MODELING WILDFIRE AND IGNITIONS FOR CLIMATE CHANGE AND ALTERNATIVE LAND MANAGEMENT SCENARIOS IN THE WILLAMETTE VALLEY, OREGON

by

TIMOTHY J. SHEEHAN

A THESIS

Presented to the Department of Biology
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Master of Science

December 2011

THESIS APPROVAL PAGE

Student: Timothy J. Sheehan

Title: Modeling Wildfire and Ignitions for Climate Change and Alternative Land Management Scenarios in the Willamette Valley, Oregon

This thesis has been accepted and approved in partial fulfillment of the requirements for the Master of Science degree in the Department of Biology by:

Dr. Bart Johnson	Co-Chair
Dr. Scott Bridgham	Co-Chair
Dr. John Bolte	Member

and

Kimberly Andrews Espy	Vice President for Research & Innovation/Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded December 2011

THESIS ABSTRACT

Timothy Sheehan

Master of Science

Department of Biology

December 2011

Title: Modeling Wildfire and Ignitions for Climate Change and Alternative Land Management Scenarios in the Willamette Valley, Oregon

I developed software to incorporate the FlamMap fire model into an agent-based model, Envision, to enable the exploration of relationships between wildfire, land use, climate change, and vegetation dynamics in the Willamette Valley.

A dynamic-link library plug-in utilizing row-ordered compressed array lookup tables converts parameters between polygon-based Envision data and grid-based FlamMap data. Modeled fires are determined through Monte-Carlo draws against a set of possible fires by linking historic fire data to future climate projections.

I used classification and regression tree (CART) and logistic regression to relate ignitions to human and land use factors in the Willamette Valley above the valley floor from 2000-2009. Both methods showed decreasing distance to major and minor roads as key factors that increase ignition probability for human ignitions but not for lightning ignitions. The resulting statistical model is implemented in the FlamMap plug-in to provide a dynamic ignition probability map over time.

CURRICULUM VITAE

NAME OF AUTHOR: Timothy J. Sheehan

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene
University of Tennessee, Knoxville
University of Colorado, Boulder
Front Range Community College, Westminster, CO
University of Missouri, Columbia
University of Missouri, St. Louis
St. Louis University, St. Louis, MO

DEGREES AWARDED:

Master of Science, Biology, 2011, University of Oregon
Master of Science, Computer Science, 1993, University of Colorado
Master of Science, Geology, 1989, University of Missouri
Bachelor of Science, Geology, 1982, University of Missouri

AREAS OF SPECIAL INTEREST:

Ecological modeling
High performance computational modeling
Large dataset analysis and management

PROFESSIONAL EXPERIENCE:

Ecological modeler, Conservation Biology Institute, Corvallis, Oregon, 2008-present

Instructor, Lane Community College, Eugene, Oregon, 2008-2009

Software Engineer, Brut ECN and Nasdaq Stock Market, Eugene, Oregon, 2001-2007

Research Assistant, Department of Computer Science, University of Oregon, Eugene, Oregon, 1998-2000

Parallel Applications Programmer/Director of Special Projects, Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, Tennessee, 1993-1998

Applications Programmer, National Center for Atmospheric Research, Boulder, Colorado, 1992-1993

PUBLICATIONS:

Sheehan, T., Malony, A., & Shende, S. (1999). A runtime monitoring framework for the TAU profiling system. *Proceedings of the Third International Symposium on Computing in Object-Oriented Parallel Environments (ISCOPE'99),* San Francisco, CA, December 1999.

Sheehan, T., Shelton W., Pratt, T., Papadopoulos, P., LoCascio P., & Dunigan, T. (1998). The locally self consistent multiple scattering method in a geographically distributed linked MPP environment. *Parallel Computing*, *24*(12-13).

Sheehan, T., Pennington, R., Papadopoulos, P., Geist, G., & Alexander, R. (1997). The seamless computing environment. *Intel Supercomputer Users Group Thirteenth Annual Conference Proceedings*.

Shelton, B., Sheehan, T., Papadopoulos, P., Mackay, D., LoCascio, P., & Pratt, T. (1997). Linked supercomputing between ORNL and SNL. *HPCU News 1*(1).

LoCascio, P. F., Bland, A. S., Dunigan, T. H., & Sheehan, T. J. (1997). A review of three generations of memory architecture design emphasizing shared memory systems. Whitepaper submitted to the Department of Energy office of Mathematical, Information and Computational Science Division.

Sheehan, T. J. (1993). *Porting the NCAR CCM2 from the Cray Y-MP to the Connection Machine.* (Unpublished masters thesis). University of Colorado-Boulder.

Sheehan, T. J. (1989). *Petrogenesis of Selected Migmatites from the Vermillion Granitic Complex of Northeast Minnesota*. (Unpublished masters thesis). University of Missouri-Columbia.

ACKNOWLEDGEMENTS

I thank my advisors, Professors Bart Johnson and Scott Bridgham for their guidance and help during the completion of my degree and the preparation of this thesis. Thanks to Professor John Bolte and Alan Ager for their help with the FlamMap Interpreter development. Thanks to the faculty and staff of the Department of Biology for intellectual and administrative support. A special thanks to Kim Sheehan for financial, emotional, and intellectual support.

This research was partially funded by the National Science Foundation Dynamics of Coupled Natural and Human Systems program, Grant No. 0816475, and by an award from the USDA Forest Service, Grant PNW 09-JV-11261900-004.

For Kim, who not only tolerates my many endeavors, but encourages them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Wildfire regimes, that is the frequency, intensity, size, and timing of wildfires, are heavily influenced by climate and are expected to be altered by global climate change. Changes in growing season onset and duration, precipitation patterns, and temperatures will affect many drivers of wildfire, including fuel loads, fuel condition, humidity, temperature, and wind. In the United States' Pacific Northwest, for example, future climate projections indicate a 0.1 to 0.6 °C per decade increase in annual temperature; warmer, wetter winters; longer growing seasons; and hotter summers with longer periods of drought (Mote and Salathé, 2010; Karl et al., 2009; Millar et al. 2007; Pachauri and Reisinger, 2007; Rapp, 2004). The resulting increases in annual fuel production and fuel combustibility are projected to lead to an increase in wildfire (Karl et al., 2009; Millar et al., 2007; Rapp, 2004).

Calculating the costs of wildfire and its management is complex and must take into consideration suppression, human infrastructure losses, and the loss of ecosystem products and services. In the United States alone, annual wildfire suppression costs can exceed one billion dollars (Busby and Albers, 2010). Annual home losses can range into the thousands, and forest product losses are substantial (Kauffman, 1990). Fire suppression and human-caused ignitions can both lead to undesirable ecological changes (Snider et al., 2006; Keeley and Fotheringham, 2003; Keeley and Keeley, 2000; Kauffman, 1990). Wildfire management seeks to mitigate these impacts, and wildfire simulation modeling has become a key element of wildfire management planning.

My research is part of a collaborative interdisciplinary project entitled "The Interactions of Climate Change, Land-Management Policies, and Forest Succession on Fire Hazard and Ecosystem Trajectories in the Wildland-Urban Interface," and is subsequently referred to by the acronym ICLF. ICLF is being conducted by researchers at the University of Oregon, Oregon State University and the USDA Forest Service, and is funded by the National Science Foundation's Dynamics of Coupled Natural and Human Systems Program. The project integrates computer simulation models of climate change, vegetation dynamics, wildfire, human population growth, and landowner decision making within an alternative futures scenarios framework (Hulse et al., 2009) that focuses on testing how different sets of land use and land management policies could affect landscape trajectories. The model as a whole is considered a "coupled systems" model in that it simulates the interactions and feedbacks among these different biophysical and human factors. The specifics of my research include the software engineering involved in integrating an existing spatially explicit fire model into the coupled systems model, and an analysis of the distribution of ignitions in the Willamette Valley based on human factors, land use / land cover, vegetation, and lightning occurrence.

The remainder of this chapter provides context and background for my thesis. Topics covered include climate modeling, the ICLF project, fire models, and ignitions modeling.

Climate Change Modeling

In recent decades, global climate change has moved from the realm of theoretical possibility to that of an observed phenomenon (Pachauri and Reisinger, 2007). Global mean temperature rose over 0.5 ˚C between 1970 and 2004 (Hansen et al., 2006). Over

the same period, some regions have experienced an increase of up to 3.5˚C, and projected global temperature increases for the last decade of the twenty-first century range from 1.1 to 6.4˚C compared to 1980-1999 (Pachauri and Reisinger, 2007).

Projecting the consequences of rising global temperatures and other climatic changes is an important and complex scientific challenge. The relationships among climatic, environmental, and socioeconomic factors are manifold and complex (Figure 1) and a combination of techniques are used to study them including historical data analysis (e.g., Mann et al., 1998), empirical analyses (e.g., Zemp et al., 2006; Menge and Field, 2007) and computational modeling.

The interrelated causes and effects associated with climate change can be separated into three categories: climatic factors, ecological and environmental factors, and human activities (Figure 1). Different types of computational models are best suited for each of these categories. Models that account for physical processes over the entire planet or that are downscaled to regional climate are used for simulating the spatial and temporal trends and variability in future climate. Dynamic global vegetation models (DGVMs) are used to model the effects of biogeochemical and climate change on vegetation and are commonly used on global to regional scales. While their use in climate modeling is relatively new (e.g., Natarajan et Al., 2011), agent-based models are used to model the effects of human decisions on the environment.

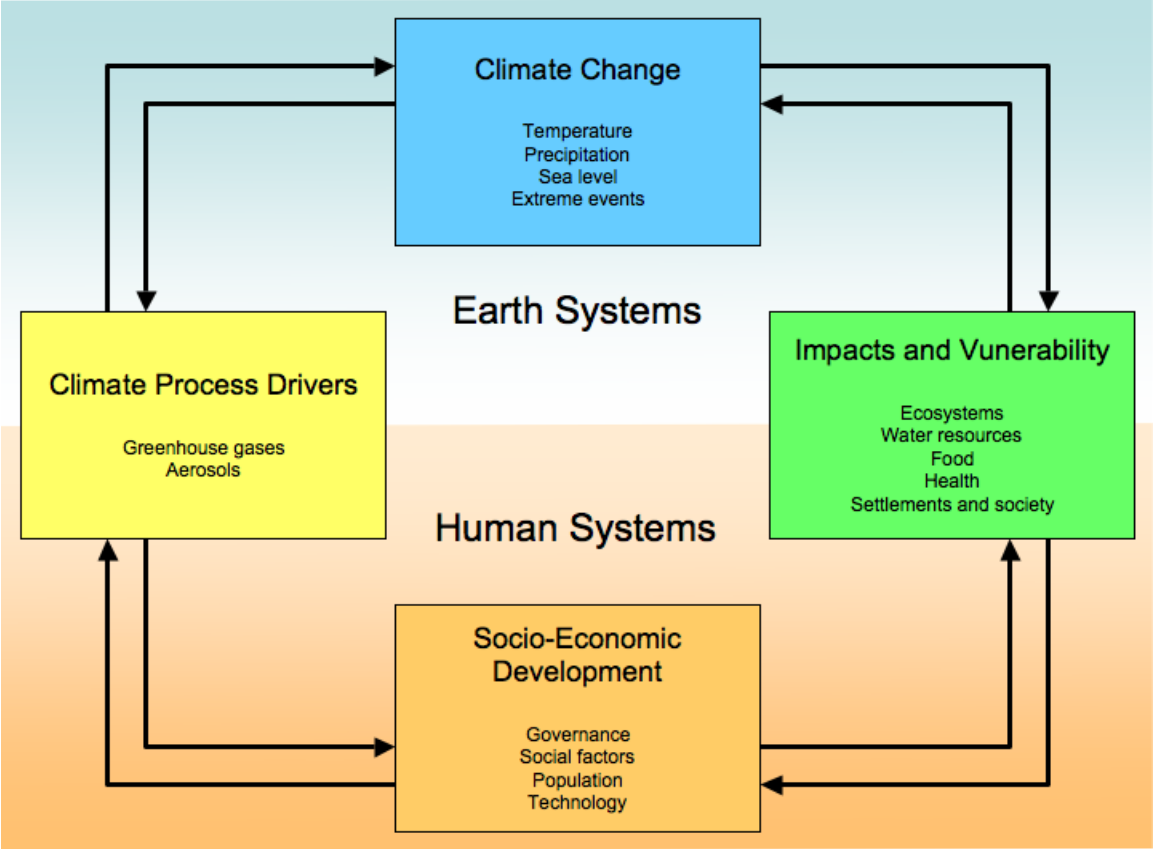

Figure 1. Climate drivers and impacts. Schematic framework representing anthropogenic drivers, impacts of, and responses to climate change and their linkages (after Pachauri and Reisinger, 2007).

For climate projections on the global scale, a variety of model types are used. These include Atmosphere-Ocean General Circulation Models (AOGCMs), Earth System Models of Intermediate Complexity (EMICs), and Simple Climate Models (SCMs) (Pachauri and Reisinger, 2007). As a check for consistency and range of possible outcomes, results from these models are commonly compared (Pachauri and Reisinger, 2007).

Such models are driven or "forced" by conditions such as changing atmospheric composition. To cover the uncertainty associated with those conditions, a range of projected scenarios are used. Perhaps the most widely used scenarios are those outlined in the *IPCC Special Report on Emissions Scenarios* (SRES) (Nakićenović and Swart, 2000)

based on four different "storylines" of projected future economic and human development. Scenarios range from those resulting in high greenhouse gas production to those under which greenhouse gas production is reduced.

Dynamic Global Vegetation Models (DGVMs) have proven useful for projecting the ecological effects of climate change (e.g., Lenihan et al., 2008; Sato et al., 2010; Zaehle et al., 2007). These models couple the biogeochemistry of carbon, nutrients, water, and energy to simulate changes in vegetation type (Nielson and Running, 1996). Processes modeled in DGVMs include plant physiology, ecosystem function, and vegetation dynamics, but they do not include individual-based tree dynamics. Commonly, estimates are generated for short- and long-term changes in net primary productivity, resource competition among plant functional types, mortality, and disturbance effects. External forcings often include soil characteristics and climate (Bachelet and Price, 2008).

DGVMs generally do not account for the effects of human land use and land management. Agent-based models (ABMs) are used to study socio-ecological processes (Matthews et al., 2007; e.g., Hulse et al., 2009; Topping et al., 2003). In these computer simulation models, a system comprises a collection of autonomous agents, or decision-making entities (Bonabeau, 2002). These models are useful for modeling systems in which individual behavior cannot be clearly defined, individual behavior is complex, model validation and calibration utilize expert judgment, or behavior has an element of stochasticity (Bonabeau, 2002). Human decision modeling is most often based on empirical qualitative and/or quantitative data derived from expert knowledge, surveys, interviews, and participant observation (Matthews et al., 2007).

The ICLF Project

The ICLF is a modeling project investigating interactions and feedbacks among climate change, vegetation succession, wildfire, and urbanization in Oregon's Willamette Valley ecoregion. A specific focus of the project is the reduction of catastrophic wildfire risk through oak savanna and prairie habitat restoration. The project uses an alternative futures scenario framework based on a two-by-two-by-two matrix of contrasting factors. The first pair of factors are the scenarios of higher versus lower effects of climate change on vegetation succession and wildfire. The second pair comprises compact versus dispersed land development to accommodate the projected 74% increase in population during the first half of the 21st century (Hulse et al., 2002). The final pair of factors contrasts two approaches to catastrophic wildfire risk management. The conventional fuel treatments policy set emphasizes fire suppression and the protection of individual homes. The mixed fuels-biodiversity policy set emphasizes landscape-scale restoration of fire-adapted oak ecosystems as a means to reduce wildfire severity and associated risks to people and residences when wildfire occurs.

The study area chosen for the project is an approximately 1000-km^2 exurban landscape located in the valley foothills to the south and east of the Eugene-Springfield metropolitan area (population ~250,000). This area includes a mosaic of farmland, forestland and rural residential housing, as well as a number of small cities and towns.

The coupled systems model integrates four modeling subsystems: climate, vegetation succession, human decision-making, and wildfire (Figure 2). The model as a whole and each submodel will be run on an annual time step for 50-100 years.

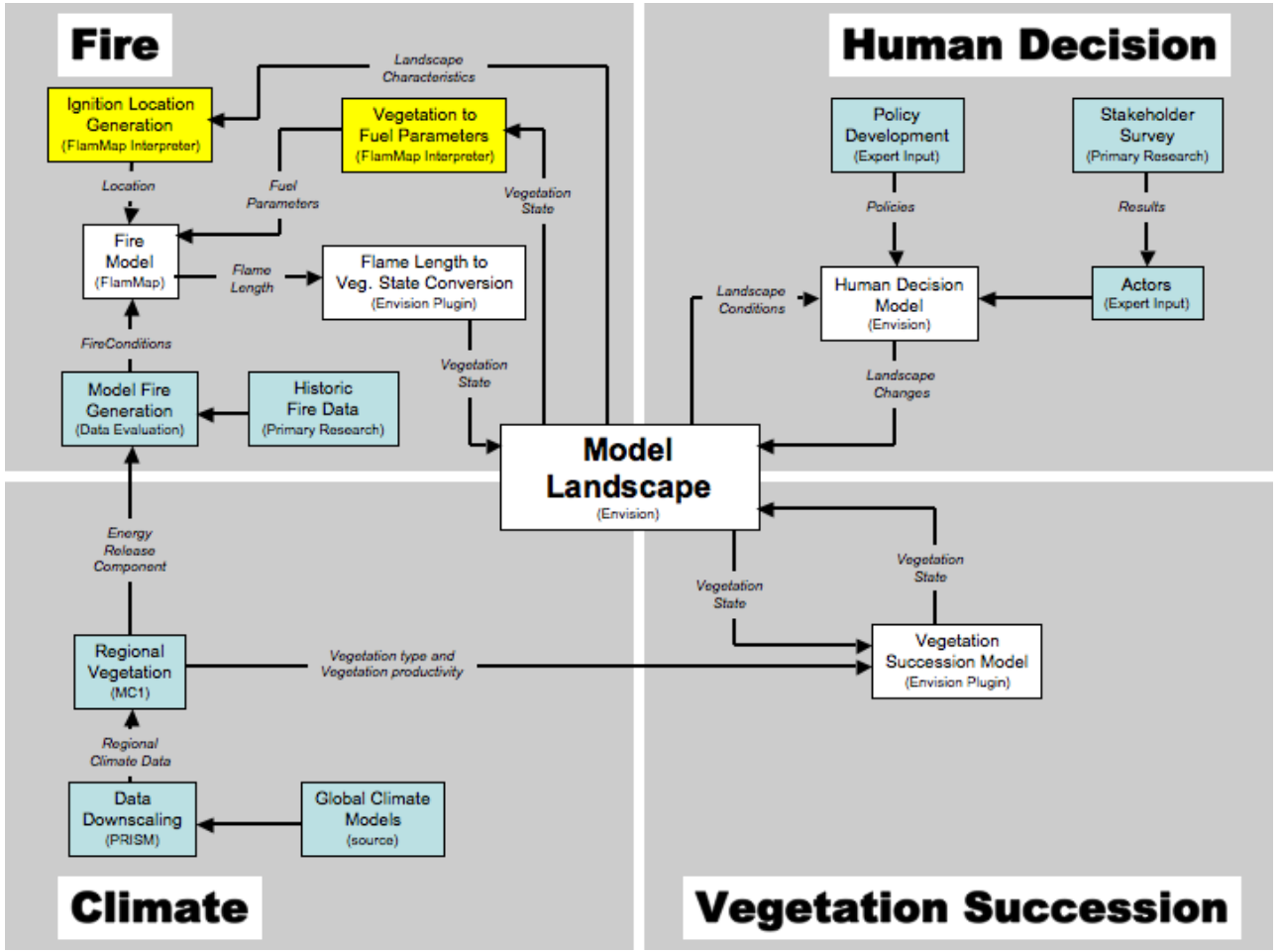

Figure 2. The ICLF coupled model. Schematic representation of the ICLF coupled model components. Components show in yellow were developed as part of this thesis. Components shown in blue are computed prior to the running of the coupled model.

Climate inputs to the model come from runs of two contrasting climate models using the IPCC A2 scenario, which projects increasing global population growth, with relatively slow and fragmented economic growth and technological change (Nakićenović and Swart, 2000). This data is downscaled using PRISM (parameter-elevation regressions on independent slopes model) (Daly et al., 2002) and used as input to the MC1 DVGM (Bachelet et al., 2001). Outputs from MC1 include calculated energy release component (an indicator of total possible heat energy release from a fire) data used to generate model fires, as well as vegetation types and vegetation productivity used by the state and transition model. The data provided by MC1 is computed prior to running the coupled model in Envision.

Vegetation modeling is performed using a state and transition vegetation succession model. In the state and transition model each landscape unit (described below) is assigned one of a suite of defined vegetation states based on its species composition and vegetation structure (e.g., tree canopy cover, and dominant species). Landscape units change states via transitions. Transitions in the ICLF project were based on the projected outcomes of tree growth and competition over time (for example a young conifer forest may transition to a mature conifer forest after a certain number of model years), or disturbance (for example a young mixed oak-conifer forest may transition to a young oak savanna after a moderate severity wildfire). Transitions were based on probabilities calculated from over 3,000 regional forest plots using the Forest Vegetation Simulator (FVS) (Crookston and Dixon, 2005) with modifications for Oregon white oak growth and mortality developed by members of our research team (Gould et al. 2011). Because FVS is an individual-based statistical forest growth and succession model parameterized

using current and historical growth conditions, MC1 was used to modify transitions based on its projections of future site productivity and potential vegetation in our study area (Yospin et al. in preparation).

The Envision agent-based modeling program (Bolte, 2011; Bolte et al., 2006 and Bolte, 2004 describe Envision's predecessor, Evoland; software download available at http://envision.bioe.orst.edu/) provides both the agent-based human decision component and the spatially explicit GIS-based framework of the coupled systems model. The landscape representation within the computer model (model landscape) is divided into polygons referred to as integrated decision units (IDUs). For the ICLF project, IDUs were created in GIS by intersecting the boundaries of taxlot parcels, the fundamental units of land use and management decisions in the U.S., with soils polygons, used to represent basic units influencing tree growth and vegetation dynamics. Each IDU is controlled by an agent or actor that makes decisions reflecting their values or mandates. Actors make decisions based on the state of their IDU and on policies – fundamental descriptors of criteria and actions. The results of actors' decisions are reflected in changes to the landscape. In the ICLF project, actors' propensities for decisions are based on the results of landowner surveys, and policies are being created by experts in land use and land management under the guidance of a stakeholder advisory group.

The work covered by this thesis lies at the interface of the models described above. Modeled fire probabilities, fire weather, and fire duration are based on data that links analyses of historic fire occurrences to potential fire intensity outputs produced by MC1. Each vegetation state from the state and transition model has been assigned fuels characteristics that are converted into landscape parameters used by the fire model, and

results from the fire model are used to update vegetation states. The results of the ignitions research in this thesis are incorporated into the fire model integration software.

Fire Modeling

Fire models have been developed for many different purposes and on many spatial and temporal scales. A fire model may be implemented as a standalone application, as portable software module, or as an integral part of a model with a fire component. Fire models typically compute area burned, burn severity expressed in terms of factors such as intensity, flame length, and temperature. They may also provide data on spatio-temporal factors such as spread rate and path. Within fire models, ignitions may be specified by input or generated deterministically or stochastically by the fire model itself. Burn extent calculations, in terms of both spatial extent and burn severity, may be based on the simple, static conditions in a single model cell, to the complex interplay of multiple processes both spatially and temporally.

Choosing a fire model for the ICLF project involved weighing the benefits of various fire models against several factors. It had to be computationally efficient enough to run repeatedly in on the order of a minute per time step of the coupled systems model. It needed to allow the input of ignition locations so they could be based on a probability distribution determined from an analysis of recent fires. The fire model also needed to operate at a spatial grain similar to the ~0.1 to 5 ha spatial grain of IDUs. Finally, it needed to be implemented as a plug-in module for Envision. The remainder of this section surveys a variety of computer models that perform fire modeling and provides an overview of FlamMap, the model chosen by the ICLF primary investigators for the project. FlamMap is described in greater detail in Chapter 2.

Fire Models

Fire is one of the disturbances often modeled within dynamic global vegetation models (DGVMs). Fire modeling in DGVMs is commonly driven by conditions computed within the model with deterministic ignitions occurring without regard to ignition cause. Burning is commonly modeled on an intra-cellular level with no fire spread between cells. MC1, for example, bases fire occurrence on an assumed ignition when a threshold based on fuel condition, fine-fuel flammability, and fire spread is reached (Bachelet et al., 2001). Surface and crown fire are both simulated, and fire effects include crown kill, root kill, and fuel consumption. While MC1 was used in the coupled systems model to produce vegetation states and energy release components, its fire modeling capabilities were not used because the project used the more mechanistic FlamMap model.

The Glob-FIRM fire model (Thonicke et al., 2001) was developed as a module for use with LPJ-DGVM (Sitch et al., 2000) and is also used in the individual-based SEIB-DGVM (Sato et al., 2007). Plant functional type (the classification of vegetation by characteristics such as phylogeny, morphology, size, leaf characteristics, leaf seasonality, etc.), plant functional type fire resistance, and fire season length are the main drivers of fire effects in Glob-FIRM. Ignitions are deterministic, based on a fire probability computed from observed relationships between daily litter moisture and fire season length. Like MC1, Glob-FIRM does not take into account land-use or human-altered fire regimes.

Crevoisier et al. (2007) developed PBA (for Potential Burned Area), a boreal forest fire model created for use in DGVMs and evaluated it over a coarse spatial and temporal scale (2° x 2.5°, 1 month). PBA includes the human-related factor of road

density, in addition to the more typical factors of precipitation, temperature, soil water content, and relative humidity.

LANDIS (Forest Landscape Disturbance and Succession) (Mladenoff, 2004) is a spatial landscape simulation model that allows for more detailed modeling of forests than most DGVMs. LANDIS models cells as forest stands based on species-specific composition and maturity with associated biophysical characteristics. In the LANDIS fire module (He et al., 2004), fire occurrence is stochastic with probabilities based on the fire return interval for the landscape and the time since last fire. Burn intensity is based on the combination of coarse and fine fuels. While fire is not modeled based on physical factors (e.g., detailed fuel properties, air temperature, wind speed), it can spread from cell to cell. LANDIS-II (Scheller et al., 2007), the successor to LANDIS, includes an extensible, modular software architecture, as well as options for variable fire return intervals and fire size distribution.

Fire behavior models, commonly used in wildland fire management, are more mechanistic than those previously discussed, and take into account fuels, weather, and terrain. Outputs from these models may include fuel and fire characteristics such as fuel moisture, flame lengths, rate of spread, area burned, and fire type (surface, passive crown fire, active crown fire).

Fire behavior models are designed with varying levels of complexity. BEHAVE Plus (Andrews, 2007) is a relatively simple model that utilizes uniform conditions temporally and spatially on a landscape having a constant slope. FlamMap (Finney, 2006; Firemodels.org, 2010) (discussed in more detail below), on the other hand, holds conditions constant temporally, and varies conditions spatially. FARSITE (Fire Area

Simulator) (Finney, 1998) varies conditions in time diurnally and by day as well as spatially. FSPro (http://www.wy.blm.gov/fireuse/2008workshop/presentations/12fspro4rx.pdf) produces probabilities of fire spread by performing thousands of model runs with differing weather scenarios.

The FlamMap Fire Behavior Model in the ICLF Project

The FlamMap fire behavior model provides the functionality and computational performance needed in the ICLF project. In addition, it is available as dynamic link library (DLL) usable with the Windows operating system. This allows it to be adapted into an Envision plug-in DLL module. As used in the project, inputs include fuel moistures, fire weather conditions, burn period, ignition location, and landscape parameters consisting of slope, aspect, elevation, fuel model (a specification of fuel characteristics), extent of canopy cover, canopy base height, canopy height, and crown bulk density.

Input files for fuel moistures, fire weather, and burn period are read at the start of each multiyear Envision model run. Ignition locations are generated by the FlamMap Interpreter for each individual model fire via a Monte Carlo method with a probability surface generated from the ignitions model from this thesis. FlamMap reads the landscape parameters from a landscape (LCP) file. These parameters come from a combination of input file data and data generated by the FlamMap interpreter.

FlamMap and its use in the ICLF project are described in greater detail in Chapter 2 and in Appendices A and B.

Factors for Ignition Modeling on the Landscape

Modeling the spatial distribution of ignition probability is a critical component of wildfire simulation models and an important part of the ICLF fire-modeling component. Accurately modeling burn areas, which depends in part on ignition locations, is important for several aspects of the ICLF model: the fine-scale modeling of vegetation, actor behavior in response to nearby fires, and fire locations themselves. Several key factors need to be considered when modeling wildfire ignitions: study area size and spatial resolution, data source and choice, statistical techniques, ignition source, and analytical factors.

Study Area Size and Spatial Resolution

A study area may range from all land on the globe to a region on the order of hundreds of km^2 with the analytic unit size generally, but not always, roughly scaling with the area covered. For instance, Krawchuk et al. (2009) did an analysis of the entire globe at a resolution of 100 km. Parisien and Moritz (2009) used data on a 1-km grid for analyses of the United States, California, and several subregions of California. Pu et al. (2007) generated and analyzed a forest fire dataset for the United States at a 1-km resolution. Cardille et al. (2001) analyzed a 280,000-km^2 area in the upper Midwest with dimensions of 10-km and 5-km grids. In an analysis covering the state of California, Syphard et al. (2007) summarized data on a county-by-county basis. On a 1,287-km^2 area of the Missouri Ozark Highlands, Yang et al. (2007) analyzed human-caused ignitions on an individual basis with a 30-m resolution for landscape characteristics. In a study of the Santa Monica Mountains National Recreation Area (approximately 600 km^2), Syphard et al. (2008) treated ignitions as point data, using a 10-m resolution for distance variables,

and a 30-m resolution for biophysical variables with the exception of 1-km resolution for temperature.

Data Source and Choice

Sources used for ignition and fire data include fire databases (e.g., Parisien and Moritz, 2009; Syphard et al., 2007; Cardille et al. 2001), remote sensing data (e.g., Pu et al., 2007), aerial photos (e.g., Vazquez and Moreno, 2001), and field research (e.g., Vazquez and Moreno, 2001).

Which wildfire data are used in a study can depend on data availability as well as the aspect of ignition or fire being studied. For instance, for their study of lands suitable for wildfire in the conterminous United States, Parisien and Moritz (2009) only used data from federal agencies for fires on federal lands. In addition, they limited the data to fires larger than 121 ha to be consistent with the data they obtained for California wildfires. Studies concerned with "wildfires" as opposed to "ignitions" commonly disregard fires smaller than a certain size (e.g., Cardille et al., 2001), and in some cases evaluate fires of different sizes separately (e.g., Brosofske et al., 2007; Cardille et al. 2001). In contrast, studies specifically concerned with ignitions generally do not filter events by the size of the burn (e.g., Reineking et al., 2010; Syphard et al., 2008; Romero-Calcerrada et al., 2008; Yang et al. 2007).

Data may be partitioned based on ignition source, commonly human-caused versus lightning-caused fires. For example, Yang et al. (2007) and Zhang et al. (2010) considered only human-caused ignitions in their studies in the Missouri Ozarks and the Inner Mongolia grasslands, respectively. Rorig and Ferguson (1999) examined factors affecting lightning-caused ignitions in the northwestern United States, as did Castedo-

Dorado et al. (2011) in the Leon province of northwestern Spain. Reineking et al. (2010) analyzed human- and lightning-caused fires separately in order to understand differences based on ignition source.

Statistical Techniques

A wide variety of techniques have been used to analyze and model ignition and wildfire data (Seidl et al., 2011 summarized many of these). These include logistic regression (Peng et al., 2002), classification and regression tree (CART) (Breiman et al., 1984), geographically weighted regression (GWR) (Brunsdon et al., 1996), weight of evidence (WofE) (Albert, 1990), multivariate adaptive regression splines (MARS) (Friedman, 1991), neural networks (Vega-Garcia and Chuvieco, 2006), and fuzzy logic (Zedah, 1983).

Logistic regression is a well-known statistical method for generating a continuous probability function based on data with binary outcomes and has been used in many ignition and wildfire studies (e.g., Badia et al., 2011; Syphard et al., 2008; Brosofke et al. 2007; Diaz-Avalos et al., 2001).

CART analysis is a nonparametric technique in which data is recursively partitioned into a tree structure through the use of rules that divide the data into cohesive groups. CART produces discrete categories of data, and maps produced from results display sharp boundaries between areas represented by the categorical data (e.g., Amatulli et al., 2006). This method has been successfully used by Brosofke et al. (2007) and Amatulli et al. (2006).

With GWR (geographically weighted regression), the formula for linear regression is extended to allow local variation in the rate of change factor. Using a GWR,

Koutsias et al. (2010) were able to increase the power of their explanatory model to 78% as compared to 52% with ordinary least squares regression. One limitation of GWR is that it does not permit discrete binary explanatory variables (Amatulli and Camia, 2007).

WofE (weight of evidence) uses multiple independent variables and conditional probabilities to determine the effect of independent variables on the dependent variable (Romero-Calcerrada et al., 2008). Romero-Calcerrada et al. (2008) and Romero-Calcerrada et al. (2010) have used this method to study human-caused ignitions in an area of Central Spain. Dilts et al. (2009) also used this method for fire occurrence in Lincoln County, Nevada.

MARS (multivariate adaptive regression splines) is a nonparametric regression method that, unlike recursive partitioning (e.g., CART), produces continuous models with continuous derivatives (Friedman, 1991). Amatulli and Camia (2007) applied both CART and MARS methods on data from the Arno River Basin in Italy. While they found CART more accurate, MARS produced smoothed, more homogenous values.

Artificial neural networks process data by connecting data elements in layers. "Learning" is achieved by modifying the weights of connections between elements (Vega-Garcia and Chuvieco, 2006). Vega-Garcia and Chuvieco (2006) used neural networks to study the relationship between landscape heterogeneity and wildfire occurrence in a region in Eastern Spain. In a study of impacts of population density and weather on wildfire risk in Japan, Li et al. (2009) found results from neural networks better captured the nonlinear relationships in the data than did polynomial regression.

Fuzzy logic is based on the concept of a continuum between true and false instead of the traditional binary view. This concept has led to analytical techniques that can be

applied in many fields. Loboda and Csiszar (2007) utilized a fuzzy logic model to create a fire threat model in the Russian Far East. Nadeau and Edgefield (2006) used fuzzy logic to produce a map of fuel types in an area of Alberta, Canada. The use of fuzzy logic allows for a degree of uncertainty in the encoding of explanatory variables used in the analyses.

Ignition Source

The relative proportions of human- versus lightning-caused wildfires vary regionally. A number of areas have few lightning-caused fires (<= 5%) such as the United States upper Midwest (Cardille et al. 2001), Missouri's Ozark Highland (Yang et al., 2007), California's Santa Monica Mountains (Syphard et al., 2008), and the State of California as a whole (Syphard et al., 2008). In contrast, other areas have a substantial percentage of lightning-caused fires, such as Bages County, Catalonia, Spain (58%) (Badia et al., 2011), the Inner Mongolia Autonomous Region in northeast China (58%) (Zhang et al., 2010), the Northwest United States (Rorig and Ferguson, 1999), and Canada (35%) (Weber and Stocks, 1998).

Many studies focus only on human-caused fires and ignitions. This is common where human causes account for the vast majority of fires (e.g., Syphard et al., 2008; Yang et al., 2007). In other studies, lightning-caused fires are not considered separately because of the dominance of human-caused fires (e.g., Cardille et al. 2001). The relative proportions of human- and lightning-caused fires are not the only reason a researcher may concentrate on human-caused fires. For example, Pew and Larsen (2001) limited their study of wildfires in the temperate rainforest of Vancouver Island, Canada to human-caused fires due to the potential to control ignitions caused by human activity.

However, the causes and effects of lightning and human caused ignitions can differ substantially and may justify their separate consideration, even in areas where human fires dominate. Reineking et al. (2010) examined human- and lightning-caused fires separately and found that explanatory factors differed significantly. They also found that the suitability of certain fire indices differed for human- and lightning-caused fires. The effects of fires from these two classes of fires can also differ significantly. In Canada, lightning accounted for only 35% of wildfires but was responsible for 85% of the area burned (Weber and Stocks, 1998).

The factors affecting the spatial distribution of lightning-caused ignitions may be complex. For instance, human activities would not be expected to influence lightning-caused ignitions (Reineking et. al., 2010), yet in a study within the Canadian western boreal forest, road network density correlated with increased lightning-caused fire (Arienti et al., 2009). The correlation was attributed to vegetation changes along roads. Furthermore, lightning-caused ignition occurrence may vary on different landscapes. In Nevada, Dilts et al. (2009) found lightning ignitions more common in mountains than in basins. Castedo-Dorada et al. (2011), however, found lightning ignitions more common at lower elevations, and attributed this to higher rainfall and lower temperatures at higher elevations.

Analytical Factors

A variety of biophysical and human factors are commonly considered in modeling ignitions and fire distribution (see for example: Zhang et al., 2010; Dilts et al., 2009; Syphard et al., 2008; Yang et al., 2007; and Dickson et al., 2006). Biophysical factors commonly include slope, aspect and vegetation type; climatic factors such as rainfall,

temperature, humidity, and lightning frequency; and roughness of terrain. Human factors include distance to roads and road density, distance to railroad, distance to development and development density, presence within and distance to the wildland urban interface, human population density, and land use/land cover.

Thesis Objectives

The general architecture of the ICLF coupled systems model required linking models and data from differing temporal and spatial domains. Careful consideration of global climate scenarios, regional climate projections, fire models, vegetation models, and human models has led to the specific architecture of the system being implemented. Likewise data issues and analytical techniques for ignitions modeling were carefully considered for the ignitions modeling portion of the ICLF project.

This thesis deals specifically with two aspects of the fire-modeling portion of the ICLF. The first is the FlamMap Interpreter, software I developed that provides the linkage between the agent-based Envision human decision model and the FlamMap spatially explicit fire model. This software allows FlamMap to be run as an Envision plug-in module, providing the necessary data manipulation, file generation, and software execution. Chapter 2 contains more detailed descriptions of Envision and FlamMap as well as a description of the FlamMap Interpreter. *The Envision FlamMap Interpreter User's Guide* and *The Envision FlamMap Interpreter Programmer's Guide*, are contained in appendices A and B respectively.

The second aspect of my thesis research was the development of an ignition probability model for a portion of the Willamette Valley, which has been implemented within the FlamMap Interpreter. Chapter 3 presents the ignitions probability research.

Topics covered include the development of the dataset, descriptive statistical analysis, CART and logistic regression modeling of the data, as well as the results and interpretation of the model.

CHAPTER II

THE ENVISION – FLAMMAP INTERFACE

Introduction

Landscape characteristics and wildfire are intimately related. Given an ignition, vegetation characteristics influence whether a wildfire will occur, and if so, how it will burn. Wildfire, in turn, can influence what vegetation grows, and how it grows. When humans are included in the mix – causing ignitions, suppressing wildfire, altering vegetation, and shaping the landscape – these interrelationships become much more complex.

The relationships among vegetation, human activity, and wildfire are central to the interdisciplinary project entitled "The Interactions of Climate Change, Land-Management Policies, and Forest Succession on Fire Hazard and Ecosystem Trajectories in the Wildland-Urban Interface," and subsequently referred to by the acronym ICLF. The agent-based human decision model Envision (Bolte, 2011; Bolte et al., 2006 and Bolte, 2004 describe Envision's predecessor, Evoland) serves as the unifying software for the ICLF's coupled systems model. Some components of the model, notably future climate projection, are utilized pre-runtime to generate model input. Others are implemented as software plug-in modules and exhibit feedback behavior with various model components, including a vegetation state and transition model and a fire model. At runtime, these plug-in modules perform data processing and exchange data with Envision.

The spatially explicit FlamMap fire model (Finney, 2006) was chosen for the ICLF project because of its computational efficiency, its ability to model wildfire at a

spatial grain and extent commensurate with that used in the rest of the project, and its availability as a Windows dynamic link library (DLL). At a high level, the desired capabilities of the fire model were to take landscape data from Envision and climate data from an external source and use these to model fires on the model study area landscape each annual model time step. This provided several challenges for the use of any fire model in general and specifically for the FlamMap DLL including:

- *Burn parameters*: climate-driven probabilistic ignition occurrence, fire weather, and fire duration all based on future climate projections.
- *Ignition placement*: the placement of ignitions on the landscape based on a spatially explicit probability distribution function as opposed to deterministically or completely randomly.
- *Landscape characteristic conversion*: the conversion and exchange of landscape characteristics between those used by Envision and those used by FlamMap.
- *Spatial domain translation*: the conversion of data between the polygon-based Envision and grid-based FlamMap spatial domains.

This chapter describes the Envision software program, the FlamMap software program, pre-runtime protocols used to generate fire weather parameters and burn probabilities, and the FlamMap Interpreter, which utilizes data from pre-runtime protocols and implements ignition placement, data conversion between Envision and FlamMap, and the running of model fires (fires within the modeling software, i.e., FlamMap).

Envision

Envision is a geographic information systems (GIS) based alternative futures modeling program (Bolte, 2011; Bolte et al., 2006 and Bolte, 2004 describe Envision's predecessor, Evoland; software download available at http://envision.bioe.orst.edu/) implemented using the C++ programming language (Stroustrup, 2000). Envision includes components that allow modeling multiple processes on a landscape and evaluating the results of their interactions and feedbacks. These components include a spatially explicit landscape, actors (also called agents), policies, autonomous processes, and landscape evaluators.

A landscape is composed of polygons termed *integrated decision units* (IDUs). IDUs are defined by the user based on criteria useful to the modeling goals. In the ICLF project they are defined by the intersection of taxlot parcels and soil polygons, and range from <0.1 ha to 5 ha.

Actors represent groups or individuals and make management decisions about individual IDUs. These decisions reflect actors' values and mandates in light of the characteristics of the IDUs they control and are reflected by changes to those IDUs. Two surveys of study area landowners (Nielsen-Pincus et al. 2010) were used to parameterize actor characteristics in the ICLF project.

Polices guide and constrain actors' decisions about IDUs. In Envision, policies are the fundamental descriptors of actions that may be taken by actors, including the criteria that determine whether or not a policy is adopted by a given actor in a given IDU in a given model time step. Policies may represent the choices of individuals based purely on their values, as well as regulations and incentives implemented by public agencies to guide land-use decisions. In the ICLF project, contrasting pairs of policy sets are use to

compare the potential outcomes of compact versus dispersed development, and a conventional versus mixed fuels-biodiversity approach to wildfire management.

In addition to the spatially defined landscape and IDUs, actors, and policies, Envision can accommodate additional components that influence model outcomes. Autonomous processes can be created to affect the landscape without actor input. Wildfire is an example of an autonomous process. Furthermore, landscape evaluators can be defined to produce landscape metrics, for instance the scarcity or abundance of certain resources, that change as a result of actor decisions and autonomous processes. These landscape metrics can, in turn, influence actor behaviors.

In Envision, real world variation in decision-making is reflected by the probabilistic responses of actors to policies and landscape conditions. For example, an actor strongly inclined towards economic development might be 70% likely to harvest timber if prices rise to a certain level, while an actor strongly inclined towards ecological restoration might be only 30% likely. Envision normally is run on the order of tens or hundreds of times for a given set of policies, actors, and initial landscape conditions, so that trends in the output from multiple runs can examined and compared.

Envision provides an interface for the incorporation of externally developed DLL plug-ins for evaluative models and autonomous processes. This interface allowed for the development of the FlamMap Interpreter as a DLL-based Envision plug-in.

FlamMap and the FlamMap DLL

FlamMap was designed to model fire behavior under constant fuel moistures, wind speed, and wind direction over a spatially explicit model landscape (Finney, 2006). Landscape inputs include parameters for topography, vegetation structure, and fuel

loading. Inputs also include ignition points, lines, or polygons. FlamMap uses these parameters along with fire weather parameters (wind speed and wind direction) to compute fire behavior, intensity, and spread within cells on a regularly spaced grid. An option also exists to perform inter-cell fire spread computations. Inputs and outputs are described in greater detail in the remainder of this section.

Fuel moistures provide FlamMap with fuel condition data for different types of vegetation and are provided to FlamMap via a fuel moistures input file. Each row of the fuel moistures file consists of six columns. The first column is a numeric identifier for a fire behavior fuel model, a mathematical characterization of live and dead ground layer fuels that control surface fire behavior (Scott and Burgan, 2005; Rothermal, 1972). The remaining five columns are moisture percentage values for 1-hr (hour), 10-hr, and 100-hr dead fuels, live herbaceous, and live woody fuels, respectively. One-hr, 10-hr, and 100-hr are classes of fuels based on the time needed for fuels of different sizes to reach approximately 63% of the difference between its moisture content and the equilibrium moisture content reflecting ambient weather conditions. FlamMap allows for the optional use of a fuel-conditioning algorithm. This option computes 1- and 10-hr fuel moistures based on an input weather stream before modeling a fire.

FlamMap uses a regularly spaced, two-dimensional grid to represent the landscape. Each grid cell is assigned parameters that define the fuel and landscape characteristics used in computing fire behavior and spread. Eight parameters, or layers are required. The first three of these, slope, aspect, and elevation, define the landscape topography. The fourth, fuel model value, is used to index into the fuel models input data described above. The next three, canopy cover (the horizontal percent coverage of forest

canopy), canopy base height, and canopy height, define the extent of the forest canopy. The final layer, crown bulk density, is a measure of the canopy in mass per unit volume.

Landscape parameters are input via a binary landscape (LCP) file (format description available at http://www.gdal.org/frmt_lcp.html) consisting of a header that contains information about the landscape and its descriptive data. Included in the header are the units used for each parameter, grid cell size and units, and the spatial extent of the model landscape. The body of the LCP file contains the parameters for each cell of the model landscape. Other FlamMap inputs include wind speed and direction which contribute to fire spread calculations and are specified interactively.

FlamMap can model burns as either static, in which each cell burns independently without influence or spread from other cells, or as fires which spread from ignition locations. The latter case, used for the ICLF project, requires the option for the minimum travel time (MTT) algorithm (Finney, 2002). This algorithm provides efficient computation of fire spread on the model landscape. FlamMap uses the MTT algorithm with a user-specified maximum simulation time parameter to determine the model fire perimeter. The MTT algorithm requires the additional input of one or more ignition locations. Ignition locations can be one or more points, lines, or polygons and may be specified interactively or read from an ArcGIS shapefile (ESRI, 1998).

Available outputs from FlamMap depend on the algorithmic options chosen. In every case, those parameters produced by intra-cell burn computations are available. These are all rasters, and include results related to fire behavior, intensity, and spread. With the MTT option, rasters and vectors primarily related to inter-cell fire spread are

available. The fuel-conditioning option produces rasters for solar radiation, as well as computed 1- and 10-hr fuel moistures.

One output from FlamMap, fireline intensity – the rate of heat energy released per unit time per unit length of fire front – is important for the ICLF project. This output is used to compute the flame length for cells that burn in a model fire. This flame length is in turn used by another Envision plug-in module to impose vegetation changes on the model landscape due to fire disturbance.

The ICLF project is designed to use spatially explicit fire modeling and thus requires the use of the FlamMap MTT option. An MTT FlamMap DLL is available and was used in the coupled systems model. More detail on how the MTT DLL is used with Envision in the ICLF project is found later in this chapter and in Appendices A and B, *The Envision FlamMap Interpreter User's Guide*, and *The Envision FlamMap Interpreter Programmer's Guide*, respectively.

Challenges to FlamMap Implementation

A variety of approaches were taken to meet the implementation challenges outlined in the introduction to this chapter. This section discusses how these challenges were met.

Burn Parameters

Historic weather data from Remote Automated Weather Stations (RAWS) were used to correlate wildfire probability and size to the energy release component (ERC) (an index of the total energy available to a fire) at the time of each fire over an area reaching from the southern Willamette Valley south to the California border (Ager, Johnson and Evers, unpublished data). RAWS data collection included areas south of the ICLF study

area to include the higher ERCs that are projected under future climate. Future weather projections were then used to calculate daily ERCs in future model years (Johnson and Evers, unpublished data), which in turn were used to calculate future daily fire probability and burn time (used as a surrogate for fire size). Correlating historic fire size to burn time under expected fuel moistures allows the team to use burn time for modeled fires, which in turn allows extant fuels to control the rate of spread and thus fire size.

During an Envision model run, the FlamMap Interpreter uses a Monte Carlo draw with the Julian day fire probability to determine which fires are modeled during each Envision annual time step. Those fires are run on the model landscape using their defined wind direction, wind speed, and burn time in conjunction with the model landscape fuels and topography.

Model fire data are pre-computed prior to starting the Envision software. With a Monte Carlo draw determining which fires are modeled, a single dataset of fires can produce variation in fire occurrence and fire severity that reflect the expected probabilistic nature of wildfire occurrence in relation to climatic conditions.

Ignition Placement

As part of this thesis, a study was undertaken to determine the factors affecting the probability of ignitions based on landscape characteristics (Chapter 3). Using a logistic regression model from this study and landscape attributes from the Envision landscape, an ignitions probability grid for the model landscape is generated once per annual time step. For each model fire that is run, a Monte Carlo draw is performed against the ignitions probability grid to produce an ignition point. This probabilistic

method allows for variation in ignition placement while taking into account factors influencing ignition occurrence.

Landscape Characteristic Conversion

FlamMap requires eight data layers to model a landscape. Three of these – slope, aspect, and elevation – do not change over time. The other five – fuel model, canopy cover, canopy base height, canopy height, and crown bulk density – are based on dynamic vegetation characteristics that change with changes in vegetation state. In the ICLF project, Envision does not store the vegetation characteristics used by FlamMap, but does store a vegetation state for each IDU that can be used to derive these characteristics.

Other members of the ICLF team investigated the fuels and structural characteristics of each vegetation state in the state-transition model based on 1260 plots of forest stand data derived from regional data sources (Johnson, Gould and Ulrich, unpublished data). Based on their results, they produced a lookup table that indexed the fuels and stand structure characteristics required by FlamMap to the vegetation states used in Envision. Using this lookup table, the FlamMap Interpreter generates the values written to the LCP file using vegetation states it obtains from Envision IDUs.

Spatial Domain Translation

The Envision model landscape is comprised of IDUs – polygons of varying shapes and sizes. FlamMap, in contrast, operates on a regularly-spaced grid. In order to translate data between Envision's polygon space and FlamMap's grid space, spatial domain mapping software called PolyGridLookups was implemented as part of the FlamMap Interpreter. PolygridLookups is implemented as a C++ object. An object is a

data structure containing both data elements and functions (commonly called methods) that perform computational operations using the object's own data elements.

Using the spatial overlap between IDU polygons and FlamMap grid cells, PolyGridLookups creates a pair of lookup tables, one to translate data from grid space into polygon space and one to translate data from polygon space into grid space. Because the majority of entries in these tables have a value of zero, indicating no overlap, these tables are implemented using row-compressed arrays (http://web.eecs.utk.edu/~dongarra/etemplates/node373.html, Appendix B). A row-compressed array is a memory-efficient data construct that reduces the storage required for an array in which most elements have the same value.

A PolyGridLookups object is generated through a four-step process (Appendix B):

1. Obtain the polygon landscape representation from Envision.
2. Create a grid-based landscape representation matching that used by FlamMap.
3. Compute the approximate area of intersection of each grid cell with each polygon and store the results in a lookup table. This lookup table allows for the fast translation of data from grid space into data in polygon space.
4. Create a transpose (row and column entries reversed) of that created in step 3. This lookup allows for the fast translation of data from polygon space into data in grid space.

The creation of a PolyGridLookups C++ object from the initial polygon and grid landscape representations is computationally intensive and can take on the order of hours to execute. Options to read/write a PolyGridLookups object as a binary file have been

implemented in the FlamMap Interpreter so that the object needs to be created from the initial polygon and grid representations only one time per landscape. In subsequent Envision sessions the PolyGridLookups object can be created from the binary file.

The FlamMap Interpreter

While FlamMap, via the minimum travel time dynamic link library (MTT DLL), provides the core fire modeling functionality needed by the ICLF project, the fire parameters, ignition locations, and landscape description must be managed or produced by a software component that interfaces with Envsion, the MTT DLL, and external files. That software component is the FlamMap Interpreter. This section provides a high level view of the FlamMap Interpreter by describing its behavior during an Envision session, making clear its interactions with Envision, the MTT DLL, and input and output files. More detailed descriptions of its use an implementation can be found in Appendix A, *The Envision FlamMap Interpreter User's Guide* and Appendix B, *The Envision FlamMap Interpreter Programmer's Guide*.

During an Envision session, four different methods are called, each at a different point in the Envision code execution: startup, the start of a run, during each time step of a run, and at the end of a run (Figure 3). The processing executed during each of these steps is described separately below.

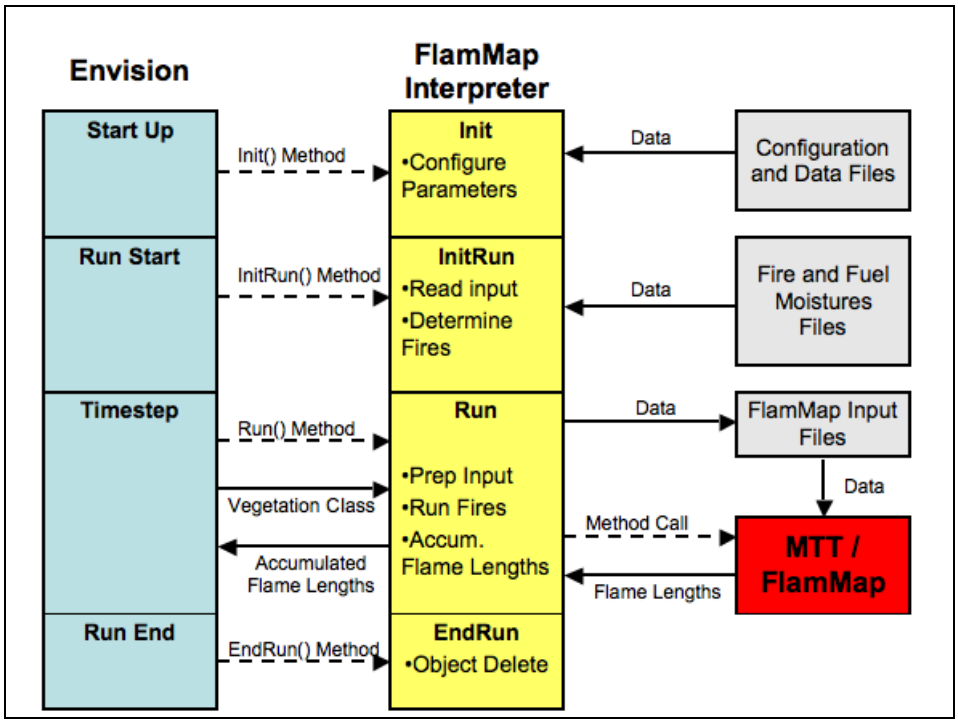

Figure 3. Overview of FlamMap Interpreter. Schematic diagram of the FlamMap Interpreter (yellow) illustrating its interactions with Envision (blue), files (grey), and FlamMap (red).

Envision Startup

Init() reads a master configuration file with parameters used to set options, the working data directory, and file names. Options include whether to create the PolyGridLookups object from scratch or read it from a file. The working data directory is the directory where files are read and written. File names include the names of the file for the vegetation class to LCP parameter data, the PolyGridLookups file, the fires file, the starting LCP file, the FlamMap input parameters file, and the root names for files produced during the Envision session including the LCP files, the FlamMap input files, and the ignition files.

After reading the input parameters, *Init()* creates C++ objects for managing the LCP file, performing the vegetation state to landscape parameter translation, and the PolyGridLookups. If the PolyGridLookups is to be written to file, this is also done.

Start of an Envision Run

At the start of an Envision run, the *InitRun()* method is called. First, the fires file – containing the model year, Julian day, fire occurrence probability, wind direction, wind speed, and burn period – is read and for each fire in the file a Monte Carlo draw determines whether that fire will be modeled during its model year. Those fires to be modeled are saved, and the others are discarded. Next, a file containing parameters that would have been specified interactively in the standalone version of FlamMap, and fuel moistures files are read. Data from these files are stored in computer memory so that they can be written for each run of FlamMap. Finally all values in the matrix that will hold the maximum flame length reached on each grid cell are set to zero.

Envision Run Time Step

During a run, Envision calls the FlamMap Interpreter *Run()* method one time for each annual model time step. *Run()* first prepares the LCP file used by FlamMap. For each polygon in the Envision landscape, *Run()* obtains the vegetation class from Envision. Using the PolyGridLookups object and the vegetation class-to-landscape values lookup, *Run()* converts the vegetation class values into gridded landscape parameters. These parameters are written to the LCP file. The LCP file is created once per Envision time step. The same LCP file is used for each fire modeled during a time step. Next *Run()* obtains the appropriate values it needs from Envision to generate the ignitions probability grid for the current model year.

Run() then loops over the fires for the model year, running each one independently of the others. For each model fire, it creates two more input files used by FlamMap: a file containing wind direction, wind speed, simulation duration time, fuel moistures, and user options; and an ignition location file. It then calls the MTT DLL to

run the model fire. Flame lengths are computed based on fire line intensity results from FlamMap. Flame lengths are accumulated in the flame length matrix. If multiple model fires affect the same grid cell, then the value of the greatest flame length modeled for the cell is used. After all fires for a model year have been run, *Run()* uses the PolyGridLookups object with the flame lengths matrix to compute the mean flame length for each IDU polygon and writes the data to the Envision data structure. Once written to the Envision data structure, flame length data are available to the vegetation model plug-in which uses flame lengths to modify vegetation states.

End of an Envision Run

EndRun() deletes the object that executed the fires for the run, a step necessary for the next run of the model.

Conclusion

The goals of the ICLF project required a fire-modeling component that could simulate spatially explicit fires on a model landscape that changes between time steps of a model run. FlamMap can provide such spatially explicit modeling on a fire-by-fire basis. However, the functionality to interface with a landscape changing in terms of both vegetation, and the number and location of ignitions, and to take into account the effect of a changing climate on wildfire frequency, size and severity had to be provided by other means. While inputs from Envision, along with pre-computed fire probabilities and parameters provide the data necessary to initialize FlamMap, the FlamMap Interpreter provides the missing functionality to convert external data into a form usable by FlamMap, to place ignitions on the landscape, to manage the running of multiple model

fires that are driven by different climatic conditions, and to provide Envision with the resultant flame length data needed to compute changes to vegetation states.

With the FlamMap Interpreter in place, the ICLF's coupled model now has the functionality to generate feedbacks among human decisions, vegetation succession, climate, and wildfire. Chapter 3 discusses how the critical linkage between human factors and wildfire ignitions used by the coupled model was generated through the statistical analysis of historical wildfires in the Willamette Valley.

CHAPTER III

THE SPATIAL DISTRIBUTION OF WILDFIRE IGNITIONS IN THE WILLAMETTE VALLEY, OREGON, USA

Introduction

Wildfire is both a costly hazard to humans and a key ecological process. In the United States alone, annual suppression costs can exceed one billion dollars and annual homes destroyed can number in the thousands (Busby and Albers, 2010). Harder to quantify are the consequences of forest product destruction (Kauffman, 1990) and the alteration of ecosystem function resulting from changes to historical fire regimes (Keeley and Fotheringham, 2003; Snider et al., 2006). Understanding wildfire – its causes, behavior, intensity, and extent – is important in predicting its likelihood and managing both its negative and positive effects. The spatial distribution of wildfire ignitions is a key component of this understanding.

Nearly all wildfire ignitions can be attributed to human activity or lightning. Both human and biophysical factors are important in predicting ignitions, with their relative importance depending primarily on ignition cause (Seidl et al., 2011). In areas where human ignitions are substantial, ignition probability is typically greater closer to roads (e.g., Syphard et al., 2008; Yang et al., 2007; Amatulli and Camia, 2007) and with greater road density (e.g., Amatulli and Camia, 2007; Cardille et al., 2001). Despite the importance of roads, to my knowledge only two studies have distinguished the effects of primary and secondary roads. Vasiliakos et al. (2009) found major roads had a stronger influence on ignitions; in contrast, Amatulli and Camia (2007) found a stronger influence from minor roads, which they attributed to a greater density of secondary roads in

wildland areas. Human ignitions also occur more frequently in and near areas of human habitation, such as the wildland urban interface (WUI) (e.g., Syphard et al., 2008; Syphard et al, 2007) and villages (e.g., Zhang et al., 2010) and cities (e.g., Cardille et al., 2001). Ignitions have also been found to be higher in areas of greater human population density (e.g., Sturtevant and Cleland, 2007; Cardille et al., 2001). However, Syphard et al. (2007) found that the number of ignitions peaked at intermediate population densities with the initial increase due to greater human presence and then a decrease due to insufficient fuels. In addition to human factors, a number of biophysical factors often are significantly correlated with human ignitions, including slope (e.g., Amatulli and Camia, 2007; Yang et al., 2007), vegetation type (e.g., Syphard et al., 2008; Syphard et al., 2007; Yang et al., 2007), elevation (e.g., Yang et al., 2007), aspect (Vasilakos et al., 2009), rainfall (e.g., Cardille et al., 2001; Pew and Larsen, 2001), and temperature (e.g., Syphard et al., 2008; Cardille et al., 2001; Pew and Larsen, 2001).

Factors associated with lightning ignitions often differ from those associated with human ignitions. Commonly, weather is a dominant factor. While the total number of lightning strikes has been shown to have little or no relationship to the number of ignitions (Rorig and Ferguson, 1999), dry lightning (lightning with little or no precipitation) is closely associated (Castedo-Dorado et al., 2011; Reineking et al., 2010; Rorig and Ferguson, 1999), and dry fuel conditions also contribute to lightning ignitions (Reineking et al., 2010; Rorig and Ferguson, 2002). The relationship of lightning ignitions to elevation is not straightforward. In Nevada, Dilts et al. (2009) found lightning ignitions more common in mountains than in basins. Castedo-Dorada et al. (2011), however, found lightning ignitions more common at lower elevations, and attribute this to

higher rainfall and lower temperatures at higher elevations. Vegetation also has been found to play a role, with lightning ignitions occurring more frequently in conifer forests than in broadleaf forests (Reineking et al., 2010; Pezzatti et al., 2009; Krawchuk et al., 2006), and in woodlands as opposed to non-forested areas (Dilts et al., 2009). A study in a western Canadian boreal forest found lightning ignitions were more frequent closer to roads (Arienti et al. 2009). This was attributed to the increased presence of native and invasive grasses along roads and the associated increased fuel flammability.

The complex and varied interactions between ignition cause and the factors affecting their distribution highlight the importance of assessing human and lightning ignitions in landscapes where both are present. Oregon's Willamette Valley offers such a location. It contains a spatially diverse mosaic of population centers, large areas of WUI, rural areas managed for agriculture, forestry and wilderness, and a network of roads ranging from major travel routes to seldom-used forest roads. Using data from this area, I examine two questions: *How do various human and land-use factors influence wildfire ignition distribution?* and *How do the relationships with these factors vary among all ignitions, human ignitions, and lightning ignitions?*

Methods

Study Area

The Willamette River Basin is situated in western Oregon between the crests of the Cascade Mountains to the east and the Coastal range to the west. It is approximately 290 km from north to south and 160 km from east to west, encompassing 29,727 square kilometers. The valley floor is comprised primarily of agricultural land, urban and rural residential development, and small amounts of remnant riparian forest, prairies and oak

habitats. The valley foothills contain a mixture of conifer, broadleaf and mixed forests, agriculture (particularly pasture), and rural residential development. Conifer forests dominate the slopes of the Cascades and the Coastal Ranges. The area is subject to wet, mild winters and dry, mild summers (Hulse et al., 2002).

The three major urban areas in the valley, Portland, Salem, and Eugene-Springfield, all lie on the valley floor. Human population is generally higher on the valley floor, the surrounding foothills, and along major tributaries of the Willamette River. Major and minor roads (interstate highways, principal arterials, minor arterials, major urban collectors, minor urban collectors, and local roads) are concentrated in the same areas. Unidentified roads and trails are common from the foothills to the higher reaches of the mountains on both sides of the valley (Hulse et al., 2002).

Ignition Data

Ignitions data, comprising all fire events logged by a federal or state agency, were derived from the Oregon Department of Forestry Fire Database (ODF, 2010) and data from the Bureau of Land Management Northwest Interagency Coordination Center (May, 2010). No reliable data were available for the valley floor and it was excluded from the study, resulting in an approximately 21,600-km^2 horseshoe-shaped area comprised mostly of hilly-to-mountainous, forested, sparsely populated terrain, with small areas of towns, agriculture, and WUI. The resulting dataset consisted of 3505 ignitions from the years 2000 through 2009, each with the date, location (as ArcGIS shapefile points (ESRI, 1999-2008) from ODF, and as latitude and longitude from BLM), area burned, and ignition cause.

Two further sets of points were needed for analyses: a) a set of random points to represent overall landscape characteristics, and b) a set of random non-ignition points for statistical model development, which utilized equal numbers of ignition and non-ignition data points. . Using ArcMap (ESRI, 1999-2008), a 30-m grid of cells was superimposed over the study area and grid cells were selected randomly using a Perl 5.8.8 (http://perldoc.perl.org/5.8.8/index.html) script. The centers of the randomly selected grid cells defined the random points. Because no random point was within 30 m (the resolution of grid data for the study) of an ignition, the same points were used to represent landscape characteristics and the set of non-ignition points.

Explanatory Variables

Nineteen explanatory variables (Table 1) were derived from Pacific Northwest Ecosystem Research Consortium (PNW-ERC) GIS datasets (http://www.fsl.orst.edu/pnwerc/wrb/access.html). For distance to road variables, I reduced their seven road types into three categories: major roads (interstate highways, principal and minor arterials, and major/urban collectors); minor roads (minor collectors and local roads); and unidentified roads (unidentified or unconfirmed roads or trails). Population density was converted from source data units of square miles to the equivalent metric units, 2.59 km^2. To explore non-linear effects with logistic regression, quadratic terms were included for all continuous and integer variables, and cubic terms were included for distance to major roads, distance to minor roads, and population density.

Table 1. Model explanatory variables.Summary of explanatory variables used in statistical analysis of ignition location.

Numeric Explanatory Variables (units)	**Data Type**	**Mean Value**	**Std. Dev.**
Distance to major road (m)	Continuous	4,603	3,895
Distance to minor road (m)	Continuous	2,503	2,922
Distance to unidentified road (m)	Continuous	576	759
Distance to railroad (m)	Continuous	16,482	14,063
2.59-km^2 population density (people / 2.59 km^2)	Integer	24	203
25.9-km^2 population density (people / 25.9 km^2)	Integer	290	1,186
Land Use / Land Cover Categories:		**Percent of Landscape**	
Hardwood forest	Binary	7.90	
Younger conifer forest	Binary	36.03	
Older conifer forest	Binary	12.23	
Open forest	Binary	0.37	
Semi-closed forest	Binary	1.74	
Closed forest	Binary	29.70	
Shrub	Binary	2.51	
Prairie	Binary	0.17	
Crops	Binary	5.91	
Developed	Binary	0.40	
Roads	Binary	0.68	
Rural structures	Binary	0.31	
Other	Binary	2.02	

The sixty PNW-ERC land use/land cover (LULC) classes were grouped into 12 categories (Table 1). Conifer forests greater than 200 years old were categorized as *older conifer*, with the remainder of conifer forest categorized as *younger conifer*. The *other* category included LULC classes for water, marshes, snow, rural non-vegetated, and barren. The roads LULC class was excluded from analysis because it was redundant with the distance to roads variables.

Statistical Samples

Explanatory variable values were assigned to ignition points using ArcMap. To avoid edge affects along the interior boundary of the study area, the valley floor was clipped after grids were computed. On the outer edge of the study area, variation in

human population and roads is minimal so there is little to no edge effect. Thirty-meter grids were computed for all explanatory variables. The 25.9-km^2 population density for each cell was computed from the 2.59-km^2 densities using a 25.9-km^2 circle around each grid cell center. Distances of ignition points to roads were calculated using Euclidian distance. Autocorrelation among explanatory variables was checked by using the non-ignition dataset to compute Pearson product moment correlations between all pairs of continuous explanatory variables. No correlation exceeded 0.57, and all variables were retained for analysis.

Ignitions were assigned to three classes: all (3,505 ignitions), human (2,364 ignitions), and lightning (995 ignitions). No cause was available for 146 points. For statistical modeling, each of these groups was matched with an equal number of non-ignitions. Each dataset was then randomly divided into two equal-sized training datasets, one for model development and one for model evaluation (De'ath and Fabricius, 2000).

Statistical Methods

Mean fire sizes for the three ignitions datasets were examined using the non-parametric Kolmogorov-Smirnov test (Chakravarti et al., 1967) to determine if the fire size distributions were statistically different for human and lightning ignitions. T-tests were performed using log-transformed data for distance and population density variables between datasets for landscape, all ignitions, human ignitions, and lightning ignitions. Histograms, medians, and percentages of the individual explanatory variables (Table 1) were compared between the landscape and ignition datasets to assess their effects on the spatial distribution of ignition locations. For LULC categories, G-tests were performed to

characterize differences between proportional occurrences in the ignition datasets and the overall landscape.

To explore differences and similarities in the results from different multivariate statistical techniques and to provide a means of qualitative cross-validation, I analyzed the all ignitions data with two methods: classification and regression tree analysis (CART) and logistic regression.

Classification and Regression Tree (CART)

CART analysis was performed only on the all-ignitions dataset (Breiman et al., 1984; Therneau and Atkison, 1977; De'ath and Fabricius, 2000; Urban, 2002). CART is a non-parametric technique in which data are recursively split into mutually exclusive groups (or nodes) to form a tree structure, with final groups of data residing at the terminal nodes, or leaves. Splits maximize homogeneity within groups and the same explanatory variable may be used to generate multiple splits in a tree. CART's one-step look-ahead algorithm does not necessarily produce optimal partitioning. Models produced by CART are commonly over-fitted, so pruning was done by removing splits that resulted in < 0.02 increase in the misclassification rate as determined by using the evaluation dataset. CART analysis was done using R statistics software (http://www.r-project.org/) and the RPART package (http://cran.r-project.org/web/packages/rpart/index.html). The ANOVA method was used for splitting data into groups.

Logistic Regression

Logistic regression was performed using datasets for all, human, and lightning ignitions. However, with over 134 million possible candidate models for each dataset, it

was necessary to use a two-phase process to limit the number of models considered. Phase one eliminated variables with little explanatory power, while phase two determined sets of "best-candidate" final models.

In phase one, subsets of models with different numbers of variables were considered, with an upper limit of ten variables to avoid overly complex models. For a given subset size, if fewer than 100,000 models existed, all possible models were considered. Otherwise, 100,000 models were randomly generated. Akaike Information Criterion (AIC: Burnham and Anderson, 2002) was used to choose a candidate set of models from the approximately 600,000 models generated in phase one. Variables present in any model whose AIC value was within 2 of the minimum AIC were included in the phase 2 analysis. In addition, any variable present in more than 20 of the models with the 100 minimum AIC values were included with the idea that variables with strong explanatory power would be more commonly present in models with lower AIC values.

Phase two consisted of an analysis on all-possible-subsets of ten or fewer of the explanatory variables yielded by phase one. Models with an AIC within 2 of the minimum AIC were considered as "best-candidate models".

Model Evaluation

Three methods were used to more fully evaluate one best-candidate model from each dataset: misclassification rates (Pearce and Ferrier, 2000); plots of binned, averaged actual data values versus computed model values (after Pearce and Ferrier, 2000); and calculations of a statistic, "quasi R^2," that I developed for this study. The first method is suitable for evaluating both CART and logistic regression models, while the latter two are only suitable for logistic regression models.

To determine the misclassification rate, the statistical model was applied to the evaluation dataset. The proportion of data points whose modeled outcome differs from its actual outcome is the misclassification rate. CART models are binary, producing true/false results that can be compared to the actual true/false values in the evaluation dataset. For logistic models, which produce a continuous probability distribution, a calculated probability cutoff value was used to classify calculated results as true or false. For instance, if the cutoff value is 0.4, those data points with a calculated probability ≥ 0.4 are counted as correctly classified, whereas those with a calculated probability value < 0.4 are counted as misclassified. The cutoff value used to determine the final misclassification rate for a continuous probability model is that cutoff value which yields the lowest misclassification rate. A misclassification rate of 0.50 is expected from random choices due to the equal size of the ignitions and non-ignitions datasets.

Plots of averaged actual data values versus computed model probabilities were produced by first dividing the predicted probabilities into 20 equal-sized bins. For each bin, the mean of the values of points whose calculated probability fell within the bin was plotted against the midpoint of the observed probability for points in the bin (after Pearce and Ferrier, 2000). This provides a visual representation of how closely the observed data fit the calculated probability.

Finally, to evaluate the relative closeness of model fit for the binned plots, I calculated what I term a "quasi R^2" (R_Q^2) that is identical to the general definition of the coefficient of determination except that the sums of squares are weighted by the number of observations in each bin. The resulting formula is:

$$R_Q^2 = 1 - \frac{\sum_{i=1}^{m} n_i * (\overline{o}_i - \overline{c}_i)^2}{\sum_{i=1}^{m} n_i * (\overline{o}_i - \overline{O})^2}$$

where m is the number of bins, n_i is the number of observations in bin i, $\overline{o}_i$ is the mean value of observed data in the bin, $\overline{c}_i$ is the mean probability of model values calculated in the bin, and $\overline{O}$ is the mean value for all observations.

Ignition Probability Maps

Using the single model generated by the CART analysis and one model selected from each of the logistic regression analyses based on best-candidate model results, ignition probability values were generated for each 30-m grid cell in the study area. These are relative probabilities that would be normalized to sum to 1 for all cells in the landscape if used to predict the location of an individual ignition. ArcMap was used to convert these sets of probability values into ignition probability surface maps.

Results

Descriptive Statistics

For all ignitions, the total area burned was 8,087 ha over 10 years with a mean fire size of 2.3 ha. Differences in the fire size distribution of human and lightning ignitions were statistically significant. Human ignitions comprised 67% of ignitions and accounted for 42% of area burned with a mean fire size of 1.4 ha. Lightning ignitions comprised 28% of ignitions and accounted for 52% of area burned with a mean fire size of 4.2 ha. Ignitions of unknown cause comprised 4% of ignitions and accounted for 6% of area burned with a mean fire size of 3.5 ha.

Human ignitions occurred nearer to major roads, minor roads, and railroads than the landscape as a whole (Table 2, Figure 4, a, c, e, and g) while lightning ignitions showed the opposite relationship (Table 2, Figure 4, a, d, e, and h). These trends were stronger for major roads than for minor roads and railroads. All ignitions showed trends similar to those shown by human ignitions, but less pronounced (Table 2, Figure 4, a, b, e, and f) due to the effect of the inclusion of lightning ignitions. No such trends were observed for unidentified roads (Table 2).

Table 2. Median road distances and population densities. Median distances to major and minor roads, and 2.59- and 25.9-km^2 population densities, for all, human, and lightning ignitions. Lower case letters after values indicate significantly different groups.

	Expected (Landscape)	**All Ignitions**	**Human Ignitions**	**Lightning Ignitions**
Distance to major road	3,617 a	1,797 b	1,025 c	4,823 d
Distance to minor road	1,345 a	637 b	361 c	2,911 d
Distance to unidentified road	351 a	409 bc	428 b	388 c
Distance to railroad	11,758 a	10,001 b	7,432 c	21,380 d
2.59-km^2 population density	0 a	2 b	14 c	0 d
25.9-km^2 population density	14 a	91 b	246 c	0 d

Both all ignitions and human ignitions occur in areas of higher-population density (both 2.59 km^2 and 25.9 km^2) than that of the landscape as a whole (Table 2, Figure 4, i-k). The opposite is true for lightning ignitions (Table 2, Figure 4, i and l).

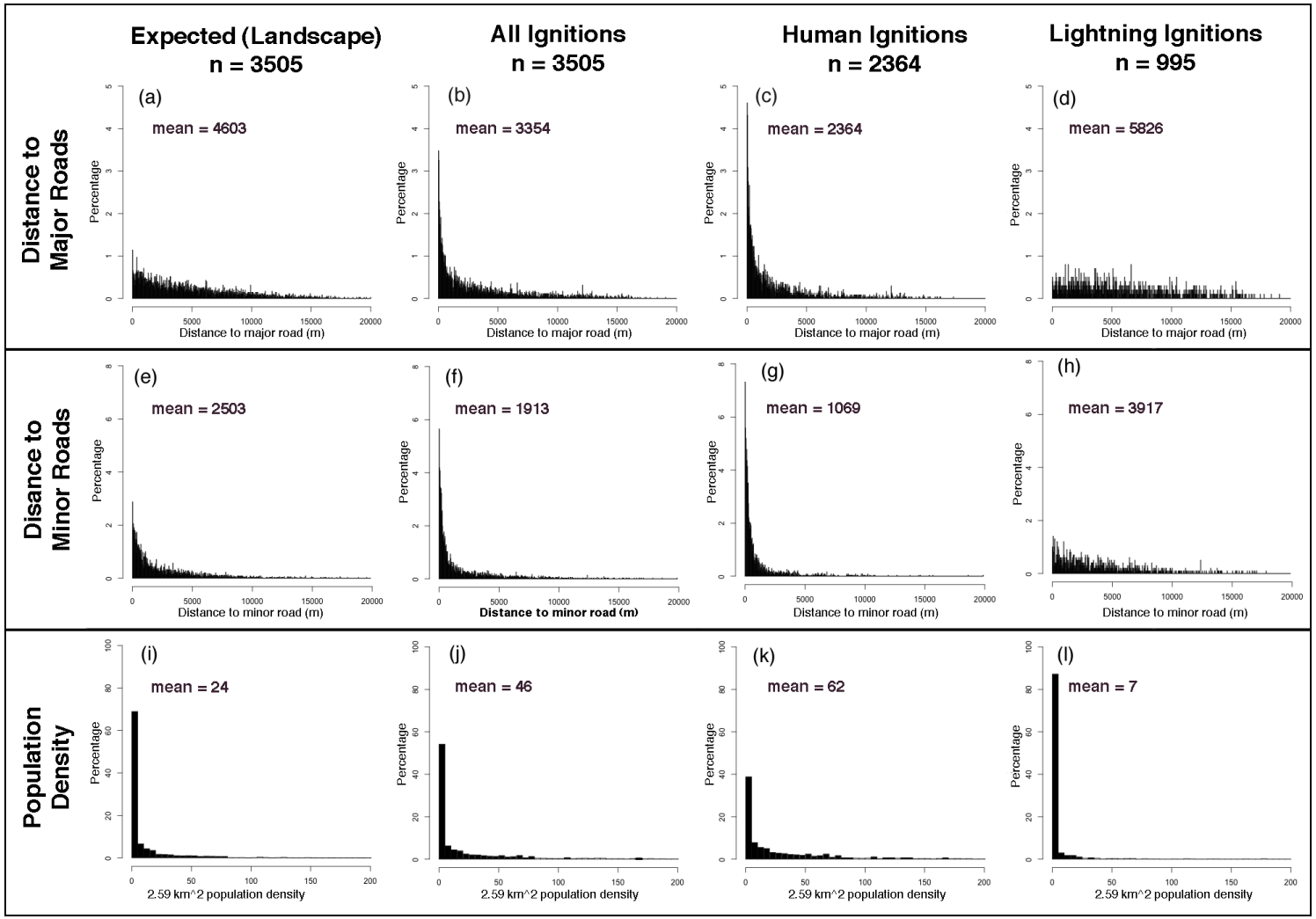

Figure 4. Distributions of distance to roads and population density. Histograms for distance to major roads (a-d), distance to minor roads (e-h), and 2.59-km^2 population density (i-l). Intervals for distances (a-h) are 30 m. Population density histograms were truncated at 200 people per 2.59 km^2, which includes 98.5% of the landscape and 95.3% of ignitions.

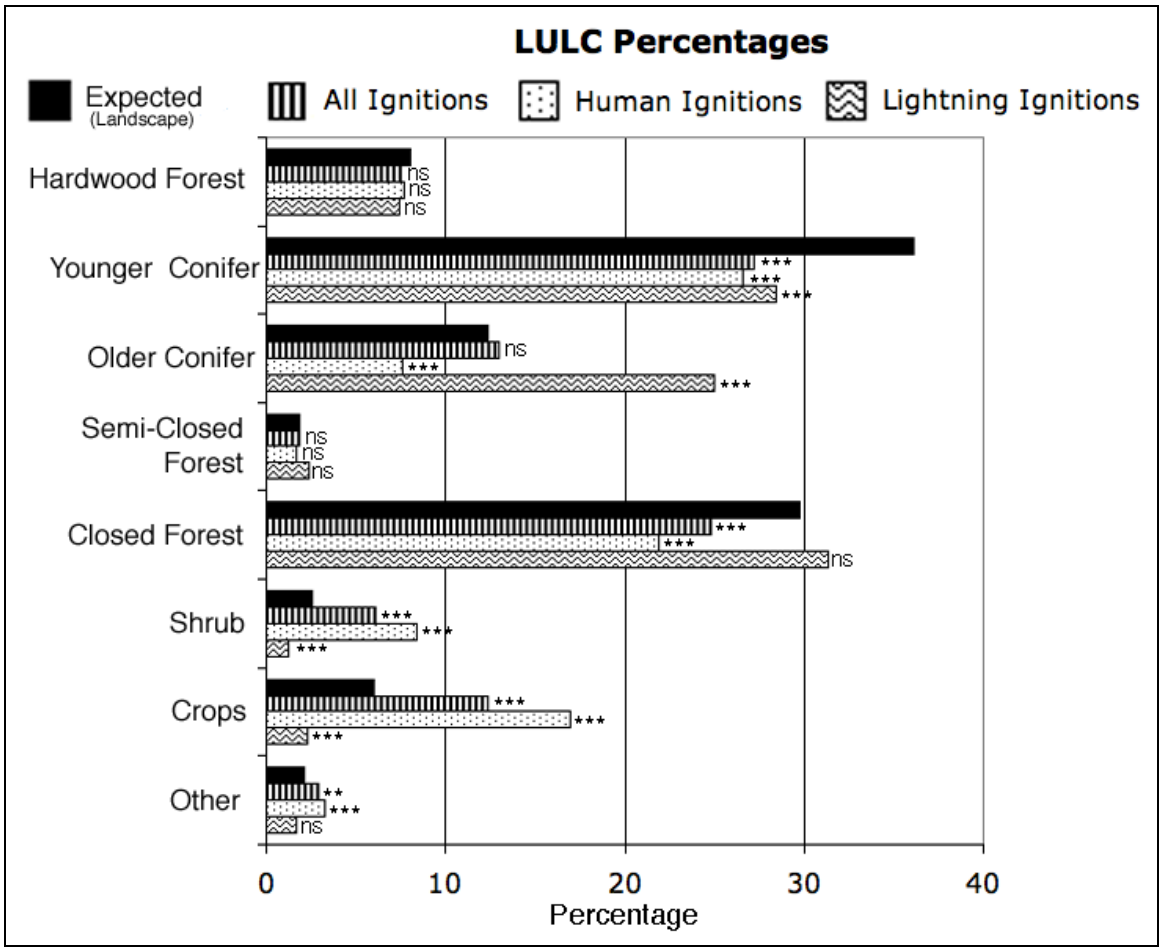

Figure 5. LULC categories in datasets. Percentages of LULC categories within the expected (whole landscape), all ignitions, human ignitions, and lightning ignitions datasets for categories comprising greater than one percent of the non-ignitions dataset. Symbols at the end of the bars indicate the significance of differences between the factor for the ignition datasets and the "expected" dataset. *** ≤0.001; ** ≤ 0.01; ns = not significant.

Several of the LULC categories showed significant differences between their occurrence in the ignitions dataset and in landscape as a whole (Figure 5). All three types of ignitions occurred less frequently than expected in younger conifer forest. All ignitions and human ignitions occurred less frequently than expected in closed forest. Ignitions in old-growth conifer forest were lower than expected for human ignitions but nearly twice as high as expected for lightning ignitions. All ignitions and human ignitions occurred

more frequently than expected in the shrubs, crops, and the "other" categories, while lightning ignitions occurred less frequently than expected in shrubs and crops.

CART

Distances to major and minor roads were the only two explanatory variables for ignition probability in the CART analysis (Figure 6). Each accounted for a single split: at 359 m for major roads and at 294 m for minor roads.

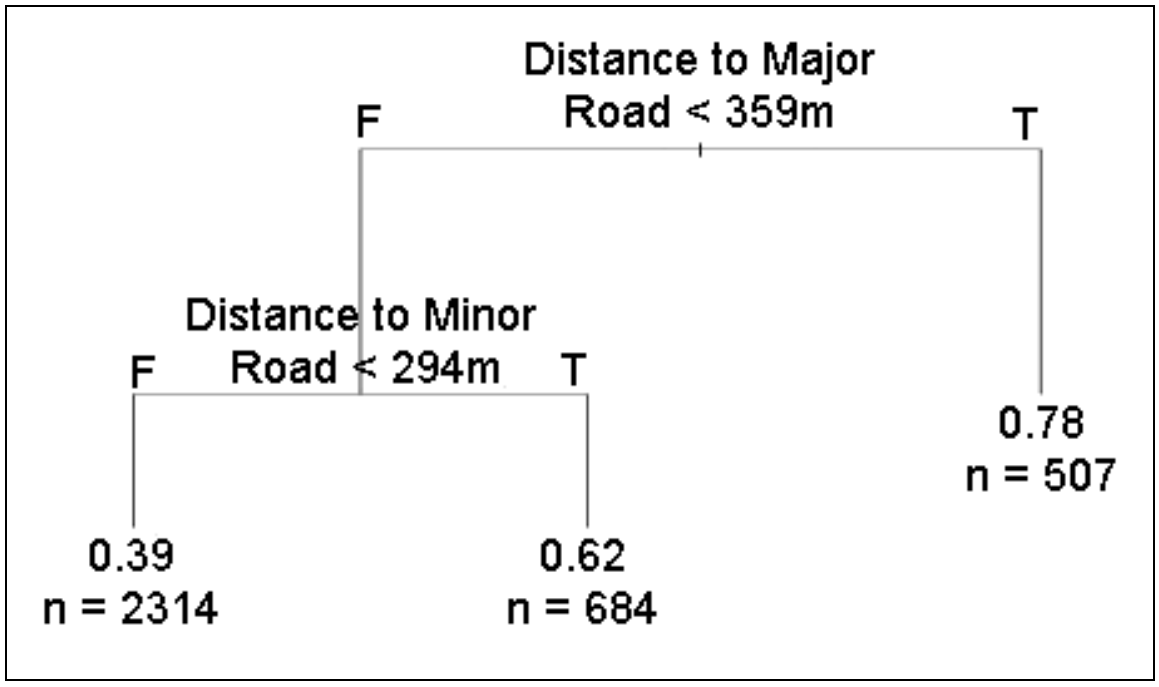

Figure 6. All-ignitions classification and regression tree. Classification and regression tree computed from the all-ignitions training dataset. **T** and **F** indicate the branches of the tree corresponding to whether the condition was met or not, respectively. The decimal fraction at each leaf indicates the proportion of data points from the parent node with an ignition. "n" indicates the number of data points from the evaluation dataset associated with each leaf.

Logistic Regression

In phase one, 16, 14, and 23 variables met one or both criteria for all, human, and lightning ignitions, respectively (Table 3). Phase two yielded three best-candidate models (within 2 of the lowest AIC value) for all ignitions, one for human ignitions, and 141 for

lightning ignitions. For all ignitions, the model with the lowest AIC was chosen for further analysis. Its explanatory variables (Table 4) included all terms for distance to major and minor roads, the quadratic term for distance to railroad, and variables for hardwood forest, younger conifer forest, and closed forest. All variables were statistically significant.

Table 3. Phase 1 explanatory variables. Explanatory variables considered in phase 1 of the logistic regression analysis. Variables present in 20 or more of the models with the lowest 100 AICs or in any model within 2 of the lowest AIC were used in the phase 2 analysis. (Abbreviations: All: All ignitions; Hum: human ignitions; Ltg: lightning ignitions).

	Presence within 2 of minimum AIC			**Presence in lowest 100 AICs**			**Used in phase 2 of analysis**		
Explanatory Variable	**All**	**Hum**	**Ltg**	**All**	**Hum**	**Ltg**	**All**	**Hum**	**Ltg**
Distance to major road	X	X	X	100	100	69	X	X	X
(Distance to major road)^2	X	X	X	100	97	24	X	X	X
(Distance to major road)^3	X	X	X	94	74	12	X	X	X
Distance to minor road	X	X	X	100	100	100	X	X	X
(Distance to minor road)^2	X	X	X	96	100	94	X	X	X
(Distance to minor road)^3	X	X	X	81	91	26	X	X	X
Distance to unidentified road		X	X	15	38	30		X	X
(Distance to unidentified road)^2			X	20	12	8	X		X
Distance to railroad	X		X	18	14	97	X		X
(Distance to railroad)^2	X		X	31	6	6	X		X
2.59-km^2 population density	X		X	24	48	9	X	X	X
(2.59-km^2 population density)^2	X			20	22	12	X	X	
(2.59-km^2 population density)^3		X	X	25	28	15	X	X	X
25.9-km^2 population density			X	9	14	45			X
(25.9-km^2 population density)^2				11	9	6			
Hardwood forest	X		X	25	11	23	X		X
Younger conifer forest	X	X	X	43	34	22	X	X	X
Older conifer forest			X	16	14	93			X
Open forest			X	17	9	9			X
Semi-closed forest			X	8	12	19			X
Closed forest	X		X	30	19	57	X		X
Shrub				18	18	7			
Prairie	X			12	12	5	X		
Crops		X	X	16	35	13		X	X
Developed		X	X	9	23	15		X	X
Rural structures		X	X	9	16	6		X	X
Other			X	9	7	9			X
Total into Phase 2	**14**	**12**	**23**	**14**	**13**	**12**	**16**	**14**	**23**

For human ignitions, the single best-candidate model (Table 4) was analyzed further. It included all explanatory variables for distance to major and minor roads, the

linear term for distance to unidentified road, and all terms for 2.59-km^2 population density. All variables were statistically significant.

For lightning ignitions, nine explanatory variables were present in more than 70% of best candidate models (Figure 7): the linear terms for distance to major road, minor road, unidentified road, and railroad; the quadratic term for distance to minor road; 25.9-km^2 population density; older conifer; and mixed closed forest. No other variables were present in more than 33% of best candidate models. For further analysis, the model using only those nine most prevalent values was chosen (Table 3). Its AIC value was within 0.02 of the lowest AIC value. Of the nine explanatory variables in the model, three were not statistically significant: distance to unidentified road, 25.9-km^2 population density, and closed forest.

For explanatory variables in common between the all ignitions and human ignitions models, the signs of coefficients were similar. For explanatory variables in common between lightning ignitions and all ignitions, and between lightning and human ignitions, the signs of the coefficients were opposite (Table 4).

Table 4. Explanatory variable coefficients. Coefficients for explanatory variables for selected logistic models for all ignitions, human ignitions, and lightning ignitions. Significance (p-value) codes: *** ≤ 0.001; ** ≤ 0.01; * ≤ 0.05; † ≤0.1; ns = not significant.

Coefficient	All Ignitions	Human Ignitions	Lightning Ignitions
Intercept	+1.172e+00 ***	+1.250e+00 ***	-1.223e+00 ***
Distance to major road	-4.375e–04 ***	-5.533e–04 ***	+4.273e–05 *
(Distance to major road)^2	+4.902e–08 ***	+6.869e–08 ***	
(Distance to major road)^3	-1.550e–12 ***	-2.539e–12 ***	
Distance to minor road	-3.515e–04 ***	-6.681e–04 ***	+1.959e–04 ***
(Distance to minor road)^2	+5.213e–08 ***	+1.017e–07 ***	-1.337e–08 **
(Distance to minor road)^3	-2.142e–12 ***	-4.098e–12 ***	
Distance to unidentified road		-1.905e–04 ***	+1.729e–04 ns
(Distance to unidentified road)^2			
Distance to railroad			+2.107e–05 ***
(Distance to railroad)^2	+1.397e–10 *		
2.59-km^2 population density		+1.243e–02 ***	
(2.59-km^2 population density)^2		-3.843e–05 ***	
(2.59-km^2 population density)^3		+2.759e–08 ***	
25.9-km^2 population density			-1.778e-04 ns
Hardwood forest	-3.473e–01 *		
Younger conifer forest	-3.730e–01 ***		
Older conifer forest			+6.514e-01 ***
Closed forest	-3.132e–01 ***		+2.740e-01 †

Model Evaluation

Misclassification rates for all models were between 0.30 and 0.40 (Table 5), and R^2_Q values for logistic models were approximately 0.9, indicating a strong fit between the models and the data. Both misclassification rates and R^2_Q values were better for the human and lightning models than for the all-ignitions model. Plots of binned mean observed values versus binned mean calculated logistic probabilities (Figure 8) show a close fit between the plotted points for each dataset, with most values very close to the 1:1 line, especially those representing larger numbers of data points.

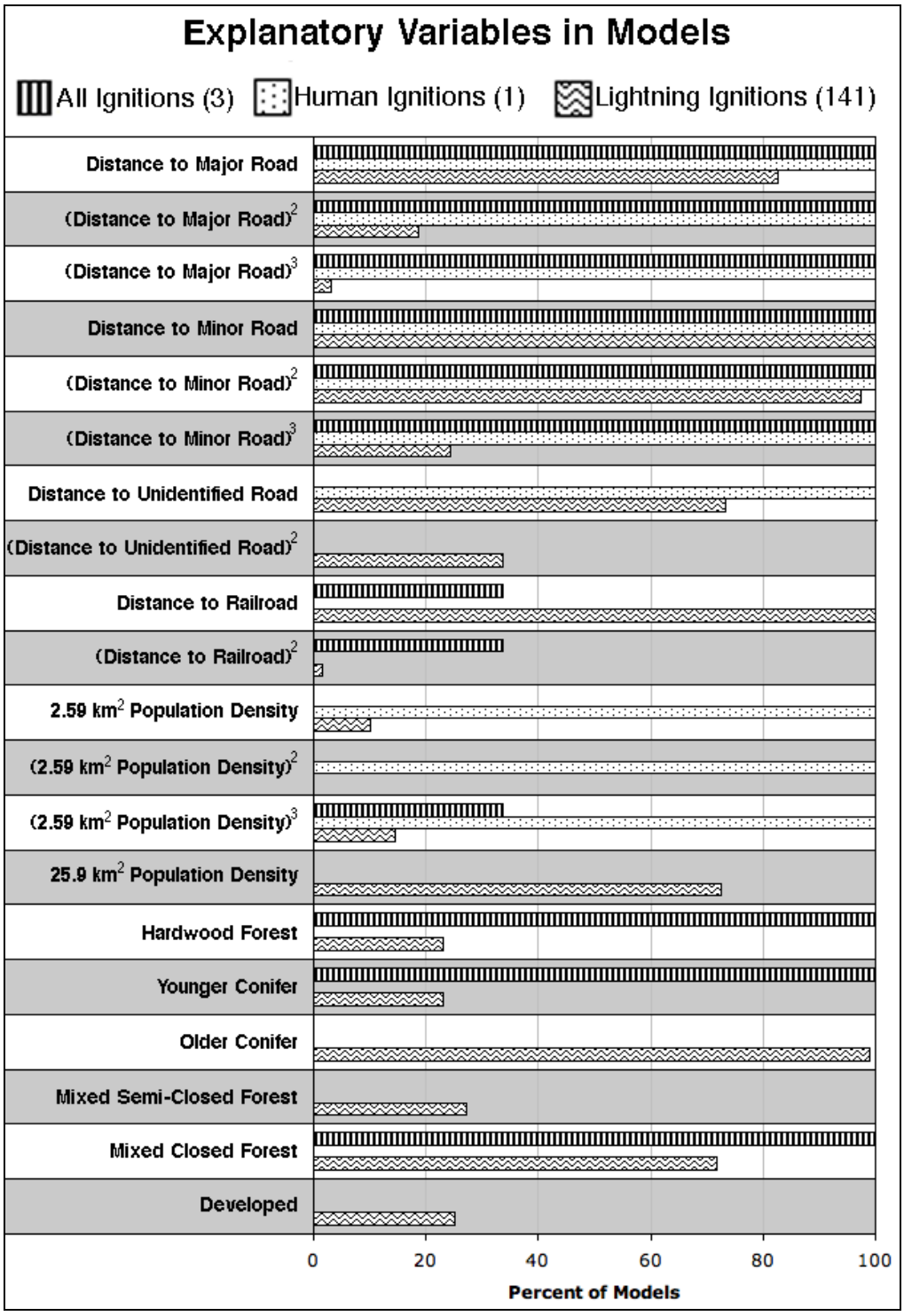

Figure 7. Explanatory variables in best-candidate models. Percent presence of explanatory values in candidate logistic regression models within 2 of the lowest AIC for all ignitions, human ignitions, and lightning ignitions. The number in parentheses after key titles is the number of candidate models. (Note that with a single candidate model, variables for human-caused ignitions are either 0% or 100%.) Variables present in less than 10 percent of models for all datasets are not shown.

Table 5. Model evaluation measures. Misclassification rates and quasi-R^2 values for CART analysis of all ignitions and minimum AIC logistic regression models of all, human, and lightning ignitions.

Analysis	Misclassification Rate	Quasi R^2
CART	0.390	NA
Logistic: All Ignitions	0.393	0.878
Logistic: Human Ignitions	0.305	0.931
Logistic: Lightning Ignitions	0.359	0.894

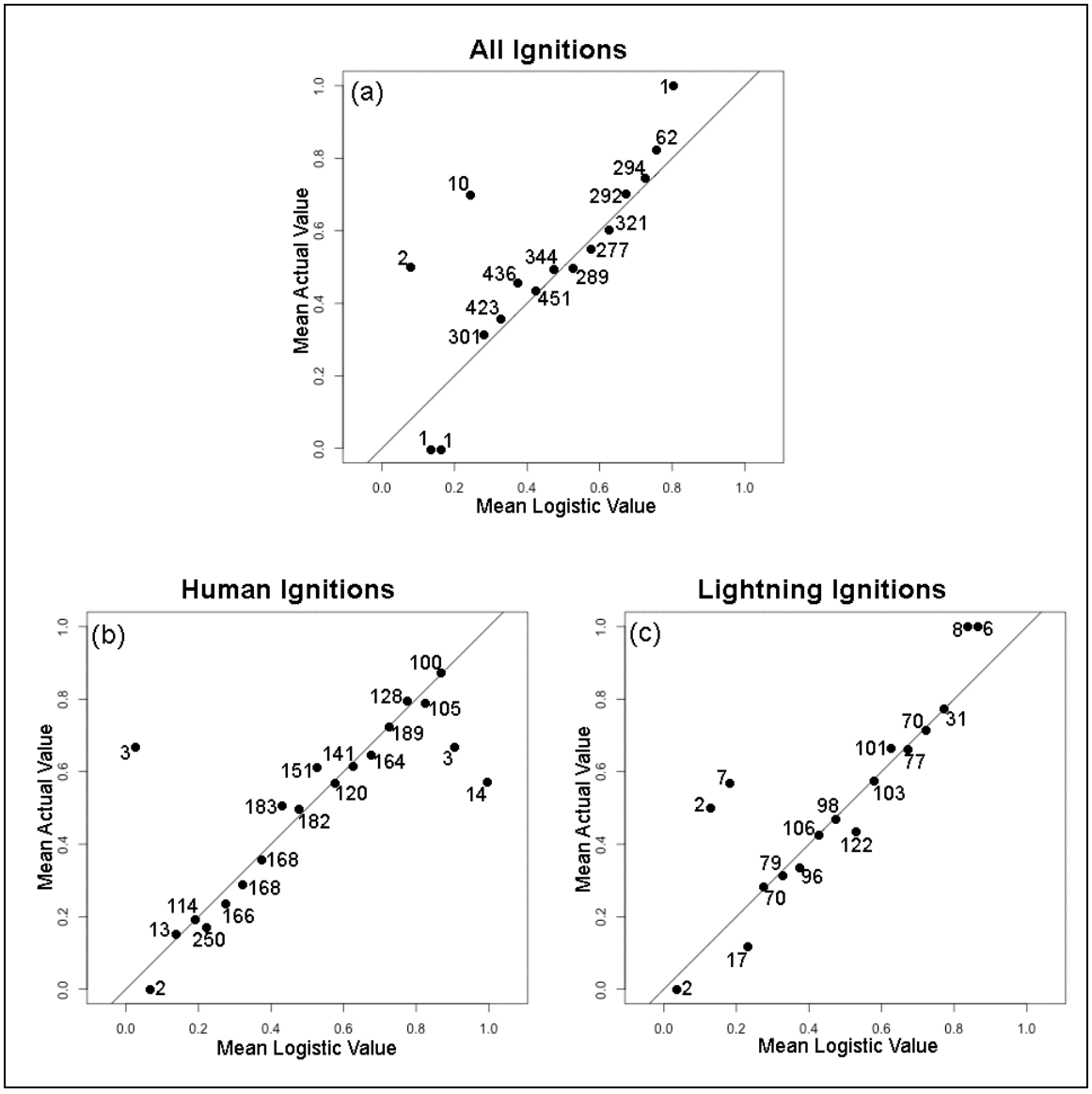

Figure 8. Logistic model evaluation plots. Binned mean actual values versus binned mean calculated logistic probabilities for **A** all ignitions, **B** human ignitions, and **C** lightning ignitions. Bin size is 0.05. Plotted points are labeled with the number of data points comprising the mean value it represents.

Ignition Probability Maps

Ignition probability surface maps for a portion of the study area (approximately 4,400 km^2 or 5% of the study area) illustrate the results of the logistic and CART models (Figure 9). High probability areas in the all-ignitions logistic model were concentrated near roads. The CART model produced three probability classes of 0.78 near major roads, 0.62 near minor roads, and 0.39 over the remainder of the landscape. In the logistic model for human ignitions, high probabilities were concentrated near roads, more so near major than minor roads, and in areas of higher population. In contrast, the logistic model for lightning ignitions showed low probabilities concentrated near roads and higher probabilities away from roads. The human ignitions logistic model showed a sharper contrast near roads than did the all-ignitions logistic model (Figure 9 b and d). The probability ranges were greater for both the human ignitions logistic model (0.0 to 1.00) and the lightning ignitions logistic model (0.0 to 0.97) than for the all ignitions logistic model (0.027 to 0.84).

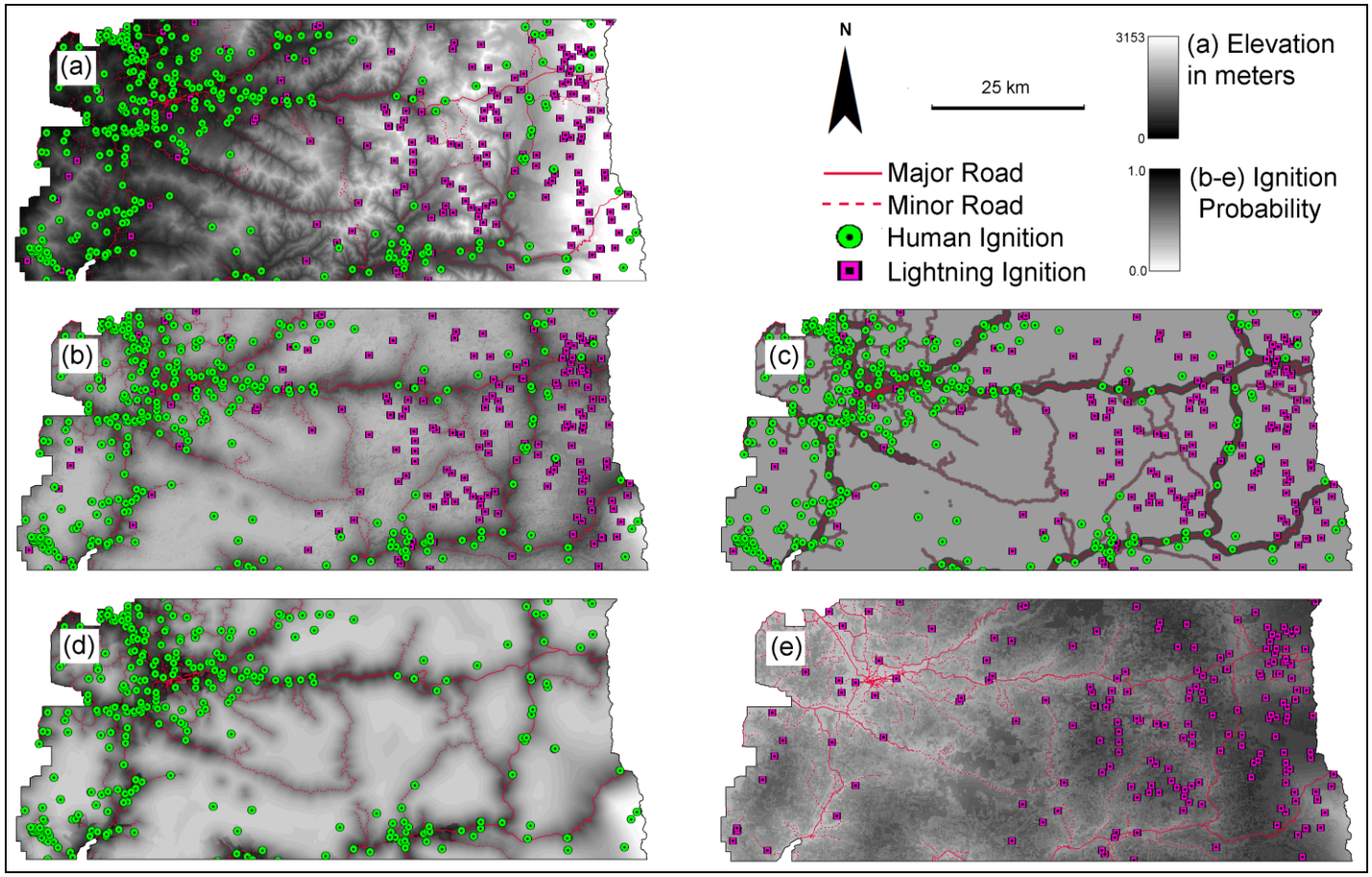

Figure 9. Ignition distributions in the study area. Distribution of ignitions over a portion of the study area. (a) digital elevation model with all ignitions, (b) probability surface and ignitions for the all ignitions logistic model, (c) probability surface and ignitions for the all ignitions CART model, (d) probability surface and ignitions for the human ignitions logistic model, and (e) probability surface and ignitions for the lightning ignitions logistic model. Mapped area covers valley foothills from north of Springfield, OR in the southwest corner to the Lebanon, OR area in the northwest corner and to the crest of the Cascade range on the east.

Discussion

The factors controlling the probabilities of human and lightning ignitions differed substantially and considering each type of ignition separately improved ignition location prediction. Compared to the landscape, human ignitions were more common nearer to indicators of human presence and activity. The higher relative probabilities along major roads as compared to along minor roads suggest a proportional relationship between the level of human activity and ignition probability. This is consistent with Vasiliakos et al., (2009), but stands in contrast to Amatulli and Camia (2007). Ignition probabilities are higher in areas of higher population, which is consistent with other studies (e.g., Sturtevant and Cleland, 2007; Cardille et al., 2001). There is no apparent reduction of probability in the most highly populated areas as found by Syphard et al. (2007), this effect would likely not be present due to the absence of large, densely populated regions within the study area.

Lightning ignitions, on the other hand, were more common further from indicators of human presence and activity. Although Arienti et al. (2009), found the opposite relationship for lightning ignitions and distance to roads, no other study I am aware of has found any association between human factors and lightning ignitions. I do not attribute a causal relationship between lightning ignitions and human factors, however. Instead, it appears that the human factors serve as surrogates for high elevation and ridgelines in the model. As is visually apparent in Figure 9(a) and in a map shown in Rorig and Ferguson (1999), lightning ignitions are more common at higher elevations in the study area. Major and minor roads tend to run through valleys in the study area. Population centers tend to be at lower elevations near the Willamette Valley floor, and to

a lesser extent along roads in the foothills and mountains. Increased lightning ignition probability with higher elevation is consistent with Dilts et al. (2009).

Other studies have found lightning ignitions to be correlated with conifer forest area (Reineking et al., 2010; Pezzatti et al., 2009; Krawchuk et al., 2006). I found lightning fires to be strongly associated with the area of older conifer stands, and to be somewhat negatively associated with the area of younger conifer stands. Whether these relationships are due to vegetation type or vegetation distribution with respect to elevation is unclear.

Analyzing all ignitions together produces meaningful results, but masks the more accurate and precise relationships that are revealed when they are analyzed separately. For the all-ignitions dataset, increasing ignition probability is strongly influenced by closeness to major and minor roads. This is apparent in both the sharply delineated CART probability surface map, with its three probability zones, as well as the more nuanced map produced by the logistic model. With over two thirds of ignitions caused by humans, it is not unexpected that the logistic models for human ignitions and all ignitions produce similar probability surface maps. However, the effects of population density that appear in the logistic model for human ignitions are lost, as is the negative association between lightning ignitions and roads. Furthermore a dampening effect is apparent with a less sharply defined probability surface map and a narrow range of probabilities than for either the human ignitions or lightning ignitions logistic models. Also lost in considering the two groups of ignitions together is the fact that fires from human ignitions tend to be smaller than those from lightning ignitions. This is likely due to

greater difficulties associated with fighting the more remote lightning fires (Weber and Stocks, 1998).

Conclusions

The differences between human and lightning ignitions are striking and they should be considered separately in terms of both cause and effect. Including lightning ignitions with human ignitions in predictive models can have a confounding effect on results. Excluding them altogether can mask their disproportionate effects on area burned and present a false impression about the proportion of ignitions that occur close to human infrastructure. Unless lightning ignitions in a study area can truly be considered negligible, studies seeking to present a complete picture of wildfires and their ignitions should consider both human ignitions and lightning ignitions, and consider them separately.

The absence of topographic factors, most notably elevation, was a weakness of this study, especially in light of the apparent importance of elevation in determining the distribution of lightning ignitions in the study area. A reevaluation of the data with slope, aspect, and elevation as explanatory factors is planned and will likely result in a more satisfactory model for lightning ignitions.

The differing degrees of probability associated with major and minor roads illustrates the importance of considering not just the presence of human activity on the landscape, but also the level of that activity. More precision in factors associated with human presence and activity, for instance utilizing factors such as vehicle traffic levels in conjunction with road class, may provide more accuracy in ignition location models and would be worth pursuing in future research.

CHAPTER IV

CONCLUSIONS

This thesis has dealt with two very different aspects of The Interactions of Climate Change, Land-Management Policies, and Forest Succession on Fire Hazard and Ecosystem Trajectories in the Wildland-Urban Interface (ICLF) project: the more theoretical ignitions study and the more practical FlamMap Interpreter development. Both have been valuable to the project, but the ignitions study stands out as having a more direct contribution to the broader field of wildfire science and I summarize it first in this chapter.

Wildfire Ignitions Study

The wildfire ignitions study examined the statistical relationships between wildfire ignitions and explanatory values related to human presence and activity as well as land use/land cover classifications. Three different groups of ignitions were considered: all ignitions, human ignitions, and lightning ignitions.

Summary

Results from the ignitions study reinforce the importance of considering human and lightning ignitions separately. Descriptive statistics and logistic models both showed that the effects of many explanatory variables were not just different but were qualitatively opposite for these two classes of ignitions. These effects were apparent for both human factors – for example human ignitions were more likely closer to roads and lightning ignitions further from roads – and vegetation factors – human ignitions less likely in older conifer forest (< 200 years old) and lightning ignitions much more likely. The probability surfaces derived from the models showed how considering human and

lightning ignitions together can lead to an explanatory model that is less precise and less accurate than those models which consider the two classes of ignitions separately.

The importance of roads for ignition probability was clear in the results of both the CART and logistic models. Also clear was the larger relative effect of major roads as compared to that of minor roads. While class of road is a good indicator of the level of human presence – major roads being generally more well traveled than minor roads – one can infer that level of human presence and activity near roads is also important from the fact that ignition probabilities did not fall suddenly with distance from roads, but tapered off. Along with class of road, other factors indicating levels of associated human activity bear examination, for instance scenic routes where travelers make frequent stops, or roads through popular hiking areas. This also points to the consideration of other factors that might draw human activity, such as parks, campgrounds, and scenic destinations.

The probability surface maps produced by the CART model and its logistic counterpart illustrate the categorical nature of CART results in contrast to the continuous nature of logistic results. In some cases a measure of accuracy, such as the misclassification rate, may be used to consider one of these methods "better" than the other. But ultimately, the utility of model results must be considered. The simpler nature of CART results may be more appropriate in some contexts, for instance a zonal ignitions risk map that might be distributed to the general public, while the more nuanced nature of logistic results might better serve fire managers and land planners attempting to design specific plans of action.

Methods from the ignitions study made contributions beyond the scope of the ICLF project. The two-phase process used to generate logistic models provided a novel

way to reduce the number of explanatory variables for logistic model development with large numbers of potential explanatory variables. In a situation where the number of variables and size of the datasets combine to make an all-possible-subsets approach to model development computationally intractable, this method can be used. The quasi-R^2 statistic developed to evaluate the logistic model results provides a new method to evaluate the predictive power of logistic regression models.

Study Weakness

The exclusion of topographic factors, especially elevation, was a weakness of the ignitions study. Visual inspection of ignitions data plotted on a digital elevation model of the study area suggested a strong correlation with elevation and ridgelines for lightning ignitions. The conclusion from the study is that the human-related explanatory variables correlated with lightning ignitions are surrogates for elevation. The possibility that vegetation effects are also ultimately driven by elevation or topography must be considered in the absence of topographic explanatory variables.

Future Study

Three aspects of the ignitions study deserve consideration for further study. First, adding topographic factors to explanatory variables in the current study might result in a more accurate model for lightning ignitions, and would likely produce a model less dependent on human factors. Reevaluating this study's data with topographic factors included is planned.

Second, considering classes of roads separately showed that discerning level of human activity can lead to more precision in model results. Future studies might examine factors relating to the amount of travel along transportation corridors such as traffic data

or population levels along or near the ends of corridor sections. Other indicators of human presence, such as permits for trails and campgrounds might prove useful for backcountry areas.

Finally, logistic regression is a popular statistical technique but does not have an equivalent to the coefficient of determination (commonly known as R^2). Further evaluation of the quasi-R^2 statistic would determine its potential as an equivalent to or stand-in for the coefficient of determination to quantify the predictive value of logistic models.

The FlamMap Interpreter

The FlamMap Interpreter is the software plug-in module developed as this thesis. It creates the necessary interface between Envision and the FlamMap dynamic link library, which in turn provides spatially explicit wildfire modeling for the ICLF coupled systems model.

Development

As part of the ICLF coupled systems model, the FlamMap Interpreter provides key functionality needed to integrate not only spatially explicit wildfire modeling, but also the influences of a changing climate and human activity on the occurrence and extent of wildfire. The logistic modeling results from the ignitions probability study provide the central equation used to create model ignition locations.

In a software development context, the FlamMap Interpreter illustrates a successful software implementation for linking models with differing data requirements and differing representations of the same landscape. Data that neither model was able to generate, notably fire weather and duration parameters, were precomputed and provided

via an input file. Where possible, data was converted between the domains of Envision and FlamMap. This was done with vegetation data definitions using a predefined lookup table to translate Envision's vegetation states into the vegetation-related landscape characteristics used by FlamMap. This was also done for the spatial distribution of vegetation states and flame lengths using the PolyGridLookups algorithm. These techniques provide a template that developers can follow to incorporate other modeling components into the Envision software framework, or to link models in other projects.

Software Deployment

The success of software developed for research purposes will be determined in part by its use after it is completed. The ICLF project will be using the FlamMap Interpreter for in the coupled systems model. Currently, it is being tested along with its input data for the ICLF project. The FlamMap Interpreter will also be used for a project similar to the ICLF project but for a study area located in central Oregon.

The PolyGridLookups component of the FlamMap Interpreter is being used in at least two additional Envision-based projects. One project utilizes it as part of a habitat model, and another to interface with gridded data from the MC1 dynamic global vegetation model. PolyGridLookups is also being used in Envision by general-purpose code that interfaces with gridded data conforming to a standard format.

Final Remarks

As is the case in many regions over the globe, the Willamette Valley is facing a future that includes climate change and rapid population growth. By utilizing data and components from a variety of sources, the ICLF project seeks to model landscape changes over a portion of the Willamette Valley using a variety of projected future

scenarios. The relationships between roads, population and ignitions found in the ignitions study and implemented in the FlamMap interpreter provides a key coupling between human presence and wildfire. The FlamMap Interpreter is also key to a further feedback implemented in Envision that couples human decisions, land use, vegetation changes, fire regime, and wildfire. As the ICLF project moves from development to production modeling, a more detailed characterization of these interactions will emerge. Possibly as important as those modeling outcomes will be the proof of concept that through science and software engineering alternative futures modeling can link climate projections to vegetation, wildfire, and human activity to model future development and its ecological effects.

APPENDIX A

THE ENVISION FLAMMAP INTERPRETER USER'S GUIDE

The Envision FlamMap Interpreter User's Guide

by

Tim Sheehan

November 11, 2011

Audience

This guide is intended for Envision users who wish to incorporate fire modeling into their Envision scenarios. It assumes that users are familiar with both the Envision software package (Envision) and FlamMap (Finney, 2006; Missoula Fire Sciences Laboratory, 2006). It also assumes that users have a working knowledge of the csv file format, and the ability to edit and save csv files as well as ASCII files.

Guide Organization

This guide is divided into the following chapters:

- **Overview of Structure and Files**

  An overview of how the FlamMap Interpreter works, what files are needed by different functions in the Interpreter, and what files must be provided by the user.

- **File Descriptions**

  A detailed description of the content and format of each user-provided file.

- **Factors Affecting Model Fires**

  A brief description of how various inputs affect the number of ignitions, model fire intensity, and model fire spread.

- **Example Run**

  A walk-through of setting up a hypothetical Envision run that uses the FlamMap Interpreter.

Overview of Structure and Files

Introduction

The FlamMap Interpreter is a plug-in software module that allows the Envision computer program to run and exchange input and output data with the FlamMap fire-modeling program. It is implemented as a Dynamic Link Library (DLL) and, like Envision, is designed to work only with the Microsoft Windows operating system. Envision makes function calls to the FlamMap Interpreter, which reads input, prepares data for output, calls the FlamMap DLL, and returns data to Envision (Figure 1).

> **Note: In this guide, an *Envision session* refers to an execution of Envision from the time the user invokes Envision until the time the user exits Envision. An *Envision run* refers to the execution of a series of time steps associated with a single *Envision scenario*.**

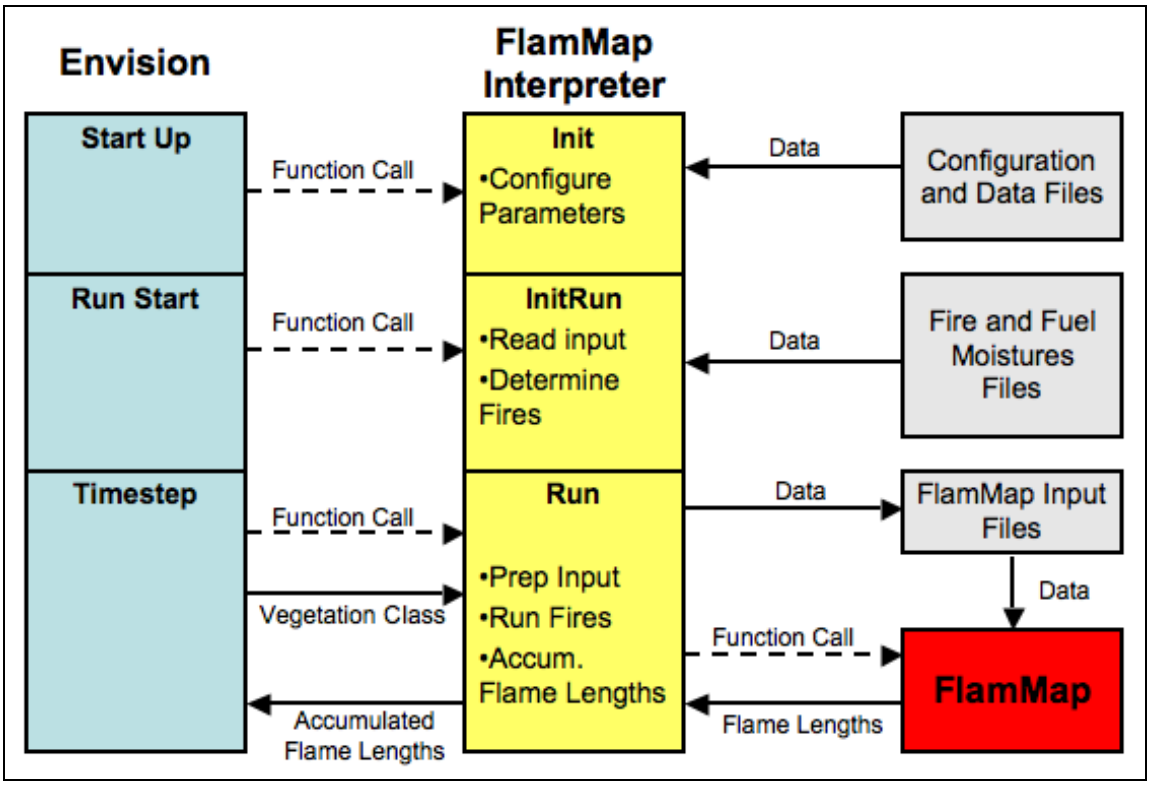

Figure 1. Relationship between Envision, the FlamMap Interpreter, and the FlamMap DLL. Software components are shown in color, files in grey.

Session Initialization

The *Init* function in the FlamMap Interpreter handles all the initialization necessary for one Envision session (Figure 2). Envision passes the name of the initialization file to the Init function. *Init* reads the file and sets the values of parameters that guide the execution of the FlamMap Interpreter. These parameters include the names and locations of other files used by the FlamMap Interpreter.

The starting FlamMap model landscape file (LCP file) should cover the same spatial extent as used by Envision. If both do not match exactly, results may still be produced but will be invalid. This file is used as a starting template for the creation of the LCP files that FlamMap uses during the course of an Envision run and is also used to create the polygrid data structure (described below) if no polygrid file is specified in the FlamMap Interpreter initialization file. The starting LCP file must have accurate values for slope, aspect, and elevation. The vegetation-related values for fuel model, canopy cover, stand height, crown base height, and canopy bulk density may contain any value, as these values will be reset by the FlamMap Interpreter during an Envision run.

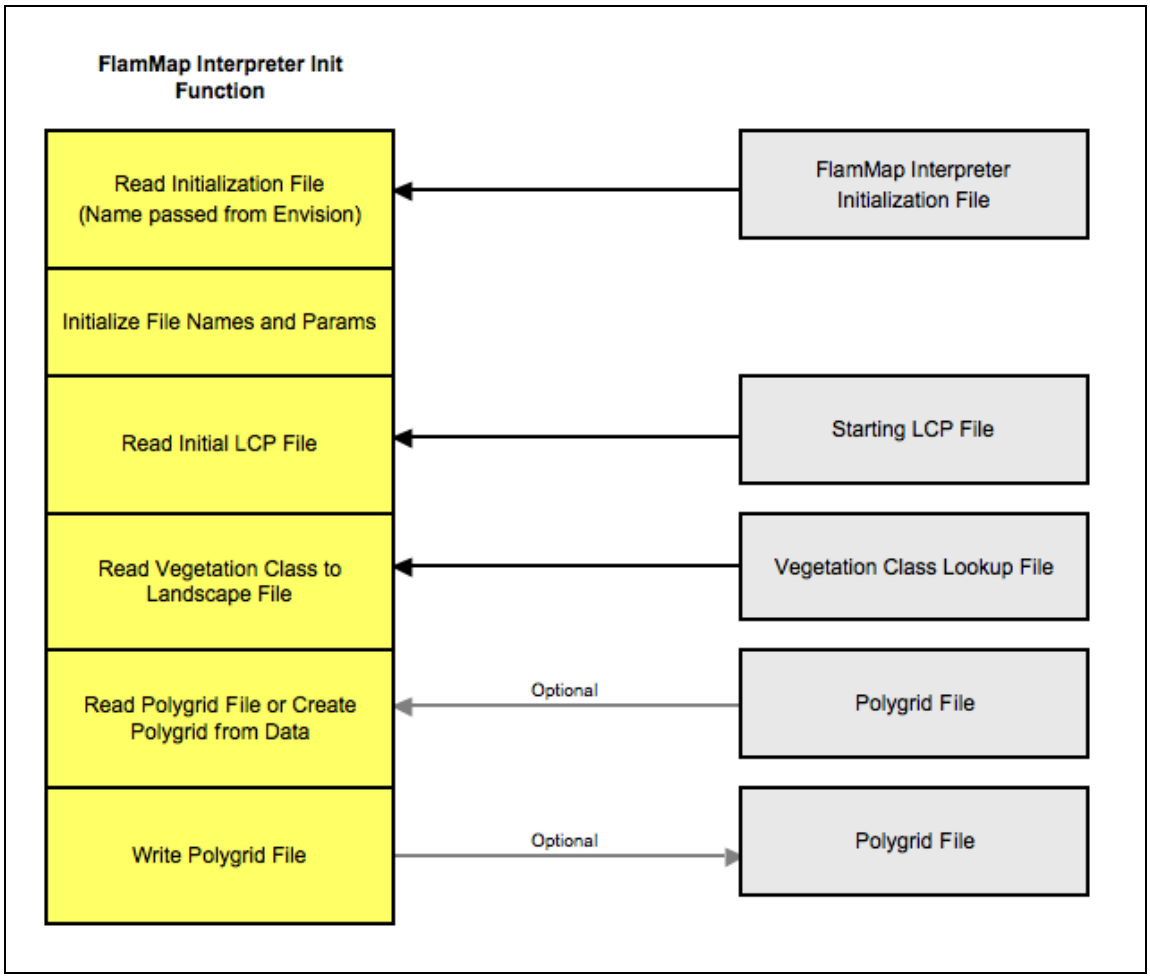

Figure 2. Functionality diagram of FlamMap Interpreter *Init* function.

The vegetation class lookup file provides a one-to-one mapping between the Envision's vegetation class values and the vegetation-related LCP values used by FlamMap. The data structure for the translation is created by *Init* and is used by the FlamMap Interpreter during an Envision run.

The term polygrid refers to the translation of data between Envision's polygon format and FlamMap's grid format. Envision "divides" a landscape into non-overlapping polygons termed integrated decision units (IDUs). Each polygon may have a different shape and size. FlamMap, on the other hand, divides a landscape into a regular grid of identically sized, square cells. In order for FlamMap to use data from Envision, values must be translated from polygon space into grid space. Likewise, for Envision to use data from FlamMap, the data must be translated from grid space into polygon space. An array-based algorithm called "polygrid lookup" or simply polygrid was developed to do this. The array and associated parameters used by the algorithm are unique to any pair of Envision and FlamMap landscapes. If Envision's polygon definitions or FlamMap's grid cell size are changed, a new polygrid must be defined.

Generating a polygrid can consume substantial computational time. Once it is generated for a landscape, it is more efficient to save the polygrid to a file and read that

file for subsequent Envision sessions. Options for creating, reading, and writing polygrid files are specified in the FlamMap Interpreter initialization file. Polygrid files are stored in binary format and should not be modified.

Run Initialization

An Envision run performs computations over a series of model years. The *InitRun* function in the FlamMap Interpreter handles all the initialization necessary for one Envision run (Figure 3).

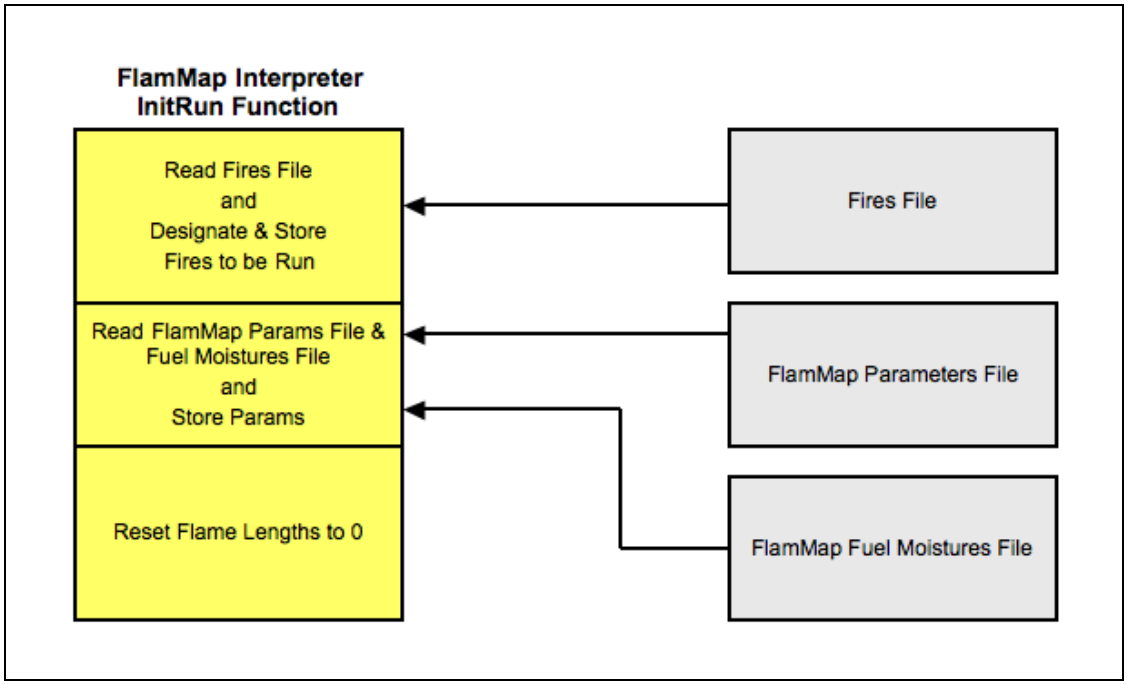

Figure 3. Functionality diagram of FlamMap Interpreter *InitRun* function.

The fires file contains a list of potential fires for each model year. Each fire has a year, a Julian date, a probability, and parameters for wind speed, wind direction, and burn duration. Using a Monte-Carlo technique, *InitRun* determines which fires will be modeled and saves a list of these fires in a data structure accessed during the Envision run.

The FlamMap parameter file contains options used by the FlamMap DLL that would normally be specified interactively with the standalone version of FlamMap. These include wind speed, wind direction, options for the minimum travel time (MTT) algorithm, and other options. For each model fire, wind speed, wind direction, and the time of simulation (MTT_SIM_TIME) are changed by the FlamMap Interpreter before the FlamMap DLL is invoked. The other parameters from the FlamMap parameter file are used without change.

The FlamMap fuels moisture file contains a list of fuel models and their fuel moistures used by the FlamMap DLL. This data is not changed when it is read in. It is combined with FlamMap parameters and written to the FlamMap input parameters file during the Run function.

Run

The FlamMap Interpreter *Run* function is called once per Envision time step. *Run* creates a single LCP file used by each fire modeled during a single Envision time step. *Run* then steps through each model fire for the Envision time step, creating an ignition file and a FlamMap input parameters file, and then running the FlamMap DLL. After executing the FlamMap DLL, *Run* adds the flame lengths to a data structure that stores the maximum flame length attained for every FlamMap grid cell during the Envision time step (Figure 4).

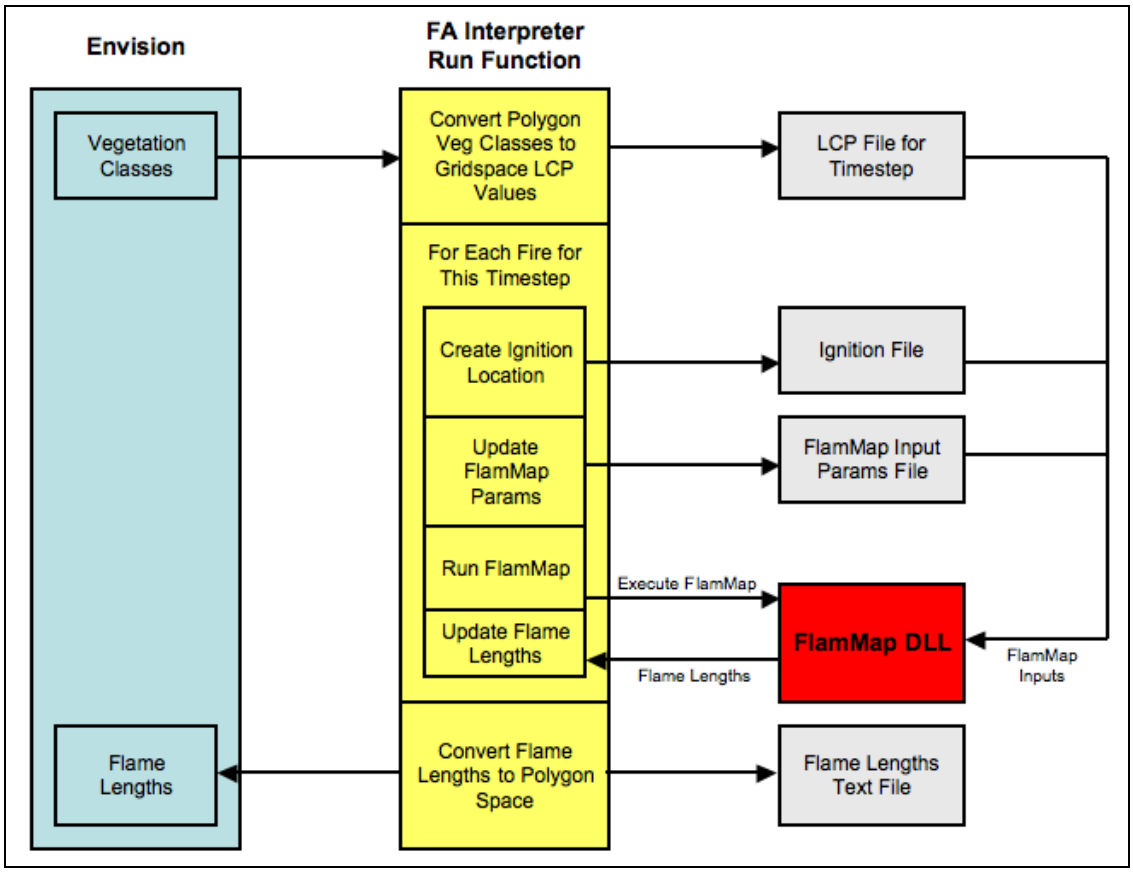

Figure 4. Functionality diagram of FlamMap Interpreter *Run* function. Software components are shown in color, files in grey.

The LCP file for one Envision time step contains the FlamMap landscape parameters used by each run of the FlamMap DLL during the time step. The file contains eight layers. The three layers describing topography (slope, aspect, and elevation) do not change during an Envision session. The five layers related to vegetation and fuels characteristics (fuel model, stand height, canopy cover, crown base height, and crown bulk density) are initialized every time step by an algorithm using the vegetation class obtained from Envision and the conversion table read from the vegetation class lookup file during the *Init* routine. .

For each model fire, a probabilistic algorithm generates an ignition point location. This algorithm uses values for distance to major road, distance to minor road, and

population density obtained at runtime from Envision IDU attributes to generate an ignitions probability map. A Monte Carlo draw determines the ignition point location that is then written to the ignition file.

The names of the input files for the run of FlamMap along with parameters from the FlamMap parameters file read by the *InitRun* function are written to the FlamMap input parameters file. (Note that the FlamMap parameters file (Figure 3) is not the same as the FlamMap input parameters file (Figure 4).

After all of these files have been written, *Run* executes the FlamMap DLL. When the FlamMap DLL finishes its execution, *Run* reads FlamMap's flame lengths for the entire landscape. For any grid cell on the FlamMap landscape, if a calculated flame length from the run is greater than the currently recorded flame length, the new flame length replaces the currently recorded flame length. In other words, for an Envision time step, the maximum flame length for each grid cell on the FlamMap model landscape is recorded.

After all the fires for an Envision time step have been run, the maximum flame lengths are converted into polygon space. For each IDU polygon, the weighted mean of flame lengths for overlapping FlamMap grid cells is written to a data structure in Envision. These same values are also output to disk as an ASCII file.

File Location

The name and path of the FlamMap Interpreter initialization file are specified in the Envision `.envx` file. All other files must reside in one directory whose name is specified in the FlamMap Interpreter initialization file.

File Descriptions

FlamMap Interpreter Initialization File

The FlamMap Interpreter initialization file (File Listing 1) is read by the FlamMap Interpreter *Init* function. The parameters in this file determine the names of all other files used by the Interpreter as well as options needed to direct how the interpreter runs. The user should pay special attention to the options dealing with units. If the specified units do not match the units in the vegetation class lookup file, the FlamMap DLL will run without error or warning but the results will be invalid.

File Listing 1. Example FlamMap Interpreter initialization file.

```
# Initialization File For Running FlamMap from Envision
#
# Pound sign in the first column indicates comment
# Blank lines ignored
#
# Lines with an invalid Field Name or Value will
# cause an error and program exit
#
# Strings must be quoted if they contain spaces

# Paths for input and output files
```

```
WorkingPath:                    C:\Envision\StudyAreas\Lebanon\FlamMap\

# Logging for profiling, etc.

DoLogging:               false
LogFName:                FlamMapAPLog.txt

# Options and filenames for Poly/Grid Lookup Table
# if not read from file, then generated

PolyGridReadFromFile:    true
PolyGridSaveToFile:      false
PolyGridReadFName:       PolyGridLkUp90Meter.pgl
PolyGridSaveFName:       PolyGridLkUp90Meter.pgl

# Files and options for Veg Class to LCP conversion

VegClassLCPLookupFName:  vegclass_fire_vddt_transitions.csv

# Units used in the VegClassLCPLookup file
# Height:
#    1 = meters, 2 = feet, 3 = m x 10, 4 = ft x 10
# Bulk Density:
#    1 = kg/m^3, 2 = lb/ft^3, 3 = kg/m^3 x 100, 4 = lb/ft^3 x 1000
# Canopy Cover:
#    0 = categories (0-4), 1 = percent

VegClassHeightUnits:           1
VegClassBulkDensUnits:         1
VegClassCanopyCoverUnits:      1

# Files associated with FlamMap runs

# RunParams are the run parameters needed for each run of
# FlamMap, these are used to build the .input file.

FiresFName:              LebanonFires.csv
StartingLCPFName:        Lebanon90Meter.lcp
StartingFAParamsFName:   FlamMapInputParams.txt
FuelMoisturesFName:      FlamMapFuelMoistures.fms

#   root names for files which are unique to each
#   run of FlamMap

LCPFNameRoot:             Lebanon90Meter
InputFNameRoot:           FlamMapInput
IgnitionFNameRoot:        LebanonIgnitions
```

The syntax used in the Interpreter init file is as follows:

- A comment line begins with a pound sign (#):

  ```
  # This is a comment line
  ```

- A parameter name is terminated by a colon and must be one of:

```
WorkingPath:
DoLogging:
LogFName:
PolyGridReadFromFile:
PolyGridSaveToFile:
PolyGridReadFName:
PolyGridSaveFName:
VegClassLCPLookupFName:
VegClassHeightUnits:
VegClassBulkDensUnits:
VegClassCanopyCoverUnits:
FiresFName:
StartingLCPFName:
StartingFAParamsFName:
FuelMoisturesFName:
LCPFNameRoot:
InputFNameRoot:
IgnitionFNameRoot:
```

- Any parameter value with a space in it must be quoted (using spaces in name parameters is discouraged):

```
VegClassLCPLookupFName:  "My Spaced Name"
```

Using valid values (Table 1) for parameters is important to insure that the FlamMap Interpreter executes in the desired manner. Each parameter should only be present one time in the initialization file. If it appears multiple times, the value of the last occurrence will be used.

Table 1. FlamMap Interpreter parameters specified in the Interpreter's initialization file.

Parameter Name	**Valid Values**	**Description**
`WorkingPath:`	Valid file path specification	The file path (i.e. directory) from which all files are read and to which all files are written.
`DoLogging:`	**`true`** **`false`**	Whether to log certain debugging information to a log file. Used by developers.
`LogFName:`	Valid file name (without path)	File to which log information is written. Used by developers.
`PolyGridReadFromFile:`	**`true`** **`false`**	Whether to read polygrid data from a file (**`true`**); or to generate polygrid data from scratch (**`false`**).
`PolyGridSaveToFile:`	**`true`** **`false`**	Whether to write polygrid data to a file (**`true`**); or not (**`false`**).
`PolyGridReadFName:`	Valid file name (without path)	Name of polygrid file to read.
`PolyGridSaveFName:`	Valid file name (without path)	Name of polygrid file to write.
`VegClassLCPLookupFName:`	Valid file name (without path)	Name of file used to convert vegetation class values into landscape values.

VegClassHeightUnits:	**1**, **2**, **3**, **4**	**1** = meters, **2** = feet, **3** = m x 10, **4** = ft x 10
VegClassBulkDensUnits:	**1**, **2**, **3**, **4**	**1** = kg/m^3, 2 = lb/ft^3, 3 = kg/m^3 x 100, **4** = lb/ft^3 x 1000
VegClassCoverUnits	**0**, **1**	**0** = categories (0-4), **1** = percent
FiresFName:	Valid file name (without path)	Name of file containing candidate fires to run.
StartingLCPFName:	Valid file name (without path)	Name of LCP file used as a base to build LCP files during Interpreter run.
StartingFAParamsFName:	Valid file name (without path)	Name of file containing FlamMap parameters used by FlamMap runs.
FuelMoisturesFName:	Valid file name (without path)	Name of fuel moistures file used by all FlamMap runs.
LCPFNameRoot:	Valid file base name (without path or extension)	Base name for all LCP files written by the FlamMap Interpreter and used by the FlamMap DLL.
InputFNameRoot:	Valid file base name (without path or extenstion)	Base name for all parameter files written by the FlamMap Interpreter and used by the FlamMap DLL.
IgnitionFNameRoot:	Valid file base name (without path or extenstion)	Base name for all ignition files write by the FlamMap Interpreter and used by the FlamMap DLL.

Polygrid lookup file

A polygrid lookup file can only be generated within the FlamMap Interpreter. FlamMap can either read the file from a saved version, or generate one from scratch. For large landscapes, generation is computationally intensive. A polygrid file can be saved by specifying `true` for the `PolyGridSaveToFile`: option and a valid file name for the `PolyGridSaveFName`: option in the FlamMap Interpreter initialization file.

LCP (Landscape) file

An LCP is a binary file used by the FlamMap DLL. The FlamMap Interpreter reads a starting version of this file to get the number of rows and columns in the FlamMap grid and to create a starting template for building the LCP files used by the FlamMap DLL during the course of a run.

> **NOTE: The slope, aspect, and elevation values in the starting LCP file are never changed and must be accurate for the landscape.**
>
> **NOTE: The spatial extent of the starting LCP file must match the spatial extent of the Envision landscape.**

There are various ways to create the starting LCP file. One of the most straightforward is by using the utility lcpmake.exe (part of the FARSITE download package at http://www.firemodels.org/index.php/farsite-software/farsite-downloads), which takes a set of Arc Grid ASCII files, each ASCII file representing an LCP layer, and converts them into an LCP file.

Fires file

The fires file (File Listing 2) is a csv (comma separated variables) file that lists possible fires by fire year. Its first row must contain column names. Subsequent rows contain fields identifying year, fire probability, day, and fire weather parameters (Table 2). For any line entry in this file that is used to model a fire, the values for `burn period` (in minutes), `wind mph`, and `azimuth` are included in the FlamMap Input Parameters file (Figure 4) for that model fire.

File Listing 2. Sample rows from a FlamMap Interpreter fires file as they would appear in a Microsoft Excel spreadsheet.

year	prob	julian	burn period	wind mph	azimuth
2010	0.01	120	1	20	270
2010	0.01	121	1	20	270
2010	0.01	122	1	20	270
2010	0.02	123	1	20	270
2010	0.02	124	1	20	270
2010	0.02	125	1	20	270
2010	0.02	126	1	20	270
2010	0.02	127	1	20	270
2010	0.03	128	1	20	270
2010	0.03	129	1	20	270
2010	0.03	130	1	20	270
2010	0.02	131	1	20	270
2010	0.03	132	1	20	270
2010	0.03	133	13.51	20	270
2010	0.03	134	1	20	270
2010	0.04	135	28.02	20	270
2010	0.03	136	525.41	20	270
2010	0.03	137	84.07	20	270
2010	0.02	138	0	20	270
2010	0.03	139	30.02	20	270

Table 2. Columns in the FlamMap Interpreter fires file.

Col #	Col Name	Valid Values	Description
1	Year	0 to number of run years	The model year of the run. This is not the same as a calendar year.
2	Probability	0.00 to 1.00	The probability that the fire described in the row will be run.
3	Julian Date	1 to 365	The Julian date of the year.
4	Burn Period	0 to max fire duration	The length of time in minutes the fire will be run. Units are minutes.
5	Wind Speed	0 to max wind speed	The wind speed.
6	Wind Azimuth	0 to 360	The wind direction.

Fuel moistures file

The fuel moistures file (File Listing 3) is an ASCII file that is virtually identical to that used by the standalone version of FlamMap, the difference being the first line of the file. The version for the DLL file requires the first line identify the number of lines in the file. The first column in the file is the fuel model, the next three are fuel moistures for 1-hr, 10-hr, and 100-hr dead fuel moistures, and the final two are fuel moistures for live herbaceous and live woody fuel. The contents of this file are used in part to create the FlamMap Input Parameters file (Figure 4).

File Listing 3. A portion of an example fuel moistures file.

```
FUEL_MOISTURES_DATA: 257
0 3 4 6 125 125
1 3 4 6 125 125
2 3 4 5 120 100
3 3 4 6 125 125
    . . .
[Lines deleted from this example]
    . . .
252 3 4 6 125 125
253 3 4 6 125 125
254 3 4 6 125 125
255 3 4 6 125 125
256 3 4 6 125 125
```

FlamMap parameters file

The FlamMap parameters file (File Listing 4) is an ASCII file containing parameters used by the FlamMap DLL. Parameters `WIND_SPEED`, `WIND_DIRECTION`, and `MTT_SIM_TIME` are updated by the FlamMap Interpreter. The other parameters remain unchanged. The contents of this file are used in part to create the FlamMap Input Parameters file (Figure 4).

File Listing 4. Sample FlamMap parameters file.

```
InputFileHeader: ShortTerm-Inputs-File-Version-1

WIND_SPEED: 6
WIND_DIRECTION: 299
GRIDDED_WINDS_GENERATE: No
GRIDDED_WINDS_RESOLUTION: 200
FOLIAR_MOISTURE_CONTENT: 100
CROWN_FIRE_METHOD: Finney
NUMBER_PROCESSORS: 1
SPREAD_DIRECTION_FROM_NORTH: 0

MTT_RESOLUTION: 30
MTT_SIM_TIME: 200
MTT_TRAVEL_PATH_INTERVAL: 500
MTT_SPOT_PROBABILITY: 0.0
```

```
# Request some flammap output as well...
FLAMELENGTH: true
```

Vegetation class lookup file

The vegetation class lookup file (File Listing 5) is a csv (comma separated variable) file that matches a vegetation class to its associated LCP values. The file commonly used for this is dual purposed and includes flame length threshold values used by another Envision plug-in to interpret the effects of different flame lengths returned from FlamMap on Envision vegetation states. This does not affect the FlamMap Interpreter as long as the first 7 columns contain the expected data. The first row of the vegetation class lookup file must be column names. The `VEGCLASS` and `VARIANT` columns correspond to the vegetation class and vegetation class variant values that are stored by Envision for each IDU. The remaining columns correspond to LCP values for fuel model, canopy cover, canopy height, canopy base height, and canopy bulk density.

File Listing 5. Sample rows from a FlamMap Interpreter vegetation class lookup file as they would appear in a Microsoft Excel spreadsheet. There may be more columns than shown in this example, as long as the first seven columns match the seven columns shown here.

VEGCLASS	VARIANT	LCP_FUEL_MODEL	LCP_CNPY_COV	LCP_CNPY_HT	LCP_CNPY_BS_HT	LCP_CNPY_BLK_DNS
1	1	91	0	0	0	0
2	1	91	0	0	0	0
3	1	91	0	0	0	0
4	1	91	0	0	0	0
92	1	161	55	18	6	0.03
93	1	181	50	8	1	0.02
95	1	183	65	60	15	0.09
200	1	104	0	1	-999	0
201	1	104	9	12	8	0.00301741
202	1	104	11	21	12	0.00206693
203	1	104	12	29	17	0.00191507
204	1	104	5	48	-999	0
210	1	141	47	37	15	0.0199068

Factors Affecting Model Fires

For a single model year, the number, intensity, and spread of modeled fires all contribute to the flame length data returned to the Envision model landscape. A qualitative understanding of which inputs affect the different aspects of a model fire year can help the user make adjustments for desired outcomes. This section provides an overview of how adjusting certain inputs can change the number, intensity, and spread of model fires, but it is incumbent on the user to understand the implications of changes to input and to insure that inputs are scientifically well founded.

Number of Ignitions

Two input factors influence the number of fires in a given model year; both are characteristics of the fires file. The first is the number of Julian days for which there can be a fire. All things being equal, the greater the number of potential fire days, the greater the number of fires that will be selected to run.

The second factor is the probability of a fire. The higher a potential fire's probability, the more likely it is to be run. To guarantee a potential fire be run, its probability can be set to 1. A fire year with overall higher probability values for potential fires will generally have more fire occurrences.

Intensity

Modeled fire intensity is affected by fuel moisture and wind. Lower fuel moistures lead to a more intense fire. These can be altered in the fuel moistures file. Wind also affects burn intensity, so increasing the wind speed in the fires file will generally lead to more intense model fires.

Spread

Three factors influence the spread of model fires: fuel moisture, wind, and burn period. Lower fuel moistures lead to a faster burning fire. These can be adjusted in the fuel moistures file. Higher winds spread fire faster. Winds can be adjusted in the fire file. The longer a fire burns, the more it will spread. Burn period, set in the fire file, can be increased in order to increase the area burned.

Example Run

Setting up a directory

The first step is to choose or create the directory where all files associated with the FlamMap Interpreter will reside. For this example we will use this directory:

```
C:\MyEnvisionData\Lebanon\FlamMap
```

With the exception of the initialization file, all files used by the FlamMap Interpreter must be in this directory. In this example, we include the initialization file in the same directory. The files in this directory are:

```
FlamMapFuelMoistures.fms
FlamMapInputParams.txt
FlamMapInterpreter.ini
Lebanon90Meter.lcp
LebanonFires.csv
vegclass_fire_vddt_transitions.csv
```

Each of these is one of the input files described earlier in this guide:

- `FlamMapFuelMoistures.fms` is the FlamMap DLL fuels moisture file.

- `FlamMapInputParams.txt` is the FlamMap parameters file, containing the input parameters used by the FlamMap DLL.
- `FlamMapInterpreter.ini` is the FlamMap Interpreter initialization file.
- `Lebanon90Meter.lcp` is the starting LCP file.
- `LebanonFires.csv` is the fires file.
- `vegclass_fire_vddt_transitions.csv` is the vegetation class lookup file.

Note that there is no polygrid file in the directory. This means we will have to make sure that the FlamMap Interpreter creates one.

The FlamMap Interpreter initializtion file

The FlamMap Interpreter initialization file, `FlamMapInterpreter.ini`, should look something like this:

File Listing 6. Example FlamMap Interpreter initialization file.

```
# Initialization File For Running FlamMap from Envision
#
# Pound sign in the first column indicates comment
# Blank lines ignored
#
# Lines with an invalid Field Name or Value will
# cause an error and program exit
#
# Strings must be quoted if they contain spaces

# Paths for input and output files

WorkingPath:                    C:\MyEnvisionData\Lebanon\FlamMap

# Options and filenames for Poly/Grid Lookup Table
# if not read from file, then generated

PolyGridReadFromFile:    false
PolyGridSaveToFile:      true
PolyGridReadFName:       PolyGridLkUp90Meter.pgl
PolyGridSaveFName:       PolyGridLkUp90Meter.pgl

# Files and options for Veg Class to LCP conversion

VegClassLCPLookupFName:  vegclass_fire_vddt_transitions.csv

# Units used in the VegClassLCPLookup file
# Height:
#    1 = meters, 2 = feet, 3 = m x 10, 4 = ft x 10
# Bulk Density:
#    1 = kg/m^3, 2 = lb/ft^3, 3 = kg/m^3 x 100, 4 = lb/ft^3 x 1000
# Canopy Cover:
#    0 = categories (0-4), 1 = percent

VegClassHeightUnits:           1
```

```
VegClassBulkDensUnits:          1
VegClassCanopyCoverUnits:       1

# Files associated with FlamMap runs

# RunParams are the run parameters needed for each run of
# FlamMap, these are used to build the .input file.

FiresFName:               LebanonFires.csv
StartingLCPFName:         Lebanon90Meter.lcp
StartingFAParamsFName:    FlamMapInputParams.txt
FuelMoisturesFName:       FlamMapFuelMoistures.fms

# base names: used to build the names of files
# read by or written by the FlamMap DLL

LCPFNameRoot:              Lebanon90Meter
InputFNameRoot:            FlamMapInput
IgnitionFNameRoot:         LebanonIgnitions
```

NOTE: It is up to the user to specify the correct values for `VegClassHeightUnits:`, `VegClassBulkDensUnits:`, and `VegClassCanopyCoverUnits:`. If these do not match the units in the vegetation class lookup file, results will invalid.

The parameter values for `LCPFNameRoot`, `InitFNameRoot`, and `IgnitionFNameRoot` are used to create the filenames for input files used by the FlamMap DLL.

The .envx file

The Envision `.envx` file must be configured to run the FlamMap Interpreter and it must have the full path and name of the initialization file. For this example, we need to put the following in the `<autonomous processes>` section of the `.envx` file:

```
<autonomous_process
      name         ='FlamMap'
      path         ='FlamMapAP.dll'
      id           ='0'
      use          ='1'
      timing       ='0'
      freq         ='1'
      sandbox      ='0'
    fieldName      =''
    initInfo     =' C:\MyEnvisionData\Lebanon\FlamMap\FlamMapInterpreter.ini'

      />
```

Files produced by the FlamMap Interpreter

During an Envision run, the FlamMap Interpreter will create a number of files associated with FlamMap runs. The names of these files are built from the base names in the initialization file, the time step number (preceded by the letters "TS"), and where needed,

the Julian date of the model fire. Here are an example name and a description for each of these files:

- `Lebanon90Meter_TS0001.lcp`: The LCP file read by every run of FlamMap during time step 1. This is a binary file, but it can be accessed using the standalone version of FlamMap.
- `FlamMapInputTS0001_123.input`: The FlamMap input parameter file read by FlamMap to run the model fire on Julian date 123 of time step 1. This ASCII file is user-readable.
- `LebanonIgnitions_TS0001_123.dbf, LebanonIgnitions_TS0001_123.shp, LebanonIgnitions_TS0001_123.shx`: The three files making up the ignition location for the model fire on Julian date 123 of time step 1. These files may be used together as an ArcGIS shapefile and accessed using ArcMap.
- `FlameLength_TS0001.txt`: A user-readable ASCII text grid of maximum flame lengths produced during time step 1.

In this sample run, we specified that the polygrid was to be created and written to a file. This file, `PolyGridLkUp90Meter.pgl`, would have also been created, and available for reading by the FlamMap Interpreter in future Envision sessions.

References

Envision. Envision version 5: An analysis framework for policy-driven alternative futures analyses. Available at http://envision.bioe.orst.edu/.

Finney, Mark, 2006. An Overview of FlamMap Fire Modeling Capabilities. In Andrews, Patricia L.; Butler, Bret W., comps. 2006. *Fuels Management-How to Measure Success: Conference Proceedings*. 28-30 March 2006; Portland, OR. Proceedings RMRS-P-41. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station. p. 213-220. Available online at http://www.fs.fed.us/rm/pubs/rmrs_p041/rmrs_p041_213_220.pdf.

Missoula Fire Sciences Laboratory, 2006. FlamMap 3.0.0 for Windows (Computer software package). Available at http://www.firemodels.org/index.php/flammap-software/flammap-downloads.

APPENDIX B

THE ENVISION FLAMMAP INTERPRETER

PROGRAMMER'S GUIDE

The Envision FlamMap Interpreter Programmer's Guide

by

Tim Sheehan

November 11, 2011

Audience

This guide is intended for programmers who wish to modify, debug, or copy and use the C++ code for the Envision FlamMap Interpreter. It assumes that users are familiar with the Envision software package (Envision), FlamMap (Finney, 2006; Missoula Fire Sciences Laboratory, 2006), and the C++ programming language in the Microsoft Windows 7 environment, including Dynamic Link Library creation.

Guide Organization

This guide is divided into the following sections:

- **Introduction**

  Brief description of the FlamMap Interpreter and its purpose.

- **FlamMap Interpreter Description**

  A description of the FlamMap Interpreter, and its functional architecture..

- **Input and Output Files**

  Files used by and produced by the FlamMap Interpreter.

- **Code Walkthrough**

  A description of code logic. This walkthrough is should be used with the source files for the FlamMap Interpreter.

- **Polygrid Lookup Logic and Implementation**

  A detailed description of polygrid lookup and how it is implemented.

- **Conclusion**

  Brief thoughts on the FlamMap Interpreter and the transient nature of software documentation.

Introduction

The FlamMap Interpreter is a Dynamic Link Library (DLL) plug-in that allows the Envision computer program to run and exchange input and output data with the FlamMap fire-modeling program. Like Envision, it is designed to work only with the Microsoft Windows operating system. For a complete description of how to use the FlamMap Interpreter and the format of I/O files, see the *The Envision FlamMap Interpreter User's Guide*.

The main functional steps executed by the FlamMap are:

- Take vegetation state data from the Envision landscape and use this data to characterize the landscape in a form usable by the FlamMap DLL.
- From external files, read FlamMap DLL execution options and data describing fire and fuels parameters.
- Generate model fire ignition points on the landscape.
- Write files used by the FlamMap DLL
- Call the FlamMap DLL to execute the runs of model fires.
- Update the Envision delta array with flame lengths from the model fires so that changes in vegetation states can be generated on burned areas of the landscape.

FlamMap Interpreter Description

> **Note: In this guide, an *Envision session* refers to the use of Envision from the start of the executable until the executable is terminated. An *Envision run,* or *model run* refers to a single simulation run, comprising a number of time steps. Multiple Envision runs can take place during an Envision session.**

Figure 1 shows an overview of how Envision relates to the FlamMap Interpreter and how the Interpreter relates to input and output files. The highest level class in the FlamMap Interpreter, `FlamMapAP`, provides the four entry points methods for Envision: `Init()`, `InitRun()`, `Run()`, and `EndRun()`.

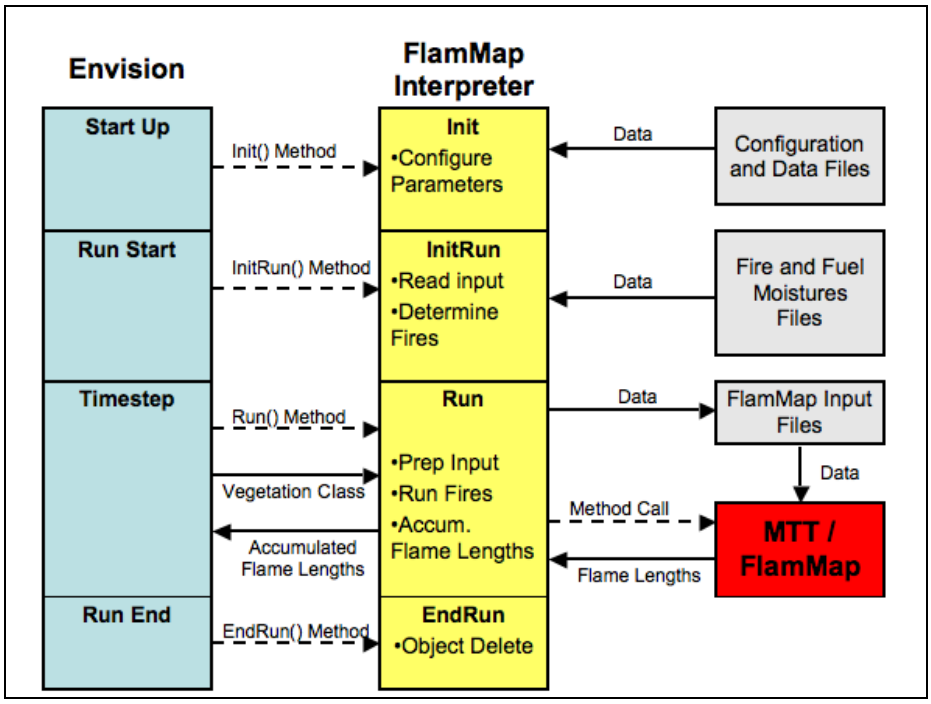

Figure 1. Relationship between Envision, the FlamMap Interpreter, and the FlamMap DLL.

During an Envision session, Envision calls `Init()` (Figure 2) one time only, as part of its startup. `Init()` reads configuration parameters, initializes variables, and instantiates classes that persist throughout the Envision session.

Envision calls `InitRun()` at the start of an Envision run (Figure 3). InitRun(), reads files, initializes variables, and instantiates classes used for the duration of a model run.

Envision calls `Run()` once per time step during an Envision run (Figure 4). `Run()` takes vegetation state data from Envision, prepares input files for the FlamMap DLL, runs the FlamMap DLL, and writes flame lengths to the Envision delta array.

Envision calls `EndRun()` at the end of a model run. EndRun() does some minor object management.

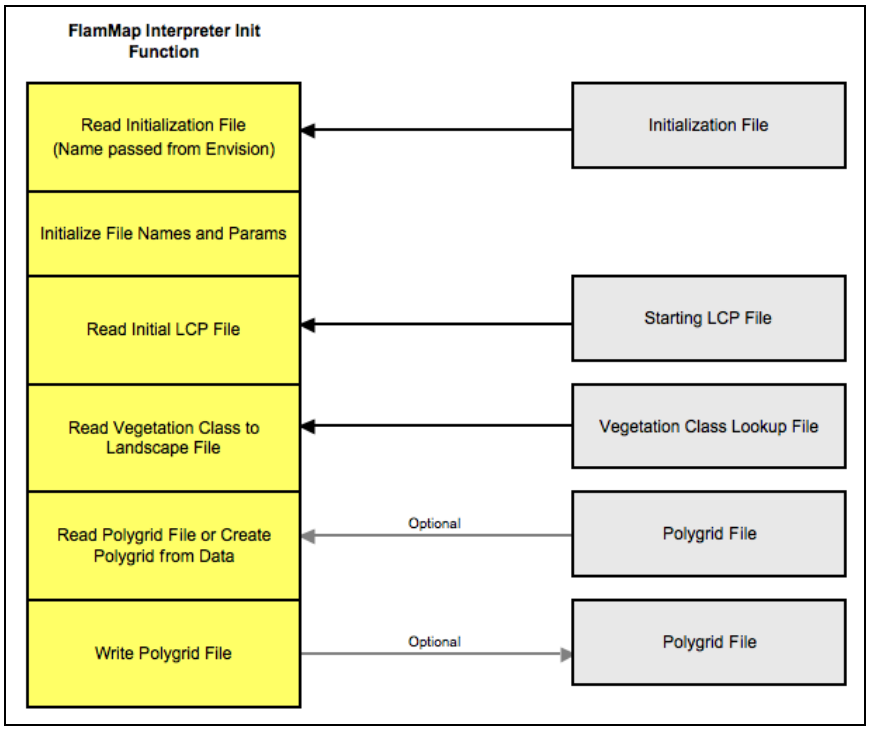

Figure 2. Functionality diagram of FlamMap Interpreter *Init* function.

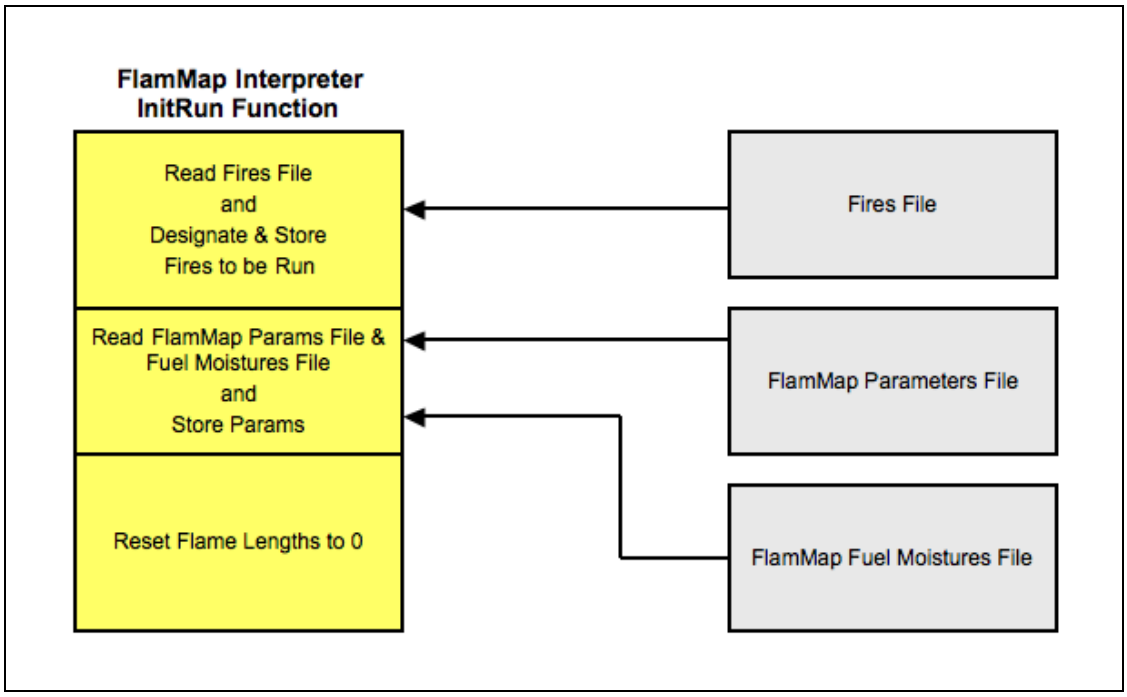

Figure 3. Functionality diagram of FlamMap Interpreter InitRun function.

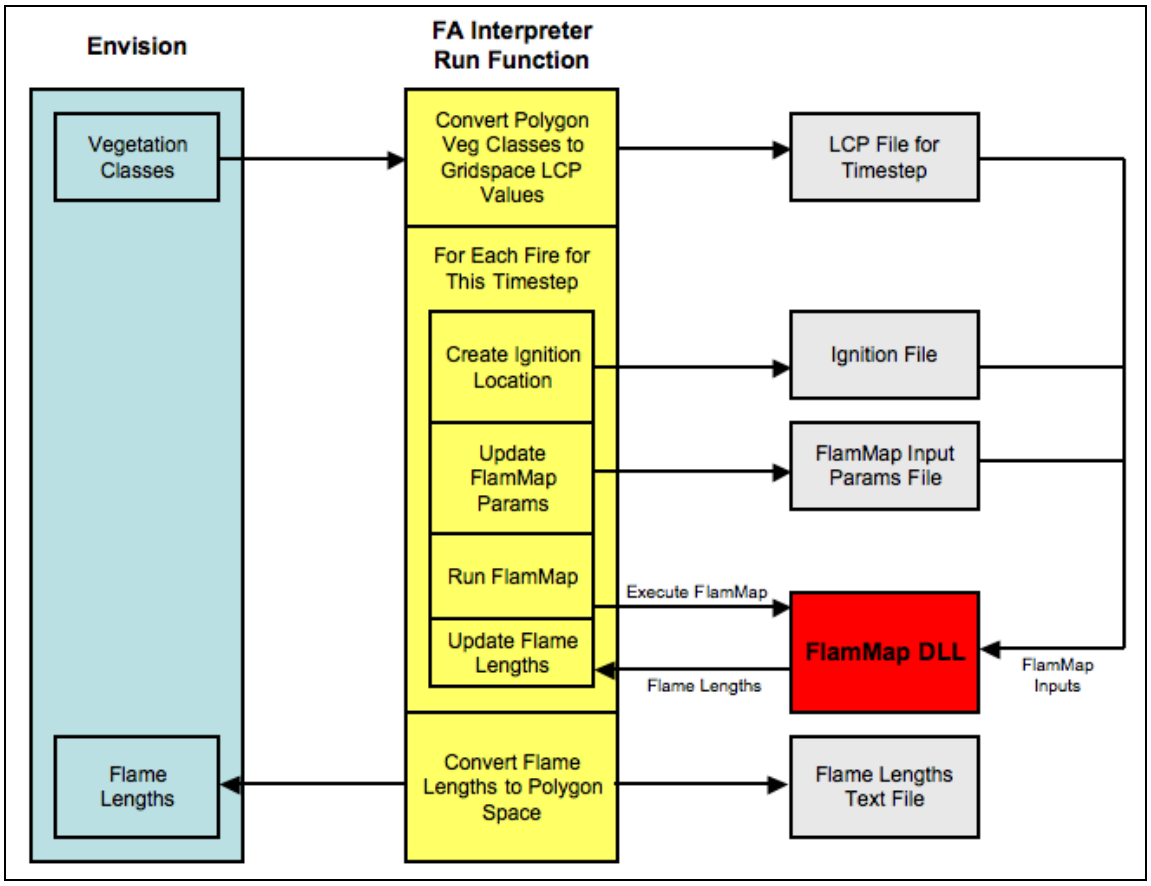

Figure 4. Functionality diagram of FlamMap Interpreter Run function.

Input and Ouptut Files

This section provides an overview of input and output files read and written by the FlamMap Interpreter. These are described in detail in The Envision *FlamMap Interpreter User's Guide*.

The files read or written by the `FlamMap::Init()` method are:

- The FlamMap Interpreter initialization file (input), containing the names and locations of files and directories used during the session as well as options for the run. The name of this file is passed to `FlamMapAP::Init()` by Envision.

- The starting LCP file (input) is specified in the FlamMap Interpreter initialization file. This file must contain an appropriate header as well as layer data for slope, aspect, and elevation. The data layers for vegetation characteristics can contain any values, as these are initialized by the FlamMap Interpreter by later processing.

- The vegetation class lookup file (input) is specified in the initialization file. It contains the data used to convert Envision vegetation states into the landscape attributes used to produce the LCP files during runs.

- The polygrid file (input) is specified in the initialization file. It is an optional file used to initialize the PolyGridLookups object used to translate spatial domains between Envision's polygon landscape representation and FlamMap's grid landscape representation.

- The polygrid file (output) is specified in the initialization file. It is an optional file that can be used to initialize the PolyGridLookups object in future runs.

The files read or written by the `FlamMap::InitRun()` method (Figure 3) are:

- The fires file (input). It contains a list of potential model fires for each model year along with parameters needed to run each of the model fires.

- The FlamMap parameters file (input). It contains input parameters that would be specified interactively with the standalone version of FlamMap. These are used by the FlamMap DLL.

- The FlamMap fuel moistures file (input). It contains the fuel moistures data used by the FlamMap DLL.

The files read or written by the `FlamMap::Run()` method (Figure 4) are:

- The LCP file, output once per Envision time step, input to the FlamMap DLL once per model fire. This is the landscape data used to model all fires during a single time step.

- Ignition file, output once per model fire, input to the FlamMap DLL once per model fire. This is the location of the model ignition point as a point shapefile in ArcGIS format.

- FlamMap input parameters file (note this is not the same as the FlamMap parameters file read by *InitRun()*), output once per model fire, input to the FlamMap DLL once per model fire. This file includes fuel moisture parameters, model fire weather parameters, model fire duration.

- Flame length text file, output once per time step. This is an ASCII file of the maximum flame length attained on each model grid point.

Source Code Walkthrough

This section discusses the more complex aspects of the source code for the FlamMap Interpreter. The reader should access the source files from the Envision source repository and refer to the source code as needed.

Figure 4 shows the static relationships among the classes used by the FlamMap Interpreter. All classes in Figure 4 were written as part of the Interpreter with the exception of `CMinTravelTime` which is the class that runs `FlamMap`.

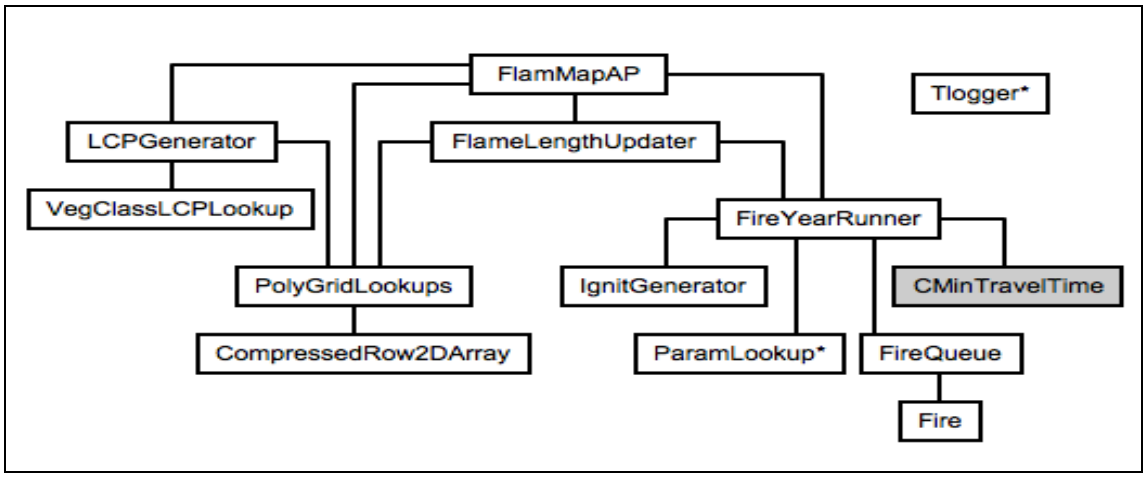

Figure 4. Static relationships among classes that are part of the FlamMap Interpreter. The class `CMinTravelTime` is the class for the FlamMap DLL and was not produced as part of the FlamMap Interpreter. Classes marked with an asterisk (*) are used with global scope in the FlamMap Interpreter.

Code Walkthrough

This code walkthrough should be used in conjunction with the FlamMapAP source files to provide an understanding of the code structure and to clarify the logic of those code segments that bear explanation. Much of the source code is not described in this walkthrough.

`FlamMapAP` is the class providing the entry point methods for the FlamMap Interpreter, and it is worth looking at the .h file in which it is declared.

`FlamMapAP.h`

In `FlamMapAP.h`,

```
#include <EnvExtension.h>
#include <Maplayer.h>
```

are includes for Envision classes. The remaining includes are for classes that are part of the FlamMap Interpreter.

```
typedef enum FlamMapRunState..
#define FLAMELENGTH       0 …
```

are needed for `CMinTravelTime`.

```
class FAGrid
```

is a grid map layer that is needed to generate the polygrid lookup.

```
class FlamMapAP
```

is the declaration for the FlamMapAP class. The four entry point methods are:

```
BOOL Init   ( EnvContext *pEnvContext, LPCTSTR initStr );
BOOL InitRun( EnvContext *pEnvContext, bool useInitSeed);
BOOL Run( EnvContext *pContext );
BOOL EndRun(EnvContext *pContext);
```

The EnvContext pointer in each of these prototypes provides a pointer that can be used access Envision's methods. The LPCTSTR in Init() is a string that Envision reads from its initialization file and passes through. It is the full pathname to the FlamMapAP initialization file.

Instantiations of LCPGenerator and FlameLengthUpdater are only needed by methods within the FlamMapAP class, so rather than declare them as pointers which can be easily passed to methods in other classes, they are instantiated at the time of declaration:

```
LCPGenerator
        m_LCPGenerator;
FlameLengthUpdater
        m_FlameLengthUpdater;
```

class CFlamMap is the class declaration for the FlamMap DLL. This could have been handled by including a .h file with the declaration, but was instead added to the FlamMapAP.h file.

Global Classes

There are two classes with instantiations that are used globally throughout the FlamMap Interpreter. The instantiations of these are done in FlamMapAp.cpp. The first of these is TLogger, which stands for *Tim's Logger*. Its instantiation is g_MyLog. This class provides methods useful for profiling and logging during code execution. It is used globally so that all logging information can be written to a single file. Its primary purpose is as a tool for debugging and performance tuning. It contains methods for turning logging on and off, so its use can be easily controlled. It is implemented completely in a .h file.

The second class with a global instantiation is ParamLookup. It, is instantiated as g_RunParams in FlamMapAP.cpp. This class allows for the dynamic creation of, update of, and access to variables associated with names. The FlamMap Interpreter deals with many parameters that are set in one part of the code and accessed in others. During development, and likely continuing with updates of the code, parameters have been added and removed. Rather than trying to track these parameters and modify variable declarations in any number of classes, g_RunParams is used as a means of parameter management.

BOOL FlamMapAP::Init(EnvContext *pEnvContext, LPCTSTR initStr)

Init() is the entry point called by Envision as part of its session startup. pEnvContext is a pointer that can be used access Envision's methods. initStr is a string that Envision reads from its initialization file and passes through. It is the full pathname to the FlamMapAP initialization file.

If any part of the FlamMap Interpreter fails, the desired behavior is for it to stop what it is doing and return a value of `false` to Envision. In several places throughout the code, macros are defined to do this. Two of these are defined in `Init()`: `FailAndReturn()` and `FailAndReturnOnBadStatus()`.

```
g_RunParams.ProcessInitFile(initStr)
```

sets the initialization parameters for the Envision session.

```
g_RunParams.SetBoolParam(_T("RunStatus"),true);
```

sets the global `RunStatus` to `true`.

`RunStatus` is maintained as a parameter in `g_RunParams` and may be set to `false` by a method that fails to complete successfully. `RunStatus` can be checked by other methods to determine if a failure has been triggered elsewhere in the code.

The code block

```
g_RunParams.AddReqStrParam(_T("WorkingPath"));
…
if(!g_RunParams.CheckAllReqParams()) {
FailAndReturn(" parameters missing from initialization file.");
} else {
       Report::InfoMsg(_T("  Required input parameters read."));
}
```

checks to make sure all the required parameters were read from the initialization file.

```
m_LCPGenerator.InitLCPValues(g_RunParams.GetStrParam(_T("StartingLCPFName")))
```

initializes the LCP file image in the `m_LCPGenerator` object using an initial LCP file. Values specifying the FlamMap grid are read from this file: `Rows`, `Cols`, and `CellDim`.

```
m_LCPGenerator.InitVegClassLCPLookup(g_RunParams.GetStrParam(
_T("VegClassLCPLookupFName")))
```

instantiates the `LCPGenerator`.

The block of code

```
MapLayer *m_pPolyMapLayer = (MapLayer *)(pEnvContext->pMapLayer);
. . .
ASSERT(gridRslt);
```

creates an Envision grid that can be used to check extents and to instantiate the PolyGridLookups.

The block of code

```
// Initializing PolyGrid. Using g_RunParams to figure out what to do
```

```
. . .
FailAndReturnOnBadStatus("PolyGrid initialization failed");
```

Determines whether `m_pPolyGridLkUp` is read from file or created from the landscape in Envsion and calls the appropriate constructor. Once created, `m_LCPGenerator` is given the pointer `m_pPolyGridLkUp.`

BOOL FlamMapAP::InitRun(EnvContext *pEnvContext, bool useInitSeed)

`InitRun()` is called by Envision at the beginning of each Envision run.

The scenario name is the name associated with an Envision run and is obtained via pEnvContext. At this time it is not clear how data for and from FlamMap runs is to be managed in light of changing scenarios, so while the scenario name is accessed and set in `InitRun()`, it is not used to create an input or output path, instead the path set by the initialization parameter WorkingPath is used to set the paths for input and output files associated with a FlamMap DLL run.

After paths for input and output files are set, delta array columns for `FlameLen`, `VegClass`, and `Scenario` are set locally from their Envision values by accessing them via `pEnvContext`.

A new `FireYearRunner`, `m_pFireYearRunner`, is instantiated for the run.

BOOL FlamMapAP::Run(EnvContext *pEnvContext)

First, the LCP file for all the current model year's fires is created by the call

```
m_LCPGenerator.PrepLandscape(pEnvContext)
```

This method converts the current Envision vegetation states to the LCP layer values needed by the FlamMap DLL, and writes the LCP file.

Second, the fires for the current model year are run by the call

```
m_pFireYearRunner->RunFireYear(pEnvContext)
```

This method zeros out the flame length array, loops over all the fires for the current model year, runs each one, and updates the flame length array.

Finally, the flame lengths generated by the fires for the current model year are written to the Envision delta array by the call

```
m_FlameLengthUpdater.UpdateDeltaArray(
       pEnvContext,
       m_pFireYearRunner,
       m_pPolyGridLkUp
       )
```

Note that this method requires `pEnvContext`, `m_pFireYearRunner`, and `m_pPolyGridLkUp`.

`BOOL FlamMapAP::EndRun( EnvContext *pContext )`

The `FireYearRunner` object created for the time step is deleted in this method.

`class ParamLookup`

This class can be thought of as a set of hashes, or maps which associate a string with a value. There is one map for each of these variable types: `CString`, `int`, `bool`, `float`, and `int`. `ParamLookup`'s methods provide access to these maps, allowing for adding, updating, checking for the existence of, and retrieving the value of elements in the maps. Maps and methods are also maintained for required parameters. This allows the programmer to set parameter names as required and to check if required parameter names are present in the `ParamLookup` object. Worth noting is the `BOOL ProcessInitFile()` method which reads an input file line by line and uses private parsing methods to determine the type of variable specified on the line of the input file and adds the variable to the appropriate map.

`class PolyGridLookups`

The abstraction provided by `PolyGridLookups` is that of a lookup table that associates a polygon and gridcell with the proportion of overlap the two have. It maintains its data using the `CompressedRow2dArray` class. PolygridLookups is explained in greater detail in a subsequent chapter. Here it will be described only briefly.

Three constructors are available. The first,

```
PolyGridLookups(
    int numGridRows,
    int numGridCols,
    int numPolys,
    BLD_FROM bldFrom,
    int maxElements,
    int defaultVal,
    int nullVal)
```

creates but does not initialize the instantiated object. It is recommended that this constructor not be used unless necessary. If this method is used, then the `set` methods must be used to fill the objects data structures. Note that, as described in `PolyGridLookups.h`, due to limitations imposed by `CompressedRow2dArray`, filling must be done in row-column order or `CompressedRow2DArray` will fail with an error.

The second builds the array from a grid map layer and a polygon map layer:

```
PolyGridLookups (
    MapLayer *m_pGridMapLayer,
    MapLayer *m_pPolyMapLayer,
    int cellSubDivisions,
    int maxElements,
    int defaultVal,
    int nullVal)
```

The final constructor reads a saved version of a PolyGridLookups object:

```
PolyGridLookups(const char *fName)
```

Constructing a `PolyGridLookups` object is can be computationally intensive so if a large number of polygons and grid cells are being used, or if a large number of subcells is used to compute the overlap, then it is strongly recommended that the `PolyGridLookups` object be saved after it is created and instantiated from file in future runs.

Class variables are described in `PolyGridLookups.h`, but four are worth discussing in more detail here.

`cellSubDivisions`, present in the second constructor listed above, is used to determine how many subcells of each gridcell will be used to calculate the grid cell's overlap with polygons. If `cellSubDivisions` is set to 1, then only the center of the gridcell will be checked to determine which polygon the gridcell overlaps. If it is set to 2, then the gridcell is divided into 4 equal subcells and the overlap of the center of each of those cells is determined. In this scenario, a cell could be determined to have a 0.25 overlap with each of 4 polygons for instance. If `cellSubDivisions` is set to 10, then the overlap would be determined by the center of each of 100 subcells.

`maxElements` is the estimate of how many elements the `CompressedRow2dArray` object will need. For instance if for each polygon were overlapped by three grid cells, `maxElements` would need to be 3 X the number of polygons. Slightly overestimating `maxElements` will use more memory than necessary, so small overestimates have little penalty, larger overestimates could impact performance or prevent a successful run. Underestimating will result in one or more memory reallocations, which are computationally expensive for large datasets.

`defaultVal` is passed through to `CompressedRow2DArray` and is the value `CompressedRow2DArray` does not actually store in the compressed array. In other words if 0 is the default value, `CompressedRow2DArray` will actually store any element with any value other than 0. It will not actually store an element with a value of 0. When an element not actually stored in the `CompressedRow2DArray` is requested, `CompressedRow2DArray` will return 0.

`nullVal` is passed through to `CompressedRow2DArray`, and must be a value that is not used as a legitimate entry in the `PolyGridLookups` data. Since the data is a tally of overlap between grids and polygons, any negative value will work. `nullVal` is used to initialize allocated memory before it is filled with valid data.

An Envision polygon index is simply an integer, however, a gridcell has both a row and a column index which must be converted to a single index for use with `CompressedRow2DArray` and converted back to row and column indexes for values obtained from `CompressedRow2DArray`. These conversions are done by the methods

```
int xyToCArrayCoord(const POINT &pt)
void CArrayCoordToxy(const int &CArrayCoord, POINT *pt)
```

The second of the above-listed constructors merits a discussion. Its definition can be found in `PolyGridLookups.cpp`. After its class variables are initialized, the following statement creates the lookup array used to relate multiple polygons to each grid cell:

```
gridToPolyLkUp = new CompressedRow2DArray<int>(
   numGridRows * numGridCols,
   numPolys,
   maxElements,
   defaultVal,
   nullVal)
```

To fill the compressed array each cell in the grid is accessed through nested `for` loops over rows and columns. Then for each subcell within the gridcell (a pair of nested `do` loops) the polygon in which the subcell resides (if there is one) is determined, and the count for that polygon is incremented:

```
polygonWt[pPoly->m_id]++;
```

After the subcells are traversed, the weights for each polygon are stored in `gridToPolyLkUp` by the call

```
SetGridPtElement(
       row,
       col,
       it->first,
       it->second)
```

Note that the weights entered into `gridToPolyLkUp` do not need to be normalized. The proportions of polygons in a gridcell or of grid cells in a polygon are calculated by the appropriate `get`… methods.

After all grid cells have been traversed, `gridToPolyLkUp` is complete.

The call

```
CreatePolyLookupFromGridPtLookup()
```

creates the `polyToGridLkUp`, also a `compressed2DArray`. This array is the transpose of `gridToPolyLkUp`.

> **NOTE: The values stored in the `CompressedRow2DArray` objects used by `PolyGridLookups` are tallies for the proportion of the gridcell in a polygon. The computation of the proportion of a polygon contained in a gridcell is calculated with this in mind.**

The `Get`… methods are used to obtain characteristics about the relationships between a polygon (gridcell) and the grid cells (polygons) it. The values returned by these can be used as needed to calculate an average, a plurality, minimum, maximum, etc. Those `Get`… methods for indexes, values, and proportions return vectors of values. For the same

polygon or grid point, the vector returned by any one of these methods corresponds to the vectors returned by the others.

Two methods in the code use the single instantiation of `PolyGridLookups`.

`LCPGenerator::PrepLandscape()`

uses it in the conversion of Envision vegetation states into gridded LCP layers, and

`FlameLengthUpdater::UpdateDeltaArray()`

uses it in the conversion of gridded flame lengths into flame lengths for Envision polygons.

class CompressedRow2DArray

`CompressedRow2DArray` implements a 2D array abstraction based on Dongarra's (http://web.eecs.utk.edu/~dongarra/etemplates/node373.html) description of compressed row storage. Because it is implemented using templates, it is wholly written in a `.h` file. The idea behind this implementation is to store only non-default values of the abstracted array and use a set of vectors for value and array cell index storage. `CompressedRow2DArray` has been extensively tested and will likely not need to be accessed in order to make changes to the FlamMap Interpreter.

`PolyGridLookups` is the only class that uses CompressedRow2DArray. For polygons, the index of the polygon is used as is, for grid cells, the row and column are converted to a single index which is used in calls to `CompressedRow2DArray` methods.

The storage structure used by `CompressedRow2DArray` is ordered. Because of this, the cells in the abstracted array must be filled in row-column order. Cells with the default value for the array (in the case of the FlamMap Interpreter, this value is 0), need not be filled. So, for example abstracted array cells could be filled in the order (1,7), (1,9), (2,3), (2,4), (3,6), but could not be filled in the order (1,7), (1,6), or in the order (2,3), (1,9).

`PolyGridLookups` and `CompressedRow2DArray` are described in greater detail in a later section.

class Fire

`class fire` contains and provides access to the variables associated with one model fire: year (`Yr`), Julian date (`JulDate`), wind speed (`WindSpd`), wind direction or azimuth (`WindAzmth`), occurrence probability (`BurnProb`), length of burn (`BurnPeriod`), and whether the fire is to be run or not (`DoRun`).

class FireQueue

As the name implies, `class FireQueue` is a queue of fires. It is instantiated by `FireYearRunner`. The queue is filled by the constructor which reads lines of a file and performs a Monte Carlo draw on each one to determine if it is to be run or not. If a fire is to be run a `fire` object is instantiated, and it is added to the queue. If not, it is discarded.

Methods are provided for accessing fire class variables from the fire object at the head of the queue and for deleting the head of the queue. `FireYearRunner` uses these methods to access the fire characteristics for the model fires it runs.

`class FireYearRunner`

`FireYearRunner` is the class that coordinates inputs and outputs for the FlamMap runs and makes the calls to the `MinTravelTime` DLL which runs FlamMap. `FireYearRunner` is instantiated by `FlamMapAP::InitRun()` at the start of each Envision run and deleted by `FlamMapAP::EndRun()` at the end of each Envision run.

The constructor sets up the object for running the model fires. In large part, it does the following:

1. Instantiates a `FireQueue` which determines which potential fires will be run and stores them.

2. Instantiates an `IgnitGenerator` which will be used to determine the ignition location of each model fire.

3. Allocates space for `m_pYrFlameLens` which, for each grid cell, stores the maximum flame lengths produced by model fires in a single model year.

4. Using a file, instantiates `m_FAParams`, a `ParamLookup` object that manages the parameters used by FlamMap.

5. Reads fuel moistures from a file into the `CString` `m_FuelMoistures`.

The `FireYearRunner::RunFireYear()` method is called one time per Envision time step. This method executes some preparatory steps, then steps through the model fires for the current model year executing each one. The preparatory steps are:

1. Create `flameLens`, the storage for the flame lengths from one FlamMap run.

2. Reset `m_pYrFlameLens`,the storage holding the maximum flame lengths, to 0.

3. Update the ignition generator to reflect the current landscape state:
   `m_pIgnitGenerator->UpdateProbArray(pEnvContext);`

4. Delete any fires on the queue from previous years.

The loop that runs the fires for the year begins with

```
while(m_pFireQ->FireCnt() > 0)
```

The steps executed within this loop are:

1. Break out of the loop if the next fire to run is not from the current model year.

2. If a fire is not to be run, delete it from the queue and loop.

3. Create an ignition file using the ignition generator.

4. Create the FlamMap parameters file containing the initialization data for FlamMap.

5. Create the names for flame length and fire intensity output files from FlamMap.

6. Delete the current fire from the `m_pFireQ`.

7. Create `mtt`, a new `CMinTravelTime` object that will be used to run FlamMap.

8. Tell `mtt` the name of the LCP file.

9. Have `mtt` load the input file.

10. Run FlamMap using `mtt->LaunchMTT()` and loop if it fails.

11. Get the flame lengths for the run. Note that the flame lengths provided by `CMinTravelTime` are not the flame lengths for the model burn. The flame lengths for the model burn are calculated from the values from fire line intensity.

    ```
    float *flameLenLayer = mtt->GetFliMapGrid()
    ```

    gets the fire line intensity, and

    ```
    flameLens[i] = 0.0775f * (float)pow((float)flameLenLayer[i],

    (float)0.46)
    ```

    computes the flame length from fireline intensity.

12. Write ASCII files for the flame length and fire line intensity.

13. Update `m_pYrFlameLens` which holds the maximum annual flame length for each grid cell.

After a fire year is run, the flame lengths for the model fire year are entered into the Envsion delta array by the `FlameLengthUpdater` class. `FireYearRunner::GetFlameLen()` is the method that the allows that allows the `FlameLengthUpdater` class to access the flame lengths accumulated by `FireYearRunner` during the run of a fire year.

The destructor for `FireYearRunner` deletes the objects and data structures that were dynamically allocated during its run.

`class FlameLengthUpdater`

`class FlameLengthUpdater` is instantiated by declaration in the `FlamMapAP` class definition in `FlamMapAP.h`. `FlameLengthUpdater::UpdateDeltaArray()` is the method through which this class does its real work: computing the mean flame length for each Envision polygon by taking the weighted mean of flame lengths of grid cells that intersect with the polygon

and assigning that value to the polygon. The first step in the `UpdateDeltaArray()` method is to traverse the `FireYearRunner`'s flame length array to determine the maximum and mean flame lengths.

The second step is to loop over Envisions polygons and compute the mean flame length for each. For each polygon, the steps involved in this are:

1. Insure sufficient space for the vectors returned by method calls to the `PolyGridLookups` object, `pPolyGridLkUp`, and initialize these vectors to 0.

2. Get the indexes for the grid cells overlapping the polygon:

   ```
   pPolyGridLkUp->GetGridPtNdxsForPoly(Poly, pGridPtNdxs)
   ```

3. Get the proportions of the polygon for each overlapping grid cell (i.e. the proportions used to compute the weighted average flame length for the polygon):
   `pPolyGridLkUp->GetGridPtProportionsForPoly(Poly, GridPtProportions)`

4. For each of the grid points obtained in step 2., add its weighted contribution to the flame length for the polygon:

   ```
   for(GridPt=0;GridPt<GridPtCnt;GridPt++) {
          TtlPolyFlameLength +=
          pFireYearRunner->GetFlameLen(
                 pGridPtNdxs[GridPt].x,pGridPtNdxs[GridPt].y) *
          GridPtProportions[GridPt];
   } // for(GridPt=0;GridPt<nGridPtCnt;GridPt++)
   ```

5. Enter the result into Envision's delta array using the call:

   ```
   AddDelta(…)
   ```

   NOTE: Due to the algorithm used to compute polygon/grid cell overlap in PolyGridLookups, it is possible that a polygon can be computed to overlap with no grid cells. If this situation arises, the polygon's flame length value will always be updated to 0.

`class IgnitGenerator`

`IgnitGenerator` maintains a probability surface for ignitions and provides a method for performing Monte Carlo draw(s) against that surface to produce an Arc shapefile of ignition points. It is instantiated by the `FireYearRunner` class

In its constructor, it allocates and zeros out `m_MonteCarloIDUProbs`, the vector it uses to store probabilities for Monte Carlo draws. The vector contains one entry for each Envision polygon.

The `UpdateProbArray()` method steps through the Envision polygons, gets the fields needed to compute the probability for a polygon, and then computes the polygon's

probability based on a hard-coded probability function. A running total of probabilities is kept in m_MonteCarloIDUProbs so that it can be used with a Monte Carlo draw.

the GenIgnition() method generates one or more ignition points and saves them to an Arc shapefile.

class LCPGenerator

Using a starting LCP file, a PolyGridLookups object, and a VegClassLCPLookup object, LCPGenerator effectively manages an LCP file image in memory and writes it to an LCP file when its WriteFile() method is called. The contents of the LCP file are stored in two separate structures. The header is stored in a struct, and the body, containing the 8 LCP layers, is stored in pointer to a short.

The header of the LCP file is implemented as a struct. Because some of the fields in the header align on 2 byte blocks, memory alignment must be set to 2 bytes before the struct is declared and then set back to the default afterwards. This is done by two #pragma directives in LCPGenerator.h. The code for that is:

```
#pragma pack(push,2)
struct LCPHeader {
… struct member declarations here …

}
#pragma pack (pop)
```

Starting LCP data is read from an existing LCP file. The landscape represented by the LCP file must align with the extent of the Envision landscape in order for the FlamMap Interpreter to return valid results. The header of the starting LCP file and the layers for slope, aspect, and elevation should all be correct. The five layers associated with vegetation characteristics will be updated when the FlamMap Interpreter runs.

LCPGenerator is instantiated by declaration in FlamMapAP.h and is initialized in FlamMapAP::Init() at run time.

LCPGenerator::InitLCPValues() opens the LCP file whose name is passed in as an argument. The head of this file is read into the header struct m_pLCPHeader. Height, canopy cover, and bulk density units must match with the values in the lookup data used by the VegClassLCPLookup object, or results will be meaningless. The LCPGenerator sets and checks these units after the header is read in to memory. After m_pLCPHeader is complete, the constructor reads in the data for the 8 LCP data layers.

LCPGenerator's VegClassLCPLookups object, m_pVegClassLCPLookup, is instantiated by a call to LCPGenerator::InitVegClassLCPLookup(). This call is made in FlamMapAP::Init().

LCPGenerator's PolyGridLookup object, m_pPolyGridLkUp is not instantiated by LCPGenerator, but is passed in as a pointer in FlamMapAP::Init() in a call to LCPGenerator::InitPolyGridLookup().

`LCPGenerator::PrepLandscape()` is the method that loads the values associated with vegetation into the in-memory representation of the LCP layers. Two polygon fields from Envision are used to do the lookup using the `VegClassLCPLookup` object: `VegClass` and `Variant`. A nested pair of loops runs over the grid rows and columns, to determine the LCP values for each cell in the LCP file the steps are:

1. Get the index of the polygon that most overlaps the grid cell (i.e. plurality rules):

   ```
   nPluralPolyNdx = m_pPolyGridLkUp->GetPolyPluralityForGridPt(i,j)
   ```

2. Get the vegClass and variant for that polygon:

   ```
   pEnvContext->pMapLayer->GetData(nPluralPolyNdx, vegClassCol, vegClass);
   pEnvContext->pMapLayer->GetData(nPluralPolyNdx, variantCol, variant);
   ```

3. Update the LCP fields with the fields corresponding to the vegClass and variant:

   ```
   SetFuelModel(i,j,m_pVegClassLCPLookup->GetFuelModel(vegClass, variant));
   SetCanopyCover(i,j,m_pVegClassLCPLookup->GetCanopyCover(vegClass,
         variant));
   SetStandHeight(i,j,m_pVegClassLCPLookup->GetStandHeight(vegClass,
         variant));
   SetBaseHeight(i,j,m_pVegClassLCPLookup->GetBaseHeight(vegClass,
         variant));
   SetBulkDensity(i,j,m_pVegClassLCPLookup->GetBulkDensity(vegClass,
         variant));
   ```

After the LCP layers are updated in memory, the header and the layers are written to file.

`class VegClassLCPLookup`

`VegClassLCPLookup` is the class used for converting vegetation classes from Envision polygon space into LCP attributes in FlamMap grid space. This involves a lookup table to translate a vegetation state and variant into LCP attributes. The lookup is implemented as a `map`.

The constructor reads the input file, checks the first row to insure field names are correct, then reads subsequent rows. The `AddValues()` method builds a single key from the `VegClass` and `Variant` values. The key is generated by the code:

```
key = (unsigned long)VegClass * 10000 + (unsigned long)Variant
```

Note that this assumes there will never be more than 10000 variants and that an unsigned long is large enough to hold a key created in this manner.

The `AddValues` method loads the field values into a `LookupValues struct` (see `VegClassLCPLookup.h`) and assigns that to the datum associated with the key.

Field values associated with keys are accessed through the `Get…` methods, which build a key and access the associated datum.

PolyGridLookups

This section provides a more detailed explanation of PolyGridLookups through an example that goes from conceptualization through data structures and coding issues.

Computing Polygon/Grid Overlap.

Envision divides the landscape using arbitrary polygons. A representation of this for a simple landscape is shown in Figure 5. In this landscape there are 8 polygons.

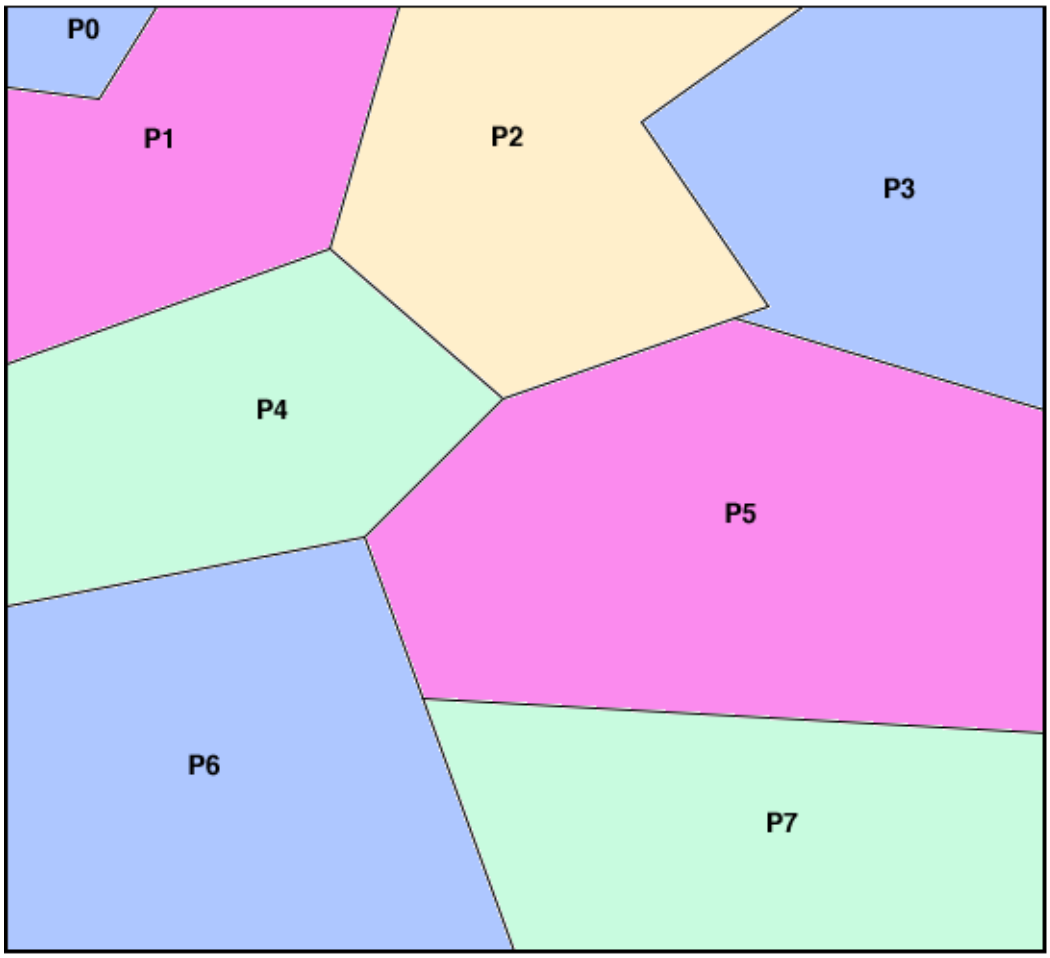

Figure 5. Landscape divided into 8 irregular polygons.

The same landscape could be covered by a grid as in Figure 6, which shows a 3-by-3 regular grid.

The question arises of how to transfer attributes from the polygon representation to the grid representation, and vise versa. For continuous variables, for example elevation, a weighted mean would work well. For categorical values, such as vegetation type, a plurality might be better suited. Either way, one needs to know the proportion of intersection between grid cells and polygons.

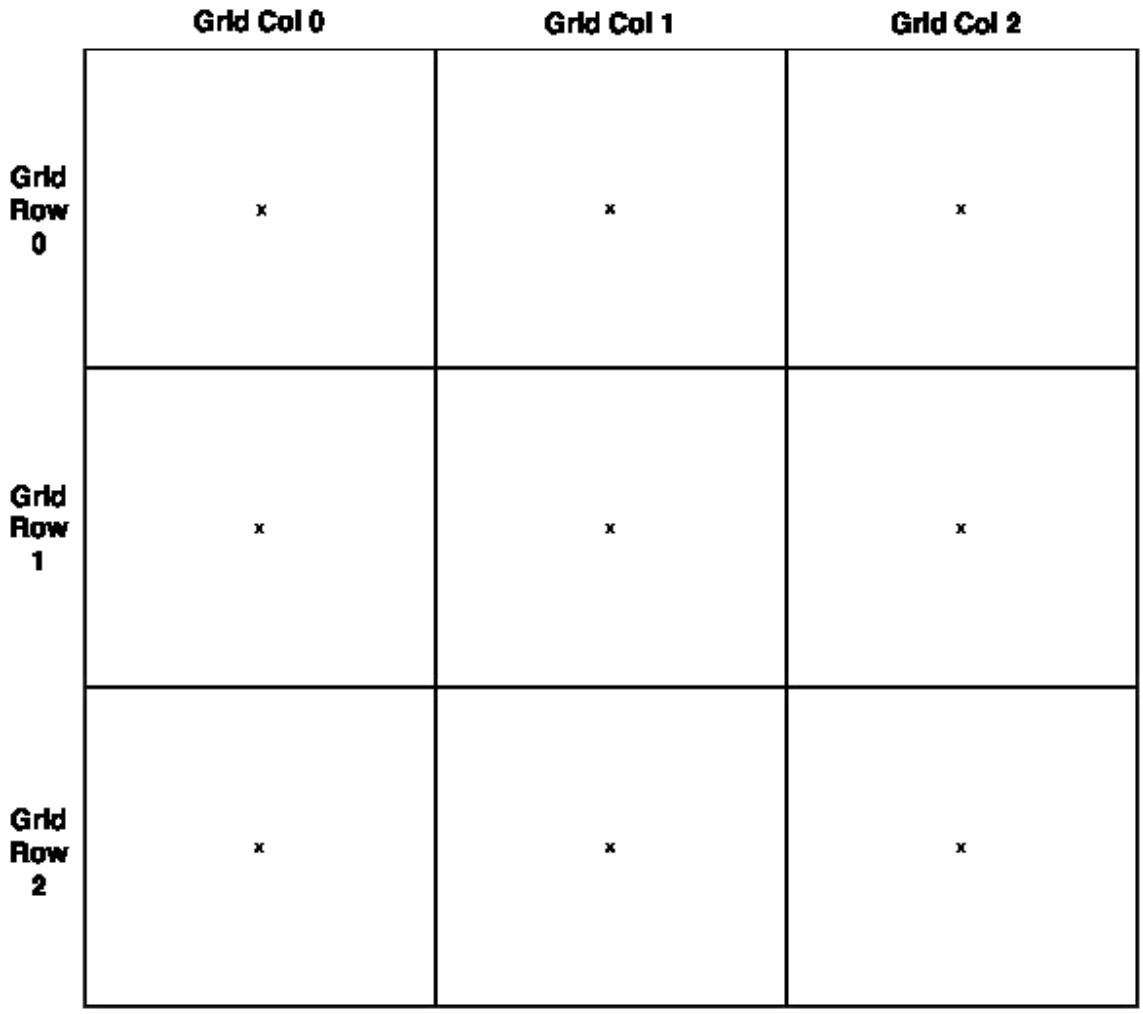

Figure 6. Landscape divided into a 3-by-3 regular grid. Grid cell centers are marked by an "x".

One computational approach is to use the centers of the grid cells to define the level of intersection. We can envision this by overlaying the polygon representation with the gridcell representation as in Figure 7.

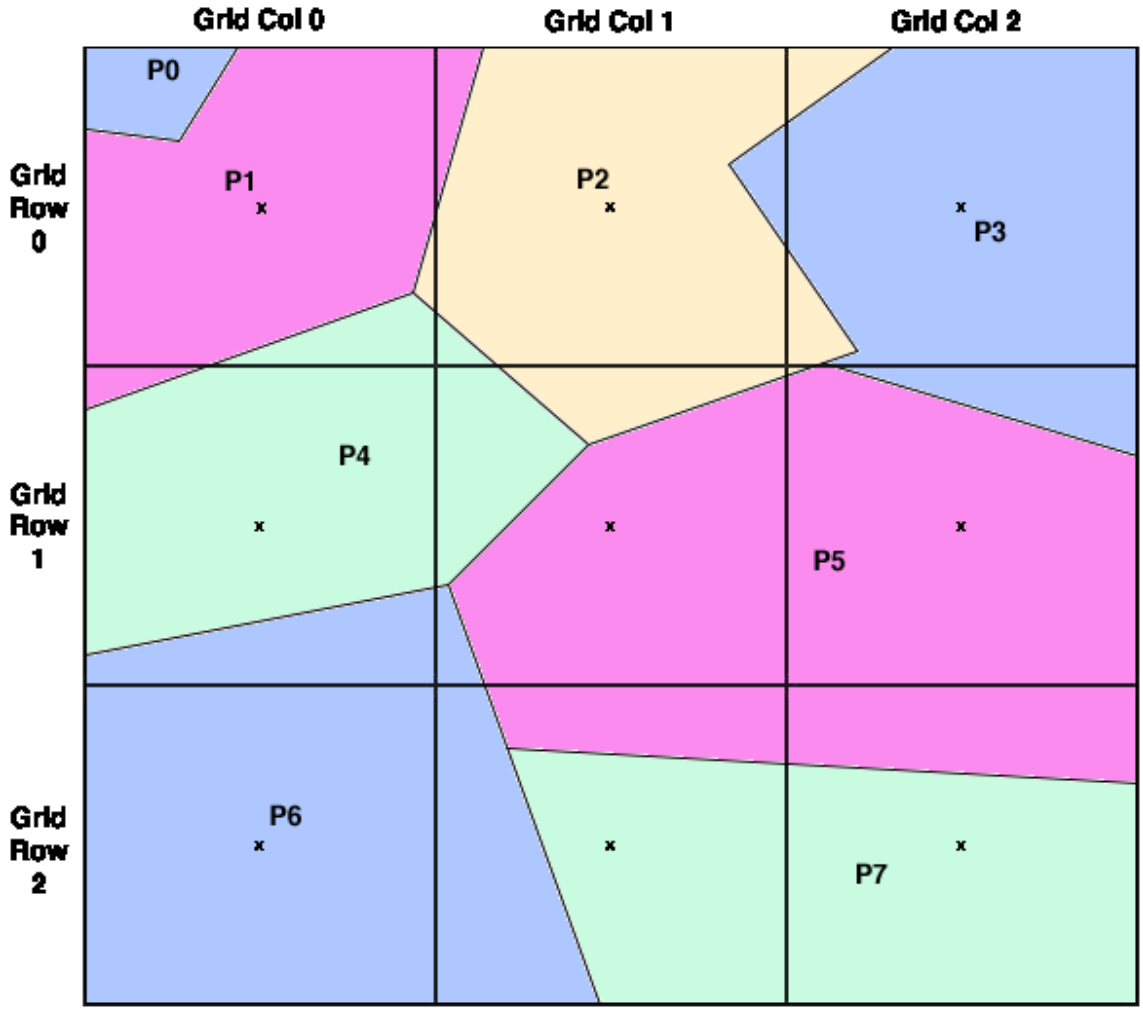

Figure 7. Grid cells overlaid onto the polygon representation of the landscape.

Using this method with the example, each grid cell would get the attributes of a single polygon, for instance grid cell (1,1) would get its attributes from polygon P5. Since polygons can span parts of various grid cells, they might lie under 0, 1, or more grid cell centers. In this example P1 would get attributes from grid cell (0,0), P5 from grid cells (1,1) and (1,2). Note that in this example, P0 is not counted as intersecting with any grid cells. This condition would need to be checked for and dealt with in the code.

A way to add more precision to the intersections of polygons and grid cells is to subdivide the grid cells and compute the intersection based on the centers of the subcells. Figure 8 shows the same grid overlay as Figure 7, but the grid cells have been divided by three on each edge to produce 9 subcells per grid cell.

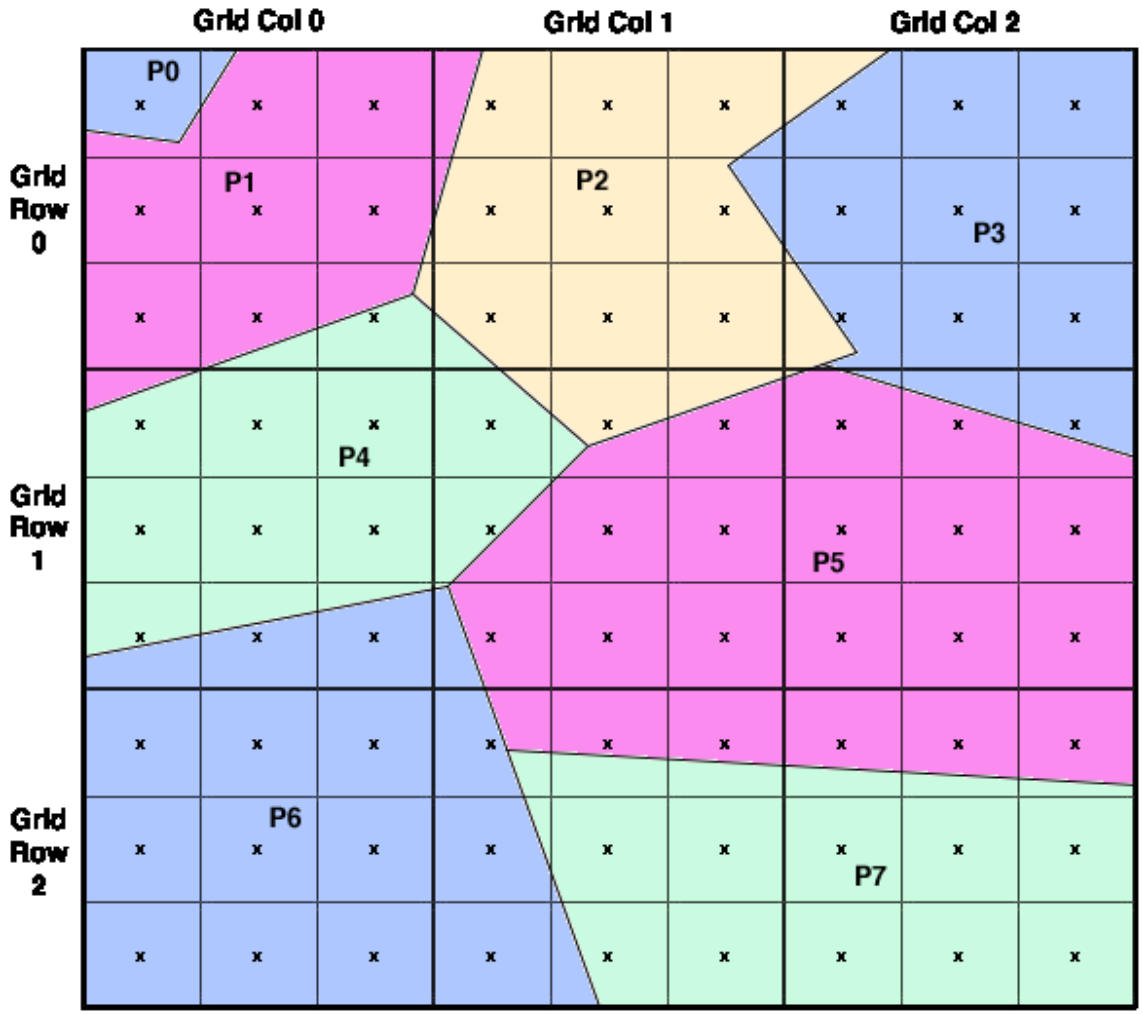

Figure 8. Subdivided grid cells overlaid onto the polygon representation of the landscape.

Using the centers of the subcells to compute overlap yields the following for grid cell (1,1):

P2: 1/9
P4: 2/9
P5: 6/9

For polygon P5 the computed overlap based on the 19 subcells that P5 intersects the subcell center is:

Grid cell (1,1): 6/19
Grid cell (1,2): 8/19
Grid cell (2,1): 2/19
Grid cell (2,2): 3/19

Three concepts should be clear at this point. First, the more subdivisions, the greater the precision of computed overlap. Second, no matter how many subcells there are, the number of polygons (grid cells) a grid cell (polygon) actually overlaps never changes. Finally, the number of overlaps between polygons and grid cells is generally dependent on the relative size of polygons and grid cells. If grid cells are substantially smaller than polygons, then most grid cells will overlap with one polygon and most polygons will overlap with multiple grid cells. Conversely if grid cells are substantially larger than polygons, most grid cells will overlap multiple polygons and most polygons will overlap with one grid cell. If grid cells and polygons are of approximately the same size, then most grid cells will overlap several polygons and most polygons will overlap several grid cells.

Overlap Representation

In the course of converting a whole landscape between polygon and grid representations, all polygons and grid cells must be analyzed for overlap with their counterparts. A two-dimensional array provides a simple and straightforward method of doing this. In order to use a two-dimensional array, however, the row-column indices of the grid must be converted into one dimension. This is easily done by the formula:

```
LookupGridIndex = RowIndex * NumColumns + ColumnIndex
```

Tables 1 and 2 show lookup tables for the overlaps depicted in Figures 7 and 8, respectively.

Using a lookup table like Table 1 or Table 2 makes it easy to get the proportion of overlap between a grid cell and a polygon. For the proportion of a polygon in a given grid cell the algorithm is:

```
RowColIndex = ComputeRowIndex(Row,Col)
OverlapCellCount = TableCellValue(RowColIndex, Polygon)
ProportionOverlap = OverlapCellCount / TotalOfRow (RowColIndex)
```

For the proportion of a grid cell in a given polygon, the formula is similar, but instead of the row sum, a column sum is used:

```
RowColIndex = ComputeRowIndex(Row,Col)
OverlapCellCount = TableCellValue(RowColIndex, Polygon)
ProportionOverlap = OverlapCellCount / TotalOfCol (Polygon)
```

For Table 2, using grid cell (1,1) and polygon 5, polygon 5 comprises 6/9 of grid cell (1,1) and grid cell (1,1) comprises 6/19 of polygon 5.

Table 1. Table of overlaps between grid cells and polygons represented in Figure 7.

			Polygon Index							
Row	**Col**	**Row/ Col index**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	**0**	**0**	0	**1**	0	0	0	0	0	0
0	**1**	**1**	0	0	**1**	0	0	0	0	0
0	**2**	**2**	0	0	0	**1**	0	0	0	0
1	**0**	**3**	0	0	0	0	**1**	0	0	0
1	**1**	**4**	0	0	0	0	0	**1**	0	0
1	**2**	**5**	0	0	0	0	0	**1**	0	0
2	**0**	**6**	0	0	0	0	0	0	**1**	0
2	**1**	**7**	0	0	0	0	0	0	0	**1**
2	**2**	**8**	0	0	0	0	0	0	0	**1**

Table 2. Table of overlaps between grid cells and polygons represented in Figure 8.

			Polygon Index							
Row	**Col**	**Row/ Col index**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	**0**	**0**	**1**	**7**	0	0	**1**	0	0	0
0	**1**	**1**	0	0	**9**	0	0	0	0	0
0	**2**	**2**	0	0	0	**9**	0	0	0	0
1	**0**	**3**	0	0	0	0	**7**	0	**2**	0
1	**1**	**4**	0	0	**1**	0	**2**	**6**	0	0
1	**2**	**5**	0	0	0	**1**	0	**8**	0	0
2	**0**	**6**	0	0	0	0	0	0	**9**	0
2	**1**	**7**	0	0	0	0	0	**2**	**3**	**4**
2	**2**	**8**	0	0	0	0	0	**3**	0	**6**

Memory Efficiency

A lookup table contains an entry for each grid cell/polygon pair. Even in the small example depicted in Table 2, only 17 of 72 cells have non-zero values. For a landscape with 100,000 polygons and 100,000 grid cells, and in which the mean overlap is 3, the resulting table would have 300,000 non-zero entries and 9,000,700,000 zero entries. At 4 bytes per entry, that represents almost 40 GB of memory. Obviously, some form of array compression is called for. For the FlamMap interpreter, a compressed-row two-dimensional array was implemented. The algorithm implemented stores only non-zero values and utilizes four vectors to present an array abstraction of the stored values. Figure 9 illustrates how these vectors are used for the array depicted in Table 2.

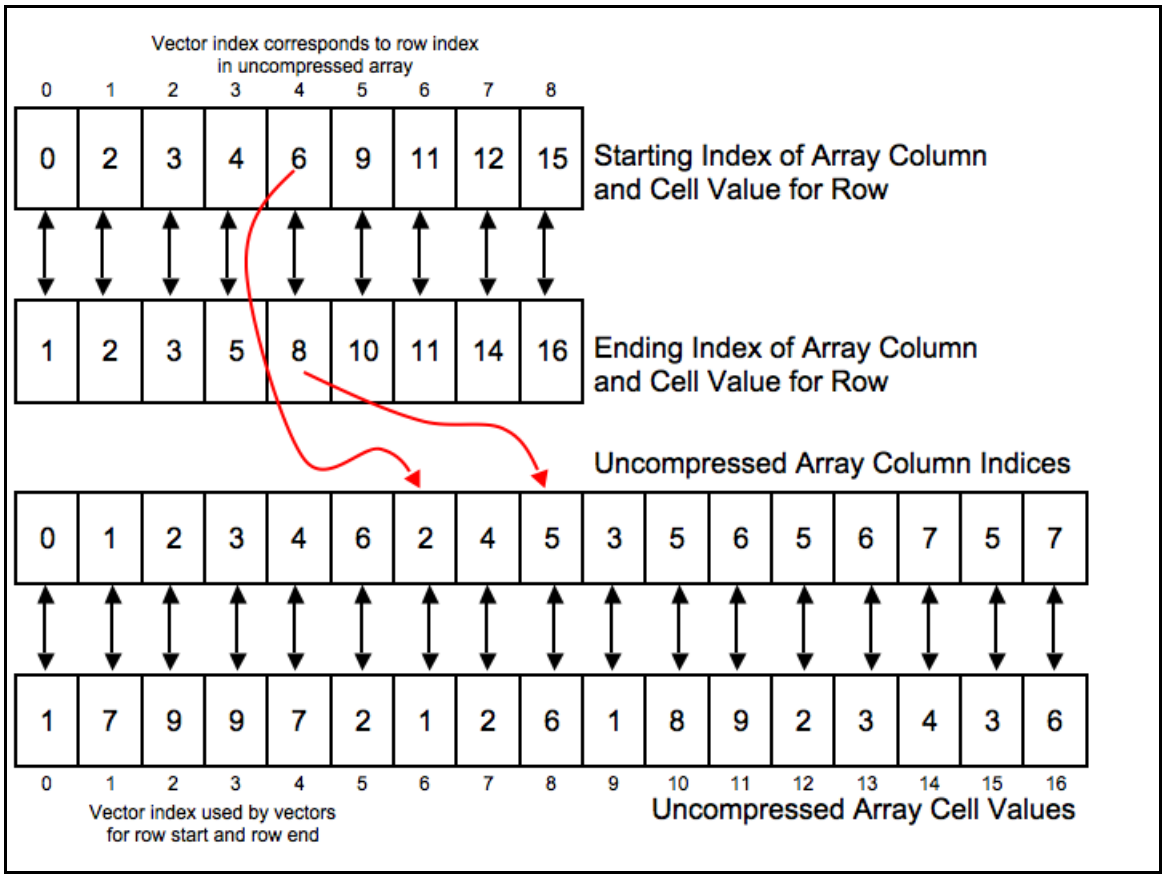

Figure 9. Compressed 2-dimensional array corresponding to the array depicted in Table 2. Red arrows correspond to entries for grid cell (1,1).

In Figure 9, the vector labeled *Uncompressed Array Cell Values* holds the non-zero values from the uncompressed array in row-column order. The vector labeled *Uncompressed Array Column Indices* holds the uncompressed array's column index for each of the values in *Uncompressed Array Cell Values.* Note that the index values for these two vectors do not correspond to anything in the uncompressed array.

For the vectors labeled *Starting Index of Array Column and Cell Value for Row*, and *Ending Index of Array Column and Cell Value for Row*, the vector index corresponds to the row in the uncompressed array. The value in each cell indexes into the *Uncompressed Array Column Indices* and *Uncompressed Array Cell Values* vectors, to get the starting and ending locations of uncompressed array column indices and uncompressed array cell values.

A walk-through of several data is accesses in the compressed array should make things clearer. First, an access of the overlap data for grid cell (1,1) and polygon 2:

The Row/Col Index for grid cell (1,1) is 4. So we go to element 4 in *Starting Index of Array Column and Cell Value for Row* and obtain the value 6. We then go to element 6 in *Uncompressed Array Column Indices* and see that the value is 2. The value 2 corresponds

to the column for polygon 2 in the uncompressed array, so we know we have found the correct entry for our grid cell column pair. We read the value from element 6 of *Uncompressed Array Cell Values*, which is 1. If we check this against the original uncompressed array, we see we obtained the correct value.

Now let's get the value for grid cell (1,1) and polygon 0. Again we start with element 6 of *Uncompressed Array Column Indices* and see that the column is 2. This tells us that the entry for column 0 was not stored in the compressed array. So we know the value of that column was 0. If we check this against the uncompressed array, we can verify that we are correct.

If we look for the value of grid cell (1,1) and polygon 3, we will have to traverse forward from our starting point in *Uncompressed Array Column Indices,* and as we do this, we need to ensure that we don't go beyond the entries associated with row 4 of the uncompressed array. We start again with element 6 of *Uncompressed Array Column Indices* and note that the uncompressed column index, 2, is less than our search column index, 3. We then check element 6 of *Ending Index of Array Column and Cell Value for Row* to make sure the next element of *Uncompressed Array Column Indices* is not beyond the end of the entries for uncompressed row 4. It turns out that elements 6, 7, and 8 of *Uncompressed Array Column Indices* are all for uncompressed row 4, so we look at the next element, element 7 which contains the value 4, telling us that column 4 is the next column in the uncompressed array that has a non-zero value. We were concerned with row 4 column 3, which is missing from the compressed array, so it has a value of 0. This is confirmed by looking at the uncompressed array.

If we look for the value of grid cell (1,1), polygon 7, we will step through the elements of *Uncompressed Array Column Indices* and find that the last non-zero element for uncompressed row 4 has an uncompressed column index of 5, and so we return the value 0.

PolyGridLookups Implementation

The basic algorithmic logic for the conversion between polygons and grid cells has been made clear, but a few further details bear discussion before describing code details. The first is how grid cell/polygon overlaps are used. In most cases, for instance with a plurality or weighted mean, information about the proportion of overlap for all overlapping cells or polygons is needed. Looping through all possible grid cell/polygon combinations for a given grid cell or given polygon to find the non-zero combinations is tedious and inefficient.

In the above example, the indexes of overlapping polygons are stored in adjacent elements of *Uncompressed Array Column Indices*, and the associated values are stored in the corresponding adjacent elements of *Uncompressed Array Cell Values*. With the precomputed starting and ending indices for the in *Starting Index of Array Column and Cell Value for Row*, and *Ending Index of Array Column and Cell Value for Row*, the indexes and proportions of all polygons overlapping a given grid cell can be easily and efficiently implemented as methods.

Unfortunately, doing the same for polygons is not as straightforward or efficient. The entire *Uncompressed Array Column Indices* would have to be searched linearly. So in order to have an efficient lookup for grid cells intersecting a given polygon, a second compressed-row 2-dimensional array is created. It represents the transpose of the other array.

With all this out of the way, here is a more detailed supplementary walk-through of the `PolyGridLookups` and `CompressedRow2DArray` classes.

`class PolyGridLookups`

It is important to first understand some of the class variables declared in `PolyGridLookups.h`:

`gridToPolyLkUp` and `polyToGridLkUp`. These are the `CompressedRow2DArray` objects used by the class. `gridToPolyLkUp` is the compressed array used to get data concerning polygons that overlap a given grid point. In other words each row in the compressed array represents a grid cell. This is analogous to the compressed array used in the examples above. `polyToGridLkUp` is a compressed array used to get data concerning grid cells that overlap a give polygon. Polygons are represented by the rows in the compressed array. `gridToPolyLkUp` and `polyToGridLkUp` represent transposes of each other.

`maxElements` is the number of elements to be stored in the compressed array. If `maxElements` is initially set too low, one or more reallocations of the storage used for the compressed array will have to be performed, with the associated penalty in computational performance. `maxElements` has no effect on performance when a `PolyGridLookups` object is read from file.

`defaultVal` is the default value used in the compressed array. In most, if not all, cases, this will be 0. This is the value the compressed array will not store and the value the compressed array will return for any query for data points it has not stored.

`nullVal` is passed through to the compressed array. `nullVal` is used to initialize allocated memory before it is filled with valid values. In most, if not all cases, -9999 is an appropriate value to use for `nullVal`.

`numPolys`, `numGridRows`, and `numGridCols` are the numbers of polygons, grid rows, and grid columns, respectively, for the landscape.

`bldFrom` is used when the user is building a `PolyGridLookups` from scratch as opposed to from layers or from file. `bldFrom` tells `PolyGridLookups` whether the user will be building the `gridToPolyLkUp` compressed array or the `polyToGrid` compressed array.

Also in the .h file the methods for converting grid point coordinates to and from the single dimension used by `CompressedRow2DArray` are defined. These are `xyToCArrayCoord()` and `CArrayCoordToxy()`.

One constructor allows for the creation of an empty PolyGridLookups:

```
PolyGridLookups::PolyGridLookups(
       int numGridRows,
       int numGridCols,
       int numPolys,
       BLD_FROM bldFrom,
       int maxElements,
       int defaultVal,
       int nullVal)
```

If map layers for the grid and polygon landscape representations are not available, then this is the constructor that will be needed. Note that the user must specify whether the initial lookup grid will be built from polygons or from grid points. This lets the code know whether to build the `gridToPolyLkUp` or `polyToGridGridLkUp` `CompressedRow2DArray`. Once the user has finished building the initial grid, the transpose grid must be build by calling one of

```
PolyGridLookups::CreateGridPtLookupFromPolyLookup()

PolyGridLookups::CreatePolyLookupFromGridPtLookup().
```

The constructor

```
PolyGridLookups::PolyGridLookups (
       MapLayer *m_pGridMapLayer,
       MapLayer *m_pPolyMapLayer,
       int cellSubDivisions,
       int maxElements,
       int defaultVal,
       int nullVal)
```

frees the user from having to worry about creating the `gridToPolyLkUp` and `polyToGridLkUp` `CompressedRow2DArray` objects and also serves as an illustration of how these objects are created. Note the `int cellSubDivisions` argument to the constructor. This is used to direct the subdivision of each grid cell when determining the overlap between grid cells and polygons. `cellSubdivisions` is the number of division used in each of the two dimensions of the grid cell. In other words, the number of subdivisions for a grid cell will be `cellSubdivisions` * `cellSubdivisions`. So if `cellSubdivisions` is 10, each cell will be divided into 100 subcells to determine the polygon grid cell overlap.

Continuing with this constructor, values are assigned, and `gridToPolyLkUp` is instantiated. Some temporary variables used in subcell computation are assigned and the size of the subcell is computed. The overlap calculation is run on each cell (the nested `for(int row=0`… and `for(int col=0`… loops)

Within the row/col nested loops are a pair of `do` loops which iterate over the subcells. For each subcell, a the polygon under the subcell center is found and a tally is taken. After the tally is complete for all subcells of a single grid cell, the tally values for each polygon are entered into `gridToPolyLkUp` in order. (Remember that entries into a `CompressedRow2DArray` must be done in row-column order). No normalization of tallies is done.

After all grid cells have been processed, gridToPolyLkUp is complete and polyToGridLkUp is created by the method call CreatePolyLookupFromGridPtLookup().

The remaining constructor reads a PolyGridLookups object from file and is fairly straightforward. Worth noting is that the reading of the gridToPolyLkUp and polyToGridLkUp objects are accomplished through calls to a CompressedRow2DArray constructor that takes the pointer of an open file as its argument.

The Get… methods are fairly self-explanatory. Val or Vals in a method name refers to the value that is stored in the lookup array, in other words, the tally of subcells for the intersection of a single grid point and a single polygon. Those returning vectors are returning all values associated with non-zero values in a lookup row. An example may serve to clarify here. Using the example from Figure 8, Table 2, and Figure 9 the following calls would yield the results shown:

GetPolyCntForGridPt(1,1)
returns: 9

GetPolyNdxsForGridPt(1,1, *ndxs)
ndxs: 2, 4, 5

GetPolyValsForGridPt(1,1, *vals)
vals: 1, 2, 6

GetPolyProportionsForGridPt(1,1, *proportions)
proportions: 0.111, 0.222, 0.666

GetPolyValTtlForGridPt(1,1)
returns: 9

GetPolyPluralityForGridPt(1,1)
returns: 5

The Set… methods are used if building a PolyGridLookups from scratch and are self-explanatory. These should not be used unless one fully understands PolyGridLookups and CompressdRow2DArray.

class CompressedRow2DArray
CompressedRow2DArray implements the structure illustrated in Figure 9. The vectors in Figure 9 are implemented as vector objects as follows:

rowStartNdx: Starting Index of Array Column and Cell Value for Row
rowEndNdx: Ending Index of Array Column and Cell Value for Row
colNdx: Uncompressed Array Column Indices
values: Uncompressed Array Cell Values

CompressedRow2DArray has two constructors. One reads a stored object from an open file and is self explanatory. The other,

```
CompressedRow2DArray(
       int numRows,
       int numCols,
       int allocSz,
       T defaultVal,
       T nullValue)
```

Initializes variables and allocates space.

The `SetElement()` method sets the value of an element. The variables `crntAddCol` and `crntAddRow` are used to check that the row and column being added conform to the row-column order requirement for adding data. A capacity check is also performed and storage increased to if necessary.

The `SetElement` and `Get`.. methods are self-explanatory if one understands the algorithm and the vector container class.

Two `WriteToFile()` methods and two `ReadFromFile()` methods are available. Those that take a filename as an argument allow for the writing or reading of a `CompressedRow2DArray` object to a named file. The other two allow for the writing or reading of a `Compressed2DArray` object to an already open file. These latter two were designed to be used in conjunction with `PolyGridLookups`.

The `Transpose()` method takes its own object and produces a new `CompressedRow2DArray` object for the transpose of itself. The idea behind this method is to first construct the `rowStartNdx` and `rowEndNdx` arrays using one pass through the untransposed array's `colNdx` vector. Once that is done, the transpose array's `colNdx` and `values` vectors can be filled by random access during a single pass through the untransposed array. The steps in this process are:

1. Instantiate `newArray`, a new `CompressedRow2DArray` object to hold the transposed matrix.

2. Count the number of column values for each row in the untransposed matrix using `vector tmpColCounter`.

3. Translate those counts into the starting and ending indexes of each row in the transpose array.

4. Create storage for and fill the vectors for column numbers and values in the transpose array

5. Set the values for `crntAddRow` and `crntAddCol` in the transposed compressed array.

Conclusion

In all likelihood, changes will be made to the FlamMap Interpreter code and this guide will either be updated or, regrettably, dated in the near future. Such is the relationship between code and its documentation.

The FlamMap Interpreter will likely be changed to use a version of PolyGridLookups recently added to the Envision Libs project. All, or nearly all, of the PolyGridLookups documentation should apply to the library version.

References

Envision. Envision version 5: An analysis framework for policy-driven alternative futures analyses. Available at http://envision.bioe.orst.edu/.

Finney, Mark, 2006. An Overview of FlamMap Fire Modeling Capabilities. In Andrews, Patricia L.; Butler, Bret W., comps. 2006. *Fuels Management-How to Measure Success: Conference Proceedings*. 28-30 March 2006; Portland, OR. Proceedings RMRS-P-41. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station. p. 213-220. Available online at http://www.fs.fed.us/rm/pubs/rmrs_p041/rmrs_p041_213_220.pdf.

Missoula Fire Sciences Laboratory, 2006. FlamMap 3.0.0 for Windows (Computer software package). Available at http://www.firemodels.org/index.php/flammap-software/flammap-downloads.

REFERENCES CITED

Albert, J. H. (1990). A Bayesian test for a 2-way contingency table using independence priors. *Canadian Journal of Statistics-Revue Canadienne De Statistique 18*(4), 347-363.

Amatulli, G., & Camia, A. (2007). Exploring the relationships of fire occurrence variables by means of CART and MARS models. In *Proceedings Wildfire2007 IV International Wildland Fire Conference* (pp. 13-17). Sevilla, Spain.

Amatulli, G., Rodrigues, M. J., Trombetti, M., & Lovreglio, R. (2006). Assessing long-term fire risk at local scale by means of decision tree technique. *Journal of Geophysical Research-Biogeosciences, 111*(G4S05).

Andrews, P. L., (2007). BehavePlus fire modeling system: Past, present, and future. In *Proceedings of 7th Symposium on Fire and Forest Meteorological Society* (pp. 23-25). Retrieved from: http://ams.confex.com/ams/pdfpapers/126669.pdf

Arienti, M. C., Cumming, S. G., Krawchuk, M. A., & Boutin, S. (2009). Road network density correlated with increased lightning fire incidence in the Canadian western boreal forest. *International Journal of Wildland Fire, 18*(8), 970-982.

Bachelet, D., Lenihan, J. M., Daly, C., Neilson, R. P., Ojima, D. S., & Parton, W. J. (2001). MC1: A dynamic vegetation model for estimating the distribution of vegetation and associated carbon, nutrients, and water (Technical report PNW-GTR 508). Portland, OR, USA: U. S. Forest Service Pacific Northwest Research Station.

Bachelet, D., & Price, D. (2008). DGVM responses to the latest IPCC future climate scenarios. *Global and Planetary Change, 64*(1-2), 1-2.

Badia, A., Serra, P., & Modugno, S. (2011). Identifying dynamics of fire ignition probabilities in two representative Mediterranean wildland-urban interface areas. *Applied Geography, 31*(3), 930-940.

Bolte, J. P., (2011). *Envision: A Guide to Application Development*. Retrieved from: http://envision.bioe.orst.edu/techdocs.htm

Bolte, J. P., (2004). *Evoland Technical Documentation*. Retrieved from: http://envision.bioe.orst.edu/techdocs.htm.

Bolte, J. P., Hulse, D. W., Gregory, S. V., & Smith, C. (2007). Modeling biocomplexity - actors, landscapes and alternative futures. *Environmental Modelling & Software, 22*(5), 570-579.

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America. 99*(Suppl 3), 7280-7287.

Breiman, L., Friedman, J., Stone, C.J., & Olshen, R.A., (1984). *Classification and Regression Trees*. Boca Raton, FL, USA: CRC Press, LLC.

Brosofske, K. D., Cleland, D. T., & Saunders, S. C. (2007). Factors influencing modern wildfire occurrence in the Mark Twain National Forest, Missouri. *Southern Journal of Applied Forestry, 31*(2), 73-84.

Brunsdon, C., Fotheringham, A. S., & Charlton, M. E. (1996). Geographically weighted regression: A method for exploring spatial nonstationarity. *Geographical Analysis, 28*(4), 281-298.

Burnham, K. P., & Anderson, D.R. (2002). *Model selection and multimodel inference: A practical information-theoretic approach* (2nd ed.). New York, NY, USA: Springer-Verlag.

Busby, G., & Albers, H. J. (2010). Wildfire risk management on a landscape with public and private ownership: Who pays for protection? *Environmental Management, 45*(2), 296-310.

Cardille, J. A., Ventura, S. J., & Turner, M. G. (2001). Environmental and social factors influencing wildfires in the Upper Midwest, United States. *Ecological Applications, 11*(1), 111-127.

Castedo-Dorado, F., Rodriguez-Perez, J. R., Marcos-Menendez, J. L., & Alvarez-Taboada, M. F. (2011). Modelling the probability of lightning-induced forest fire occurrence in the province of Leon (NW Spain). *Forest Systems, 20*(1), 95-107.

Chakravarti, I. M., Laha, R. G., & Roy, J. (1967). *Handbook of Methods of Applied Statistics* (Vol. I), New York, NY, USA: John Wiley and Sons.

Crevoisier, C., Shevliakova, E., Gloor, M., Wirth, C., & Pacala, S. (2007). Drivers of fire in the boreal forests: Data constrained design of a prognostic model of burned area for use in dynamic global vegetation models. *Journal of Geophysical Research-Atmospheres, 112*(D24).

Crookston, N.L., & Dixon, G.E. (2005). The forest vegetation simulator: A review of its structure, content, and applications. *Computers and Electronics in Agriculture 49,* 60-80.

Daly, C., Gibson, W. P., Taylor, G. H., Johnson, L. G., & Pasteris, P. (2002). A knowledge-based approach to the statistical mapping of climate. *Climate Research 22,* 99-113.

De'ath, G., & Fabricius, K. E. (2000). Classification and regression trees: a powerful yet simple technique for ecological data analysis. *Ecology, 81*(11), 3178-3192.

Diaz-Avalos, C., Peterson, D. L., Alvarado, E., Ferguson, S. A., & Besag, J. E. (2001). Space-time modelling of lightning-caused ignitions in the Blue Mountains, Oregon. *Canadian Journal of Forest Research-Revue Canadienne De Recherche Forestiere, 31*(9), 1579-1593.

Dickson, B. G., Prather, J. W., Xu, Y., Hampton, H. M., Aumack, E. N., & Sisk, T. D. (2006). Mapping the probability of large fire occurrence in northern Arizona, USA. *Landscape Ecology, 21*(5), 747-761.

Dilts, T. E., Sibold, J. S., & Biondi, F. (2009). A weights-of-evidence model for mapping the probability of fire occurrence in Lincoln County, Nevada. *Annals of the Association of American Geographers, 99*(4), 712-727.

ESRI (1999-2008). ArcMap (computer software). Redlands, CA, USA: ESRI, Inc.

ESRI (1998). *ESRI Shapefile Technical Description*. Retrieved from: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

Finney, M. A. (2006). An Overview of FlamMap Fire Modeling Capabilities. In P. L. Andrews & B. W. Butler (Comps.). *Fuels Management-How to Measure Success: Conference Proceedings* (pp. 213-220) (Proceedings RMRS-P-41). Fort Collins, CO, USA: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station. Retrieved from: http://www.fs.fed.us/rm/pubs/rmrs_p041/rmrs_p041_213_220.pdf.

Finney, M. A. (2002). Fire growth using minimum travel time methods. *Canadian Journal of Forest Research 32*(8), 1420-1424.

Finney, M. A. (1998). *FARSITE: fire area simulator-- model development and evaluation.* (Publication RP-RMRS-4). USDA Forest Service. Retrieved from: http://www.fs.fed.us/rm/pubs/rmrs_rp004.pdf

FireModels.org (2010). FlamMap [computer software]. Retrieved from: http://www.firemodels.org/index.php/flammap-software/flammap-downloads.

Friedman, J. H. (1991). MULTIVARIATE ADAPTIVE REGRESSION SPLINES. *Annals of Statistics, 19*(1), 1-67.

Hansen, J., Sato, M., Ruedy, R., Lo, K., Lea, D. W., & Medina-Elizade, M. (2006). Global temperature change. *Proceedings of the National Academy of Sciences of the United States of America, 103*(39), 14288 –14293.

He, H. S., Shang, B. Z., Crow, T. R., Gustafson, E. J., & Shifley, S. R. (2004). Simulating forest fuel and fire risk dynamics across landscapes - LANDIS fuel module design. *Ecological Modelling, 180*(1), 135-151.

Hulse, D., Branscomb, A., Enright, C., & Bolte, J. (2009). Anticipating floodplain trajectories: a comparison of two alternative futures approaches. *Landscape Ecology, 24*(8), 1067-1090.

Hulse, D., Gregory, S., & Baker, J. (Eds.). (2002). *Willamette River Basin planning atlas: Trajectories of environmental and ecological change*. Corvallis, OR, USA: Oregon State University Press.

Karl, T. R., Melillo, J. M., & Peterson, T. C. (Eds,). (2009). *Unified synthesis product: global climate change impacts in the United States.* New York, NY, USA: Cambridge University Press,. Retrieved from: http://downloads.globalchange.gov/usimpacts/pdfs/climate-impacts-report.pdf.

Kauffman, J. B. (1990). Ecological relationships of vegetation in Pacific Northwest forests. In J. D. Walstad, S. R. Radosevich, & D. B. Sandberg (Eds.), *Natural and prescribed fire in Pacific Northwest forests* (pp. 39-52). Corvallis, OR, USA: Oregon State University Press.

Keeley, J. E., & Fotheringham, C. J. (2003). Impact of past, present, and future fire regimes on North American Mediterranean shrublands. In T. T. Veblen, W. L. Baker, G. Montenegro, & T. W. Swetnam (Eds.), *Fire and climatic change in temperate ecosystems of the western Americas* (pp. 218-262). New York, NY, USA: Springer-Verlag.

Keeley, J. E., & Keely, M. B. (2000). Restoration of smoke-dependent species. *Ecological* Restoration, 18(2), 125-127.

Koutsias, N., Martinez-Fernandez, J., & Allgower, B. (2010). Do factors causing wildfires vary in space? Evidence from geographically weighted regression. *GIScience & Remote Sensing, 47*(2), 221-240.

Krawchuk, M. A., Cumming, S. G., Flannigan, M. D., & Wein, R. W. (2006). Biotic and abiotic regulation of lightning fire initiation in the mixedwood boreal forest. *Ecology, 87*(2), 458-468.

Krawchuk, M. A., Moritz, M. A., Parisien, M. A., Van Dorn, J., & Hayhoe, K. (2009). Global pyrogeography: the current and future distribution of wildfire. *Plos One, 4*(4).

Lenihan, J. M., Bachelet, D., Neilson, R. P., & Drapek, R. (2008). Simulated response of conterminous United States ecosystems to climate change at different levels of fire suppression, CO2 emission rate, and growth response to CO2. *Global and Planetary Change, 64*(1-2), 16-25.

Li, L. M., Song, W. G., Ma, J., & Satoh, K. (2009). Artificial neural network approach for modeling the impact of population density and weather parameters on forest fire risk. *International Journal of Wildland Fire, 18*(6), 640-647.

Loboda, T. V., & Csiszar, I. A. (2007). Assessing the risk of ignition in the Russian Far East within a modeling framework of fire threat. *Ecological Applications, 17*(3), 791-805.

Mann, M. E., Bradley, R. S., & Hughes, M. K. (1998). Global-scale temperature patterns and climate forcing over the past six centuries. *Nature, 392*(6678), 779-787.

Matthews, R. B., Gilbert, N. G., Roach, A., Polhill, J. G., & Gotts, N. M. (2007). Agent-based land-use models: a review of applications. *Landscape Ecology, 22*(10), 1447-1459.

Menge, D. N. L., & Field, C. B. (2007). Simulated global changes alter phosphorus demand in annual grassland. *Global Change Biology, 13*(12), 2582-2591.

Millar, C. I., Neilson R. P., Bachelet, D., Drapek, R., & Lenihan, J. M. (2007). Climate change at multiple scales. In *Forests, carbon and climate change: A synthesis of science findings* (pp. 31-60). Retrieved from: http://www.oregonforests.org/assets/uploads/For_Carbon_fullrpt.pdf

Mladenoff, D. J. (2004). LANDIS and forest landscape models. *Ecological Modelling, 180*(1), 7-19.

Mote, P. W., & Salathé, E. P. (2010). Future Climate in the Pacific Northwest. *Climatic Change 102*, 29-50

Nadeau, L. B., & Englefield, P. (2006). Fine-resolution mapping of wildfire fuel types for Canada: Fuzzy logic modeling for an Alberta pilot area. *Environmental Monitoring and Assessment, 120*(1-3), 127-152.

Nakićenović, N., & Swart, R. (Eds.). (2000). *Special report on emissions scenarios. A special report of working group III of the Intergovernmental Panel on Climate Change.* Cambridge, United Kingdom and New York, NY, USA: Cambridge University Press.

Natarajan, S., Padget, J., & Elliott, L. (2011). Modelling UK domestic energy and carbon emissions: An agent-based approach. *Energy and Buildings, 43*(10), 2602-2612.

Neilson, R.P., & Running, S. W. (1996). Global dynamic vegetation modelling: Coupling biogeochemistry and biogeography models. In: B. Walker & W. Steffen (Eds.), *Global change and terrestrial ecosystems* (pp. 451-465). Cambridge, United Kingdom and New York, NY, USA: Cambridge University Press. Retrieved from: http://secure.ntsg.umt.edu/publications/1996/NR96/Neilson_GlobChgTerrEco_1996.pdf

Nielsen-Pincus, M., Ribe, R., & Johnson, B. (2010). The sociology of landowner interest in restoring fire-adapted, biodiverse habitats in the wildland urban interface of Oregon's Willamette Valley Ecoregion. *Proceedings of the International Association of Wildland Fire second conference on the human dimensions of wildland fire.* San Antonio, Texas.

ODF, (2010). Fire Database: Oregon Department of Forestry commonly requested GIS data sets, protection districts [Data file]. Retrieved from: http://www.oregon.gov/ODF/GIS/gisdata.shtml

Pachauri, R. K., & Reisinger, A. (Eds.). (2007). *Climate change 2007 synthesis report: Contribution of working groups I, II and III to the fourth assessment report of the Intergovernmental Panel on Climate Change*. Retrieved from: http://www.ipcc.ch/publications_and_data/ar4/syr/en/contents.html.

Parisien, M. A., & Moritz, M. A. (2009). Environmental controls on the distribution of wildfire at multiple spatial scales. *Ecological Monographs, 79*(1), 127-154.

Pearce, J., & Ferrier, S. (2000). Evaluating the predictive performance of habitat models developed using logistic regression. *Ecological Modeling, 133*, 225-245.

Peng, C. Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An introduction to logistic regression analysis and reporting. *Journal of Educational Research, 96*(1), 3-14.

Pew, K. L., & Larsen, C. P. S. (2001). GIS analysis of spatial and temporal patterns of human-caused wildfires in the temperate rain forest of Vancouver Island, Canada. *Forest Ecology and Management, 140*(1), 1-18.

Pezzatti, G. B., Bajocco, S., Torriani, D., & Conedera, M. (2009). Selective burning of forest vegetation in Canton Ticino (southern Switzerland). *Plant Biosystems, 143*(3), 609-620.

Pu, R. L., Li, Z. Q., Gong, P., Csiszar, I., Fraser, R., Hao, W. M., . . . Weng, F. Z. (2007). Development and analysis of a 12-year daily 1-km forest fire dataset across North America from NOAA/AVHRR data. *Remote Sensing of Environment, 108*(2), 198-208.

Rapp, V. (2004). *Western forests, fire risk, and climate change*. Science Update 6. Portland, OR: U.S. Department of Agriculture, Forest Service, Pacific Northwest Research Station. Retrieved from: http://www.fs.fed.us/pnw/pubs/science-update-6.pdf).

Reineking, B., Weibel, P., Conedera, M., & Bugmann, H. (2010). Environmental determinants of lightning- v. human-induced forest fire ignitions differ in a temperate mountain region of Switzerland. *International Journal of Wildland Fire, 19*(5), 541-557.

Romero-Calcerrada, R., Barrio-Parra, F., Millington, J. D. A., & Novillo, C. J. (2010). Spatial modelling of socioeconomic data to understand patterns of human-caused wildfire ignition risk in the SW of Madrid (central Spain). *Ecological Modelling, 221*(1), 34-45.

Romero-Calcerrada, R., Novillo, C. J., Millington, J. D. A., & Gomez-Jimenez, I. (2008). GIS analysis of spatial patterns of human-caused wildfire ignition risk in the SW of Madrid (Central Spain). *Landscape Ecology, 23*(3), 341-354.

Rorig, M. L., & Ferguson, S. A. (1999). Characteristics of lightning and wildland fire ignition in the Pacific Northwest. *Journal of Applied Meteorology, 38*(11), 1565-1575.

Rorig, M. L., & Ferguson, S. A. (2002). The 2000 fire season: Lightning-caused fires. *Journal of Applied Meteorology, 41*(7), 786-791.

Rothermel, R. C., (1972). *A mathematical model for predicting fire spread in wildland fuels* (Research paper INT-115). Ogden, UT, USA: U.S. Department of Agriculture, Intermountain Forest and Range Experiment Station.

Sato, H., Itoh, A., & Kohyama, T. (2007). SEIB-DGVM: A new dynamic global vegetation model using a spatially explicit individual-based approach. *Ecological Modelling, 200*(3-4), 279-307.

Sato, H., Kobayashi, H., & Delbart, N. (2010). Simulation study of the vegetation structure and function in eastern Siberian larch forests using the individual-based vegetation model SEIB-DGVM. *Forest Ecology and Management, 259*(3), 301-311.

Scheller, R. M., Domingo, J. B., Sturtevant, B. R., Williams, J. S., Rudy, A., Gustafson, E. J., & Mladenoff, D. J. (2007). Design, development, and application of LANDIS-II, a spatial landscape simulation model with flexible temporal and spatial resolution. *Ecological Modelling, 201*(3-4), 409-419.

Scott, J. H., & Burgan, R. E., (2005). Standard fire behavior fuel models: a comprehensive set for use with Rothermel's surface fire spread model (General technology report RMRS-GTR-153). Fort Collins, CO, USA: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station.

Seidl, R., Fernandes, P. M., Fonseca, T. F., Gillet, F., Jonsson, A. M., Merganicova, K., . . . Mohren, F. (2011). Modelling natural disturbances in forest ecosystems: a review. *Ecological Modelling, 222*(4), 903-924.

Sitch, S., Prentice, I. C., Smith, B., Cramer, W., Kaplan, J., Lucht, W., Sykes, M., Thonicke, K., & Venevsky, S. (2000). *LPJ — A coupled model of vegetation dynamics and the terrestrial carbon cycle.* Sweden: Lund University.

Snider, G., Daugherty, P. J., & Wood, D. (2006). The irrationality of continued fire suppression: An avoided cost analysis of fire hazard reduction treatments versus no treatment. *Journal of Forestry, 104*(8), 431-437.

Stroustrup, B., (2000). *The C++ Programming Language* (3rd edition). Boston, MA, USA: Addison-Wesley Professional.

Sturtevant, B. R., & Cleland, D. T. (2007). Human and biophysical factors influencing modern fire disturbance in northern Wisconsin. *International Journal of Wildland Fire, 16*(4), 398-413.

Syphard, A. D., Radeloff, V. C., Keeley, J. E., Hawbaker, T. J., Clayton, M. K., Stewart, S. I., & Hammer, R. B. (2007). Human influence on California fire regimes. *Ecological Applications, 17*(5), 1388-1402.

Syphard, A. D., Radeloff, V. C., Keuler, N. S., Taylor, R. S., Hawbaker, T. J., Stewart, S. I., & Clayton, M. K. (2008). Predicting spatial patterns of fire on a southern California landscape. *International Journal of Wildland Fire, 17*(5), 602-613.

Therneau, M. T., & Atkison, E. J. (1997). *An introduction to recursive partitioning using the RPART routines.* Retrieved from: http://www.mayo.edu/hsr/techrpt/61.pdf.

Thonicke, K., Venevsky, S., Sitch, S., & Cramer, W. (2001). The role of fire disturbance for global vegetation dynamics: coupling fire into a dynamic global vegetation model. *Global Ecology and Biogeography, 10*(6), 661-677.

Topping, C. J., Hansen, T. S., Jensen, T. S., Jepsen, J. U., Nikolajsen, F., & Odderskaer, P. (2003). ALMaSS, an agent-based model for animals in temperate European landscapes. *Ecological Modelling, 167*(1-2), 65-82.

Urban, L.D (2002). Classification and regression trees. In B. McCune & J. Grace (Eds.). *Analysis of ecological communities*. MjM Software Design, Gleneden Beach, OR.

Vasilakos, C., Kalabokidis, K., Hatzopoulos, J., & Matsinos, I. (2009). Identifying wildland fire ignition factors through sensitivity analysis of a neural network. *Natural Hazards, 50*(1), 125-143.

Vazquez, A., & Moreno, J. M. (2001). Spatial distribution of forest fires in Sierra de Gredos (Central Spain). *Forest Ecology and Management, 147*(1), 55-65.

Vega-Garcia, C., & Chuvieco, E. (2006). Applying local measures of spatial heterogeneity to Landsat-TM images for predicting wildfire occurrence in Mediterranean landscapes. *Landscape Ecology, 21*(4), 595-605.

Voulgarakis, A., & Shindell, D. T. (2010). Constraining the sensitivity of regional climate with the use of historical observations. *Journal of Climate, 23*(22), 6068-6073.

Weber, M. G., & Stocks, B. J. (1998). Forest fires and sustainability in the boreal forests of Canada. *Ambio, 27*(7), 545-550.

Yang, J., He, H. S., Shifley, S. R., & Gustafson, E. J. (2007). Spatial patterns of modern period human-caused fire occurrence in the Missouri Ozark Highlands. *Forest Science, 53*(1), 1-15.

Zadeh, L. A. (1983). The role of fuzzy-logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems, 11*(3), 199-227.

Zaehle, S., Bondeau, A., Carter, T. R., Cramer, W., Erhard, M., Prentice, I. C., . . . Sykes, M. (2007). Projected changes in terrestrial carbon storage in Europe under climate and land-use change, 1990-2100. *Ecosystems, 10*(3), 380-401.

Zemp, M., Haeberli, W., Hoelzle, M., & Paul, F. (2006). Alpine glaciers to disappear within decades? *Geophysical Research Letters, 33*(13).

Zhang, Z. X., Zhang, H. Y., & Zhou, D. W. (2010). Using GIS spatial analysis and logistic regression to predict the probabilities of human-caused grassland fires. *Journal of Arid Environments, 74*(3), 386-393.