter has a max speed of: 1.75933893379 meters per hour, beginning on 2008-10-16
13:28:00 and ending on 2008-10-21 09:36:00 total distance= 204.08331632 total time=
4 days, 20:08:00
____________
Completed script tortspeed...
Executed (tortspeed) successfully.
End Time: Thu Jun 24 17:18:01 2010 (Elapsed Time: 3 minutes 36 seconds)
```

Tortoise Potential Path Tool


Tortoise Potential Path Tool
Completed
Close
<< Details
Close this dialog when completed successfully

```
Potential Distance = 379.020250969
Major Axis: 189.510125485 meters
z distance = 2.2360679775

Potential Distance has been limited to 1 day for large date gaps > 21.0 hours and movement less than 21 meters.
Limited Potential Distance = 42.2241344109
Limited Major Axis: 21.1120672055 meters
Minor Axis: 21.0824424982 meters
Start Point at: x=          y=
End Point at: x=          y=
Center at: x=        y=

angle = 386.565051177

------------
2008-10-30 09:02:00 to 2008-11-05 08:02:00
Time Difference = 5 days, 23:00:00
Time Difference in Hours = 143.0
Tortoise Max Speed: 1.75933893379
Potential Distance = 251.585467532
Major Axis: 125.792733766 meters
z distance = 3.0

Potential Distance has been limited to 1 day for large date gaps > 21.0 hours and movement less than 21 meters.
Limited Potential Distance = 42.2241344109
Limited Major Axis: 21.1120672055 meters
Minor Axis: 21.0587127263 meters
Start Point at: x=          y=
End Point at: x=          y=
Center at: x=        y=
angle = 90

____________
Completed script PotentialPathTool...
Executed (PotentialPathTool) successfully.
End Time: Thu Jun 24 20:53:50 2010 (Elapsed Time: 14 minutes 23 seconds)
```

Lost Tortoise Tool



```

Tex
Source Table: C:\tortoise\Tortoises.gdb\TortSpeeds
Output Feature Class: Lost_Tex_Buffer in C:\tortoise\Tortoises.gdb
7 days, 7:34:00
Creating Tortoise Location Buffer Based on Table Speed: 12.0405882459 ...
Creating Tortoise Possible Location Buffer...

____________
Completed script losttortoise...
Executed (losttortoise) successfully.
End Time: Sun Jun 27 19:05:17 2010 (Elapsed Time: 18.00 seconds)
```