Modernizing legacy desktop applications: Re-engineering WinDAM C from VB6 to WinUI 3

by

Samhitha Burugupalli

B.Tech., Andhra University, 2021

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2025

Approved by:

Major Professor
Dr. Mitchell L. Neilsen

Copyright

Abstract

This study presents the modernization of the Windows Dam Analysis Modules for Cohesive Embankments (WinDAM C), a key component of the WinDAM suite developed by the U.S. Department of Agriculture–Agricultural Research Service (USDA-ARS) and Kansas State University for simulating breach and erosion processes in earthen embankment dams. The original WinDAM C, implemented in Visual Basic 6 (VB6), was constrained as a 32-bit executable, obsolete deployment methods, and tightly coupled code structures that limited maintainability, interoperability, and accessibility. The principal objective of this work is to re-engineer WinDAM C into a modern, sustainable, and reproducible platform using C# and the WinUI3 framework. The new system adopts a Model–View–ViewModel (MVVM) architecture for clear separation of logic and presentation, integrates the NRCS Engineering Field Handbook Chapter 2 (EFH-2) hydrologic methodology for direct rainfall–runoff generation through WinTR-20 coupling, and employs the WiX Toolset for deterministic deployment without external dependencies. Validation experiments focused on verifying numerical reproducibility and scientific consistency between the legacy 32-bit implementations and the new 64-bit implementations. Batch simulations of ten benchmark WinDAM Control (WDC) projects confirmed that breach hydrograph results remained consistent across both environments, with deviations below 0.05 percent—attributable solely to floating-point precision and rounding differences between 32-bit floats and 64-bit doubles. The modernization effort further establishes a foundation for accessibility and future extensibility. Leveraging WinUI's native automation, the interface now supports system-wide text scaling, high-contrast adaptation—representing the first step toward Section 508 compliance within the WinDAM suite. Additional forward-looking features include JSON-based input/output for data interoperability with GIS and cloud platforms,

and a modular architecture designed for future spatial visualization and expanded hydrologic coupling. Collectively, these advancements transform WinDAM C from an aging desktop utility into a reproducible, maintainable, and extensible research application that preserves the validated USDA-ARS breach-erosion algorithms while unifying hydrologic input generation, simulation, and analysis. Contributions of this work include both a practical tool for dam-safety engineers and a generalizable framework for modernizing legacy scientific software systems.

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

1.1 Background and Motivation

Earthen embankment dams are among the oldest hydraulic structures created by humans. They impound, divert, and regulate water for multiple purposes—flood control, irrigation, municipal supply, navigation, hydroelectric generation, recreation, and habitat preservation. Their construction history spans more than four millennia: archaeological evidence traces compacted-earth embankments to Mesopotamia and China, where early civilizations used rudimentary spillways to regulate seasonal flooding and support agriculture.

The International Commission on Large Dams (ICOLD) maintains the World Register of Dams, which lists more than 62,000 large dams across 100 countries [4]. In the United States, the National Inventory of Dams (NID) reports roughly 91,000 structures, of which about two-thirds are earthen embankments [5]. These statistics underscore the global dependence on this dam type because of its relatively low cost and adaptability to varied geologic settings.

However, the same properties that make earthen dams attractive also render them vulnerable to characteristic failure mechanisms. Overtopping erosion occurs when flow exceeds spillway capacity, eroding the downstream face; internal erosion or piping originates from concentrated leaks or cracks through the core or foundation; slope instability and seismic shaking can further compromise integrity [4]. Although failures are rare relative to total inventory, they are often catastrophic. Syntheses of historical incidents show that overtopping, foundation defects, and internal erosion each account for roughly one-third of recorded failures [6, 7].

The social and economic consequences of dam failures are well documented. The 1889 Johnstown Flood following the South Fork Dam breach in Pennsylvania killed more than 2,200 people [8]. The 1976 Teton Dam collapse in Idaho released nearly 300,000 acre-feet of water,

inundating 180 square miles and causing losses exceeding $2 billion [9]. More recently, the 2019 Brumadinho tailings-dam disaster in Brazil killed 270 people and contaminated the Paraopeba River [37]. These examples highlight the continuing need for reliable predictive tools to evaluate and mitigate dam-failure risks.

Climate change amplifies these concerns. More frequent and intense rainfall increases overtopping likelihood, while alternating drought and flood cycles promote desiccation cracking and rapid saturation that destabilize slopes [38,39]. In the U.S., the average dam age now exceeds 50 years, compounding maintenance and risk-management challenges.

Because field inspection and physical testing cannot encompass all possible failure conditions, computational models are indispensable to dam-safety assessment. They allow simulation of hypothetical breach scenarios under controlled assumptions, producing hydrographs that can be routed downstream to estimate inundation, potential loss of life, and mitigation effectiveness. These digital tools have become essential to engineers and emergency-management agencies.

1.2 The WinDAM Suite and WinDAM C

One of the most comprehensive frameworks for physically based dam-breach simulation is the Windows Dam Analysis Modules (WinDAM) suite, developed by the U.S. Department of Agriculture's Agricultural Research Service (USDA-ARS) in collaboration with the Natural Resources Conservation Service (NRCS) and Kansas State University [3, 15]. Initiated in the early 2000s, WinDAM integrates decades of hydraulic-flume experiments at the USDA Hydraulic Engineering Research Unit in Stillwater, Oklahoma—particularly the headcut-erosion

studies by Hanson and Temple [15]—with field case histories to produce physically based erosion and breach models for cohesive and non-cohesive embankments.

The suite consists of three coordinated modules. WinDAM A simulates overtopping erosion of homogeneous, non-cohesive embankments such as sandy or silty dams; WinDAM B extends analysis to zoned embankments and vegetated or armored spillways, capturing the protective effect of surface layers before core erosion begins [16, 17]; and WinDAM C focuses on cohesive embankments with significant clay content, which resist erosion under moderate stresses but fail rapidly once headcuts or internal conduits form [18].

WinDAM C (WinDAM Cohesive) requires detailed input describing dam geometry, soil properties, and hydraulics. Parameters include height, crest length, slope angles, zoning, unit weight, cohesion, shear strength, erodibility coefficients such as critical shear stress τc and detachment rate Kd, reservoir levels, inflow hydrographs, and potential defects such as cracks or conduits. Outputs include time-series breach hydrographs, peak discharge and timing, breach-geometry evolution, and progressive erosion profiles.

Validation studies by USDA-ARS compared WinDAM C predictions with laboratory headcut experiments and documented failures, showing good agreement within expected engineering tolerances [15, 16, 17, 3]. The original WinDAM C graphical interface was written in Microsoft Visual Basic 6 (VB6), which was practical in the early 2000s but now obsolete.

1.3 Integration of EFH-2 Hydrologic Modeling

A key enhancement in modernization is the integration of the NRCS Engineering Field Handbook Chapter 2 (EFH-2) methodology for generating hydrologic inputs. In the legacy system, inflow hydrographs were manually specified from external analyses such as WinTR-20,

introducing inconsistency and transcription errors. The EFH-2 method, by contrast, computes design hydrographs directly from rainfall and watershed characteristics—drainage area, RunOff Curve Number (RCN), time of concentration (Tc), and rainfall distribution—using the NRCS Curve-Number approach and dimensionless unit hydrographs [20].

When coupled with WinTR-20, EFH-2 produces hydrographs for standard design conditions: the Principal Spillway Hydrograph (PSH), Stability Design Hydrograph (SDH), Freeboard Hydrograph (FBH), and optional user-defined events. Embedding EFH-2 inside WinDAM C ensures that every erosion simulation begins from hydrologically defensible inputs derived from nationally standardized NRCS procedures [35].

1.4 Problem Statement

Although the physical algorithms of WinDAM C remain scientifically valid, its VB6 implementation has reached the end of maintainability. The program is confined to 32-bit execution, cannot exploit multi-core processors, and depends on deprecated COM components. The tightly coupled structure merges interface logic, file parsing, and computation, making updates cumbersome and error-prone. Its installer depends on obsolete Windows libraries and frequently fails on current systems.

Equally limiting is the lack of accessibility support: the interface does not expose control names for screen readers, lacks keyboard navigation, and fails under high-contrast themes, making it non-compliant with Section 508. The proprietary .WDC file structure hinders integration with modern workflows and cloud systems. Without modernization, WinDAM C risks obsolescence—jeopardizing decades of USDA-ARS research and diminishing dam-safety analysis capacity as infrastructure ages and hydrologic extremes intensify.

1.5 Research Objectives

The objective of this research is to modernize and extend WinDAM C while preserving its validated computational core. Specifically, this work:

1. Re-implements the graphical interface in C# using the WinUI 3 framework and Model–View–ViewModel (MVVM) architecture to support 64-bit systems, improve maintainability, and separate logic from presentation.
2. Integrates EFH-2 hydrology for automatic hydrograph generation consistent with NRCS standards.
3. Validates numerical reproducibility between legacy and modernized versions using automated numdiff comparison of .WDC and .OUT files.
4. Develops a deterministic deployment package using the WiX Toolset for stable, dependency-free installation.
5. Evaluates preliminary accessibility behavior and establishes a framework for future compliance.

1.6 Significance of the Study

Scientifically, this modernization preserves the USDA-ARS cohesive-embankment breach models and extends their usable life. Technically, it contributes a replicable methodology for legacy scientific-software modernization—illustrating how algorithmic integrity can be maintained while updating architecture and interface. Practically, it provides a reliable, accessible tool for engineers and agencies engaged in dam-safety evaluation.

By linking EFH-2 hydrology directly with WinDAM C, the new system unifies rainfall–runoff generation with erosion simulation, ensuring consistent and traceable analyses across varying climatic regimes. Automated reproducibility testing guarantees that the modernization introduces no scientific drift beyond floating-point precision.

1.7 Organization of the Thesis

This thesis is organized into five chapters. Chapter 2 reviews the scientific and technical background of dam-breach modeling, EFH-2 hydrology, and software-modernization practices. Chapter 3 describes the WinDAM C system architecture and modernization process, including EFH-2 integration, reproducibility testing, and deployment. Chapter 4 presents validation and accessibility results. Chapter 5 provides conclusions and recommendations for future work.

Chapter 2 - Technical and Scientific Background

2.1 Overview

Modernizing the Windows Dam Analysis Model for Cohesive Embankments (WinDAM C) lies at the intersection of two domains: (1) physically based dam-breach and erosion modeling and (2) sustainable modernization of legacy scientific software. This chapter reviews the scientific foundations of cohesive-embankment breach modeling, the development of the USDA-ARS WinDAM suite, the hydrologic framework of the NRCS Engineering Field Handbook Chapter 2 (EFH-2), and strategies for re-engineering 32-bit VB6 systems into modern, reproducible platforms.

2.2 Evolution of Dam Breach Modeling

2.2.1 From Empirical to Physically Based Approaches

Early breach prediction relied on empirical regressions between observed failures and geometric parameters [6]. Such equations provided rapid estimates of breach width and peak discharge but ignored the governing hydraulics and soil mechanics. The transition to physically based modeling began when hydraulic shear stress and soil resistance were incorporated explicitly [15].

Laboratory studies at the USDA Hydraulic Engineering Research Unit (HERU) quantified the detachment rate of cohesive soils using jet-erosion tests [18–20]. These experiments established the excess-shear relationship

$$\varepsilon_r = K_d(\tau - \tau_c)$$

expresses erosion rate (ε_r) is the erosion rate, τ the applied shear stress, τ_c the critical shear, and K_d a material coefficient.

Temple et al. [15] and Hanson et al. [16] integrated these findings into deterministic breach formulations forming the basis of the WinDAM suite.

Parallel research extended the framework to probabilistic and simplified models. Wu [1, 2] developed DLBreach, while Zhong et al. [23, 25] compared multiple simplified physically based approaches, confirming the dominant role of τc and Kd in controlling breach progression. Recent work by Atkinson and Neilsen [30] applied formal uncertainty analysis to WinDAM C and DLBreach, quantifying how 32-bit versus 64-bit arithmetic affects computed hydrographs.

2.2.2 Laboratory and Field Validation

Large-scale flume tests conducted by Hanson, Temple, and Visser [16, 17] reproduced overtopping and internal-erosion scenarios in cohesive materials. These experiments provided quantitative data for calibrating WinDAM C. Later validation using field cases and auxiliary-spillway studies [3, 10] confirmed that model predictions agree with measured erosion depths and peak discharges within accepted engineering tolerances. The consistency of these results underpins USDA-ARS confidence in the physical formulations preserved in the modernized code.

2.3 Development of the WinDAM Suite

2.3.1 Origins and Structure

The WinDAM suite was created by USDA-ARS in collaboration with NRCS and Kansas State University to translate laboratory research into operational software [15]. It comprises:

- WinDAM A – non-cohesive overtopping erosion;
- WinDAM B – zoned or vegetated auxiliary spillways;
- WinDAM C – cohesive embankments with potential internal erosion [3].

Each module uses plain-text input/output files to maintain transparency and reproducibility

2.3.2 Erosion Mechanics and Formulation

WinDAM C computes flow hydraulics across the evolving breach, evaluates local shear stress, and applies the excess-shear law to each soil zone. Layer-specific erodibility allows simulation of zoned cores, protective veneers, or natural armoring. Because all computations are deterministic, reproducibility depends primarily on floating-point precision and consistent input serialization—topics examined later in Chapters 3 and 4.

2.3.3 Data and File Conventions

WinDAM projects use human-readable files: .WDC (project definition), .OUT (full time-series results), .DIS (summary for interface display), .D2D (spillway input), and .DG# (graph data). This explicit architecture allows full audit, rerun, or automated comparison [3].

2.4 Hydrologic Foundations and EFH-2 Integration

2.4.1 Need for Hydrologic Consistency

In the legacy workflow, inflow hydrographs were imported manually from external models such as WinTR-20 or HEC-HMS, creating opportunities for inconsistency. Embedding

NRCS EFH-2 procedures within WinDAM C ensures that runoff and rainfall inputs are derived from standard NRCS methods [35].

2.4.2 FH-2 and WinTR-20 Methods

EFH-2 computes direct-runoff depth (Q) and peak discharge (q_p) from rainfall (P) and watershed parameters via the Curve-Number (CN) method:

$$Q = (P-0.2S)^2 / (P+0.8S) \quad , \quad S = (1000/CN) - 10$$

These data are routed through WinTR-20 using regional rainfall distributions and dimensionless unit hydrographs to generate complete inflow hydrographs [35].

2.4.3 Integration Workflow

When EFH-2 is enabled, the user specifies watershed parameters and desired design hydrographs—Principal Spillway (PSH), Stability Design (SDH), Freeboard (FBH), or User Defined. WinTR-20 calculates the corresponding discharges, which WinDAM C inserts automatically into the [HYD] blocks of the .WDC file. This ensures that hydrologic, geometric, and material parameters remain synchronized within one dataset.

2.5 Limitations of the Legacy VB6 System

Visual Basic 6 was discontinued in 2008 [34], confining WinDAM C to 32-bit runtimes and obsolete COM dependencies. Its monolithic event-driven structure intertwined interface and computation, impeding testing and modernization. Accessibility was nonexistent, violating Section 508 requirements. Such technical debt [32] threatened long-term reproducibility and compatibility with modern Windows environments.

2.6 Modernization Strategies for Scientific Software

2.6.1 Incremental Re-Engineering

Modernization approaches include complete rewrite or incremental refactor [33]. For WinDAM C, incremental re-engineering preserved validated USDA-ARS algorithms while reconstructing the user interface and file handling in C# and WinUI 3. This strategy reduces risk, maintains scientific credibility, and enables modular maintenance using MVVM architecture [36].

2.6.2 Deployment and Accessibility

Deployment was re-engineered through the WiX Toolset to produce deterministic installers free of legacy dependencies. Modern WinUI 3 features—automatic text scaling, high-contrast adaptation, and UI Automation—lay the groundwork for Section 508 and WCAG 2.2 compliance [29]. Numerical reproducibility follows Atkinson and Neilsen (2024) by using numdiff comparisons between 32- and 64-bit executables.

2.6.3 Emerging Trends

Modern research computing increasingly employs containerization [31] and automated validation pipelines. Future WinDAM C extensions may leverage these to support distributed simulations, GIS integration, and cloud-based collaboration [29].

2.7 Summary

This chapter reviewed the historical and technical context for WinDAM C modernization: the evolution from empirical to physically based erosion models [6, 15, 18],

USDA-ARS validation efforts [3, 16, 17], integration of EFH-2 hydrology [35], and modern software-engineering strategies [33, 32]. Together these foundations justify re-engineering WinDAM C to preserve its validated algorithms while achieving sustainability, accessibility, and reproducibility. The next chapter details the system architecture and data flow of the modern implementation.

Chapter 3 - System Architecture and Execution Flow

3.1 Overview

This chapter explains how the WinDAM C program is organized and how data move through it during a simulation. It focuses on the practical structure of the system—the sequence in which files are created, processed by simulators, and displayed in the user interface. The purpose is to document how the program operates and how these operations were refined in the updated version.

WinDAM C is used to model erosion and breach of earthen embankment dams. It relies on two companion simulators developed by the U.S. Department of Agriculture–Agricultural Research Service (USDA-ARS):

- WinDAMSim.exe, which performs the main dam-breach calculations, and
- SitesSim.exe, which analyzes erosion in auxiliary spillways.

Both simulators exchange information through plain-text files, allowing the entire process to be reproduced or reviewed at any stage. The diagrams and screenshots referenced in this chapter show how these files are created, used, and stored as part of a complete project.

Figure 1 illustrates the overall data flow between the WinDAM C interface and the two simulators. Figures 2, 3, and 4 show the arrangement of the summary table and user-interface screens used for reviewing results and entering data. Figure 5 shows the EFH-2 hydrology screen that connects WinDAM C with WinTR-20, the program that generates rainfall–runoff hydrographs following the Engineering Field Handbook (EFH-2) procedures.

Together, these figures describe how WinDAM C transforms input data into numerical and graphical results and how the same process has been reorganized in the modernized version.

3.2 Legacy System Architecture

The flow diagram in Figure 1 shows how data move through the original WinDAM C system—from the user interface to the simulation programs and back to the display of results. Every stage exchanges information through plain-text files that are stored in the project directory.

The process is linear and transparent: each simulator reads a specific file, performs its calculations, and writes a new file that becomes the input for the next stage.

3.2.1 Input Preparation and Project Definition

Users enter all model parameters—geometry, soil properties, hydrology, and spillway details—through the VB6 interface. When Save As is selected, WinDAM C creates a single project definition file with the file extension of WDC. This text file includes all data necessary for a complete analysis, organized into keyword-delimited blocks such as [OPTION], [HCMODEL], [HYD], [CRESTPRFL], [ENDTABLE], [UPSTREAM], [DAMCREST], [DWNSTREAM], [PSpINLET], [PSpDATA], [PSpCOEFFS], [TAILWATER], [ASSURFACE], [SSEA], [ASSPRFL], [ASSURFACE], [ASDATA], [BTMWIDTH], [ASMATERIAL] and [ASCOORD]. The .WDC file serves as the main project definition and contains every parameter required for the dam-level simulation.

Each auxiliary spillway is described by a separate SSEA (Soil-Spillway Erosion Analysis) block within the same file.

3.2.2 Dam-Level Simulation with WinDAMSim.exe

The first step is the main embankment analysis performed by WinDAMSim.exe.

The interface issues a command such as:

WinDAMSim.exe "Example10.WDC"

WinDAMSim.exe reads the .WDC, performs the erosion and breach calculations, and writes a single output file—Example10.OUT—to the project folder. This .OUT file contains all computed time-series data such as reservoir elevation, discharge, headcut depth, and breach width for each time step.

After the simulation finishes, WinDAM C parses the .OUT file internally and extracts the most important results (peak inflow, peak outflow, maximum pool elevation, and key warnings). These values are saved in a condensed file named Example10.DIS, which is also placed in the project folder. The .DIS file is used solely by the interface for quick loading of tables and graphs; it is not produced directly by WinDAMSim.exe.

3.2.3 Generation of .D2D Files for Spillway Analysis

After the dam-level computation is complete, WinDAM C examines the .WDC for SSEA blocks and creates a corresponding .D2D file for each auxiliary spillway. Each .D2D file merges two information sources: The geometric, soil, and material properties from the related SSEA section of the .WDC; and The discharge–time data Hydrograph extracted from the dam-level .OUT file. Together, these define the boundary conditions for spillway erosion modeling. All spillway-related files are placed in a subfolder named after the .WDC file (for example, a project Example10.WDC creates a folder Example10/). This organization keeps the dam-level and spillway analyses separate while maintaining clear traceability.

3.2.4 Auxiliary-Spillway Simulation with SitesSim.exe

Each .D2D file in the subfolder is processed by SitesSim.exe, the program that models headcut migration and erosion depth along the auxiliary spillway channel. For every spillway, SitesSim.exe produces:

Table 1. Files Generated by SitesSim.exe

File	Description	Location
.OUT	Full text report of erosion depth, headcut position, and flow duration	Subfolder
.DIS	Condensed summary file created by WinDAM C after parsing the .OUT	Subfolder
.DG#	Numeric datasets for erosion-depth and headcut plots	Subfolder
.DHY	Hydrograph record for spillway outflow	Subfolder

These files collectively describe the hydraulic and erosion response of each auxiliary spillway.

3.2.5 Integration Back into the Interface

When all analyses are finished, the interface scans both the main project folder and its spillway subfolders for new .DIS files. The results are read and displayed in the Summary Table, which lists for each run: Peak inflow and outflow values, Maximum pool and breach elevations, Erosion depths for each spillway, and The number of warnings or errors. Selecting View Text opens the corresponding .OUT file in a text viewer, while View Graphs displays curves created from .DG# and .DHY. Because all files are plain text, users can verify results directly or compare them across runs.

3.2.6 Summary of Flow

The overall data sequence can be summarized as follows:

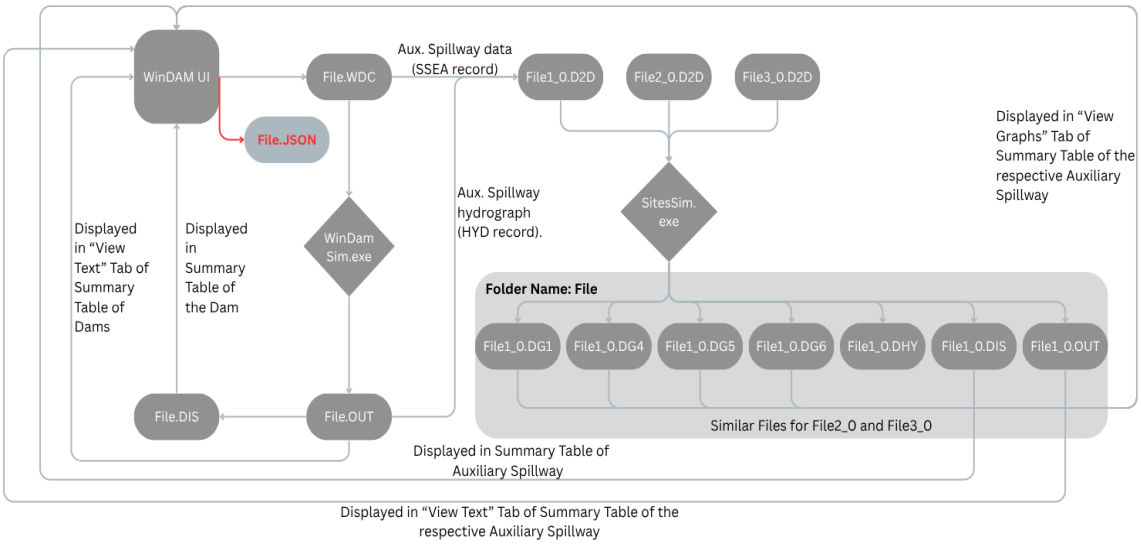


Figure 1. Overall data flow between the WinDAM C interface, WinDAMSim, and SitesSim (red indicates additions in new application)

This architecture ensured that every run left a complete, inspectable set of files and that all results could be regenerated from the same inputs. Its main limitation was that the user had to manage many text files and subfolders manually. The updated system keeps the same logical

sequence but automates the file handling, ensuring consistent paths and names for every simulation.

3.3 User Interface and Input Arrangement

The WinDAM C interface provides a structured environment for entering project data, running simulations, and viewing results. The design follows a file-driven architecture, meaning each screen directly corresponds to data stored in the project folder. Figures 2, 3, 4 show the two main windows of the legacy program: the input panels used to define a model and the Summary Table used to review simulation results.

3.3.1 Input Organization

Model input is arranged in a sequence of tabbed screens. Each tab represents one group of parameters—project information, geometry, material properties, spillway characteristics, or hydrologic inputs—and directly maps to a section of the .WDC file. When the user selects Run Simulation, all tab values are serialized in that order, producing the input file used by WinDAMSim.exe.

This arrangement makes the interface a visual editor for the text-based input structure. Conditional visibility ensures that only the relevant tabs appear; for example, principal-spillway options are hidden when the selected analysis does not require them. The layout minimizes data-entry errors and maintains one-to-one correspondence between what the user sees on screen and what is written to the .WDC file

3.3.2 Summary Table and Results Display

After the simulation is complete, WinDAM C switches to the Summary Table view.This window contains two levels of data presentation:

The upper table lists dam-level results, one row for each .WDC file in the project folder. These results are extracted from the .OUT file produced by WinDAMSim.exe and summarized into a .DIS file for faster display.

The lower table, labeled Auxiliary Spillway Summary Table, displays the outputs from the second-level subfolders. Each row represents an auxiliary spillway analyzed by SitesSim.exe. The program automatically scans the subfolder named after the .WDC (for example, Example10/) and loads the .DIS files created from the spillway .OUT results into this table.

Together, these two linked tables allow the user to view both the main dam analysis and all related spillway erosion results in a single interface. The View Text and View Graphs buttons open the associated .OUT or .DG#/.DHY files directly for detailed review.

3.3.3 Relationship Between Interface and File System

Every operation performed in the interface corresponds to a change in the file system. Creating a new project generates a .WDC; running a simulation adds a .OUT; and parsing that result produces a .DIS file for display. When auxiliary analyses are completed, their results are saved in the project's subfolder, but are automatically read and displayed in the second-level spillway table of the Summary window.

This design allowed WinDAM C to present all numerical results without requiring a database or background services. The close link between the interface and file structure ensured

that projects could be copied, reopened, or re-run on any computer with identical outcomes—a characteristic that was preserved during modernization.

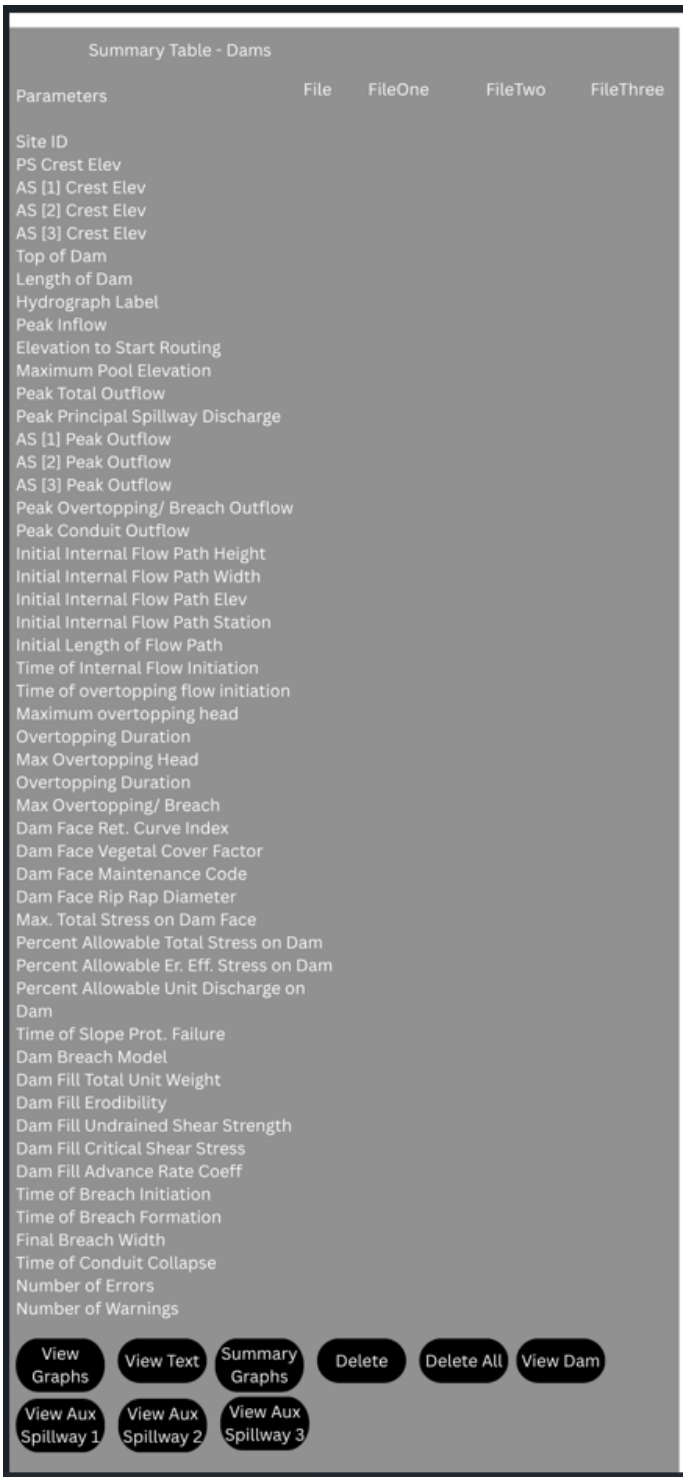


Figure 2. Summary Table of the WinDAM C interface showing dam-level results.

Figure 3. Summary Table of the WinDAM C interface showing auxiliary-spillway results retrieved from subfolder .DIS files.

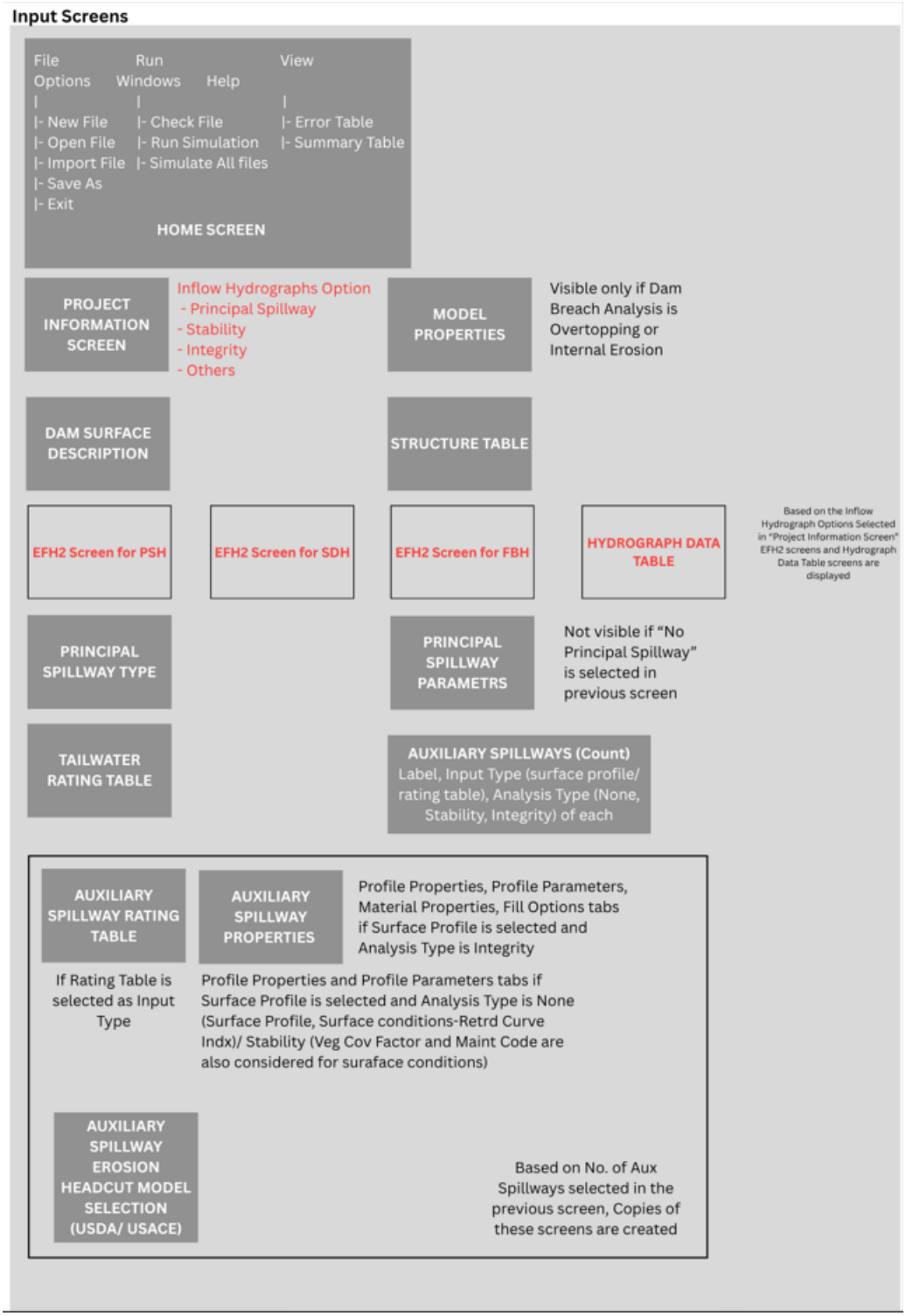


Figure 4. Input-screen layout of the legacy WinDAM C interface with tabbed data-entry panels corresponding to .WDC file sections.

3.3.4 Data Processing and Visualization

The modern interface reads .OUT and .DIS results directly into memory. Peak discharge, breach time, and erosion statistics update dynamically in the UI. Graphs are rendered with OxyPlot, avoiding intermediate data files.

All data are stored in a lightweight .json file, ensuring that every simulation can be reconstructed exactly. This dual text-plus-JSON serialization supports both backward compatibility and future integration with web or cloud analysis frameworks [1, 23].

Hydrologic consistency is central to WinDAM C's predictive accuracy. The legacy system relied on externally prepared hydrographs, often created in WinTR-20, and manual file imports. The modern system embeds EFH-2 logic directly in its workflow [35].

When enabled, EFH-2 input panels appear dynamically, allowing users to define rainfall depth, curve number, drainage area, and time of concentration. WinTR-20 is then executed as a sub-process to generate standardized NRCS hydrographs for Principal Spillway, Stability Design, and Freeboard conditions. These are written automatically into the [HYD] block of the .WDC.

This integration ensures all design hydrographs originate from validated NRCS methods, eliminating the manual data-entry errors common in the older workflow.

3.4 EFH-2 Hydrology and Data Flow

Hydrologic input represents one of the most critical components of dam-breach and stability simulations. In the original WinDAM C framework, these data were not produced within the application but had to be imported from other models such as NRCS WinTR-20, HEC-HMS, or SITES. Each hydrograph was prepared externally, exported as a tab-delimited

text file, and then imported into WinDAM C to populate the [HYD] block of the project's .WDC file.

While functional, this procedure introduced several sources of inconsistency. The hydrograph was separated from the watershed information that produced it, and any modification to upstream parameters—such as drainage area, rainfall depth, or curve number—required regeneration of the input file. As a result, hydrologic data were often out of sync with the geometric and material properties defined elsewhere in the project.

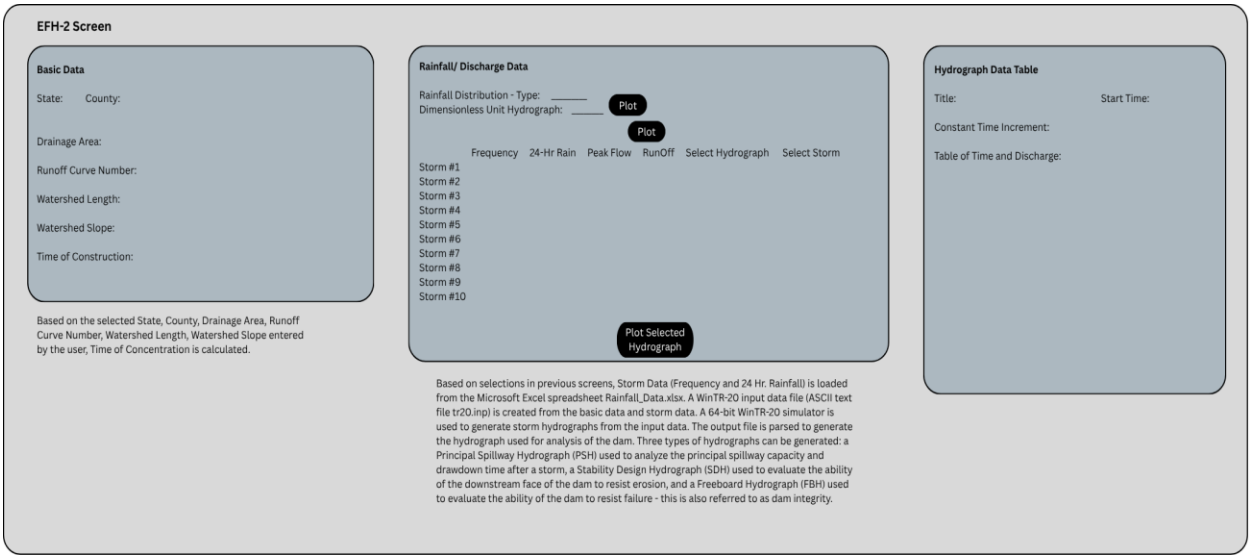


Figure 5. EFH-2 UI Design

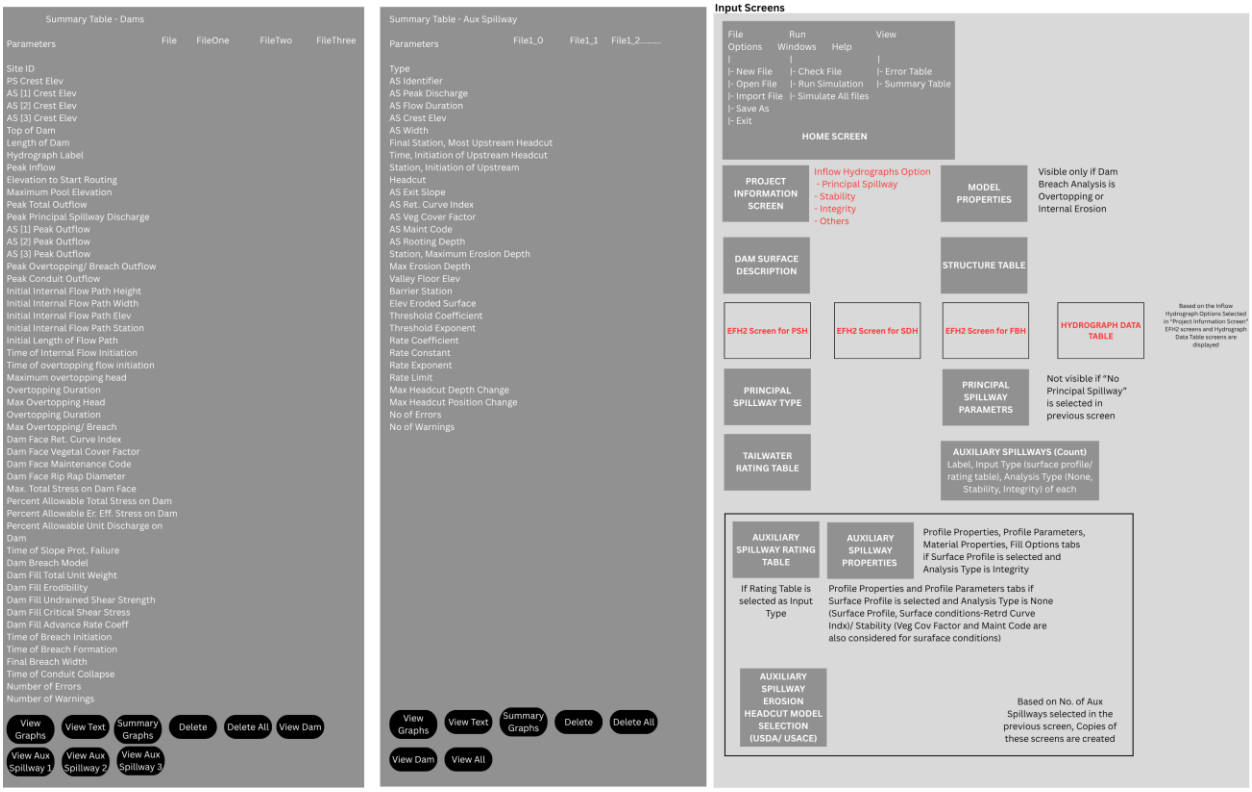


Figure 6. EFH-2 Integration into WinDAM C

3.4.1 Conditional Integration of EFH-2 Hydrology

The modernized system incorporates rainfall–runoff generation directly through the Engineering Field Handbook Part 2 (EFH-2) methodology, implemented using the NRCS WinTR-20 engine.

To maintain compatibility with existing workflows, hydrology generation is not always mandatory. The program determines the visibility of EFH-2 modules based on the parameters defined in the Project Information stage. If an existing hydrograph or time-series dataset is already available, the project configuration suppresses EFH-2 input panels and uses the provided data.

When hydrologic computation is required, the EFH-2 subsystem becomes active and allows the specification of one or more design conditions. These conditions correspond to the standard NRCS design hydrographs: the Principal Spillway Hydrograph (PSH), Stability Design Hydrograph (SDH), Freeboard Hydrograph (FBH), and additional user-defined scenarios. Each can be enabled independently, reflecting the fact that multiple hydrologic design cases may be analyzed for a single dam rather than a single exclusive option. This configuration flexibility mirrors practical engineering procedures and supports comprehensive scenario evaluation.

3.4.2 Automated Data Flow

Once activated, the EFH-2 module establishes a direct computational link with the WinTR-20 hydrology engine. Watershed and rainfall descriptors—such as drainage area, curve number, slope, and rainfall distribution—are transferred automatically to WinTR-20, which calculates the corresponding rainfall–runoff relationship. The resulting discharge–time data are then returned to WinDAM C and written directly into the [HYD] section of the .WDC file.

Each hydrologic scenario is recorded as an independent block, containing the scenario identifier, time step, and discharge values in consistent units. These [HYD] records follow the same plain-text, tagged format as other WinDAM input sections, ensuring transparency and version-control compatibility.

During a simulation, WinDAMSim.exe reads the block corresponding to the active design condition and applies it as the inflow boundary condition for reservoir routing and erosion modeling.

3.4.3 Synchronization and Consistency

Integrating EFH-2 hydrology within the main program ensures that hydrologic inputs remain synchronized with all other model parameters. Any revision of watershed characteristics or rainfall assumptions triggers a corresponding update of the associated hydrograph, eliminating discrepancies between data sources.

Because the generated hydrographs are stored in text form alongside the rest of the project data, the entire hydrologic derivation can be reconstructed from a single file without external dependencies. This linkage unifies watershed-scale and structure-scale modeling, producing a coherent dataset for reproducible simulation.

3.4.4 Advantages of the Integrated Approach

The embedded EFH-2 implementation provides several scientific and operational benefits:

- Standardization: All hydrographs are derived using NRCS-approved WinTR-20 procedures, ensuring methodological consistency.
- Traceability: Watershed parameters and computed inflows reside within the same project file, providing a complete audit trail of each analysis.
- Reproducibility: Identical inputs yield identical hydrographs across computing environments, removing ambiguity caused by external file handling.
- Efficiency: Multiple design hydrographs can be produced within a single configuration, reducing setup time and simplifying comparative assessments.

3.5 Modern Implementation in C# and WinUI 3

The modernization of WinDAM C replaced the legacy Visual Basic 6 platform with a sustainable 64-bit implementation written in C# using the WinUI 3 framework under the Windows App SDK. The new architecture preserves the verified USDA-ARS breach and erosion algorithms while re-engineering all supporting components for long-term maintainability, interoperability, and accessibility. Modern data serialization through both tagged-text (.WDC) and structured JSON formats enables reproducible workflows and integration with automated testing tools.

3.5.1 Architectural Design

The re-engineered system follows the Model–View–ViewModel (MVVM) pattern, separating data, logic, and presentation to achieve clarity and modularity.

Model layer – Defines all entities required for simulation—dam geometry, spillway characteristics, material parameters, hydrologic inputs, and results. Each entity can serialize to either the legacy WDC text format or an equivalent .json representation, ensuring compatibility with older datasets while supporting modern data exchange and version control.

ViewModel layer – Manages execution flow, command bindings, and file I/O. It converts model data into serialized files, launches simulators, monitors execution, and parses resulting outputs. Observable properties automatically update the View when values change.

View layer – Implemented in WinUI 3 XAML, providing the user interface for data entry, simulation control, and output visualization. The View responds to theme, font-size, and contrast settings, supporting a consistent and accessible presentation.

This layered structure replaces the monolithic event-driven VB6 code with modular assemblies that can be maintained or extended independently.

3.5.2 File Workflow and Simulation Control

WinDAM C retains its file-based reproducibility while introducing modern serialization methods. When a project is executed, all parameters are written simultaneously to two formats:

1. the legacy .WDC text file for compatibility with existing simulators, and
2. a corresponding .JSON file used for storing, loading, and potential cloud workflows.

The ViewModel then launches WinDAMSim.exe as a 64-bit process. Upon completion, the resulting .OUT file is parsed, condensed into .DIS, and displayed. If auxiliary spillways are present, .D2D input files are generated and processed by SitesSim.exe, with outputs stored in a subfolder named after the parent .WDC file. This organization maintains the transparent directory structure required for reproducibility while enabling external tools to interact with the same data through JSON.

3.5.3 Key Outcomes

The C# and WinUI 3 implementation yields several long-term advantages:

- Maintainability – Modular MVVM architecture simplifies updates and debugging.
- Reproducibility – Deterministic serialization to .WDC and .json ensures identical results across environments.
- Performance – 64-bit execution removes memory limits and accelerates file operations.
- Interoperability – JSON export enables forward compatibility if needed to integrate with cloud infrastructure
- Accessibility – Built-in UI Automation provides a foundation for inclusive design.

- Extensibility – The modular codebase supports future enhancements such as GIS visualization, batch processing, or cloud deployment.

3.6 Validation, Deployment, and Reproducibility

Verification of the modern WinDAM C application required a structured process to ensure that modernization did not alter either the USDA-ARS breach-erosion algorithms or their numerical performance.

This section documents the methods used to perform validation testing, software deployment, and data traceability so that every comparison between the legacy and modern systems can be reproduced exactly.

3.6.1 Validation Procedure

The validation procedure consisted of two stages designed to confirm that (1) the modern interface reproduced legacy input files exactly, and (2) the new 64-bit simulator produced results consistent with the original 32-bit implementation.

Stage 1 – WDC File Comparison

./numdiff -E -F 2 -r 0.00 -b KarlTestTen32/01-PSwDD.WDC ValidateWDC/01-PSwDD.WDC

The first stage compared the .WDC files generated by the modern WinUI 3 interface with those created by the legacy VB6 version. Each pair was evaluated using numdiff with zero tolerance (-r 0.00), enforcing an exact match of text and numeric values.

This confirmed that:

- All parameter fields appeared in the same order and units.

- Numeric precision, rounding, and formatting were identical.
- Default and optional values were preserved.
- Ten benchmark cases provided representative embankment-dam configurations.
- All ten files matched exactly, verifying that the modern interface reproduced the USDA-ARS input schema with complete fidelity.

The comparisons were executed through the Bash script numdiffTenWDC.bash, and thresults were recorded in numdiffTenWDC.text for documentation.

Stage 2 – Simulation Output Comparison (32-bit Legacy vs Modern 64-bit)

./numdiff -E -F 2 -r 0.05 KarlTestTenNewUI/02-SDHstability.OUT KarlTestTen32/02-SDHstability.OUT

The second stage evaluated numerical consistency between the legacy 32-bit simulator and the modern 64-bit version distributed with the new interface. Because the .WDC inputs were identical, any difference in output reflected only floating-point precision or time-integration effects. A relative tolerance of -r 0.05 was used to accommodate negligible rounding variation while maintaining equivalence in hydraulically significant parameters such as: Peak discharge and hydrograph shape, Breach formation timing, and Erosion depth and final breach geometry. The Bash script numdiffTen.bash performed these comparisons across all Karl Test cases, producing concise textual summaries of absolute and relative differences for each dataset.

All comparison activities were performed within a single working directory to maintain transparency and reproducibility:

Testing/

├── KarlTestTen32/ # Legacy 32-bit simulation outputs

├── KarlTestTenNewUI/ # Modern 64-bit simulation outputs

```
├── ValidateWDC/          # WDC-file comparison inputs
├── numdiffTenWDC.bash    # Script for WDC-file comparisons
├── numdiffTenWDC.text    # Output log of WDC comparisons
├──numdiffTen.bash        # Script for simulation-output comparisons
└── numdiffTen.text       # Output log of simulation-output comparisons
```

Keeping the scripts, reference files, and comparison results in one directory eliminates path dependencies and simplifies verification by other researchers. Running either Bash script reproduces the validation exactly, ensuring a clear and auditable record of all tests.

3.6.2 Deployment

The legacy WinDAM C installer depended on 32-bit Visual Basic 6 runtimes and COM registrations, which limited its compatibility with modern systems. Deployment was therefore redesigned using the WiX Toolset, which creates a deterministic, dependency-free installation package. The installer establishes a fixed directory hierarchy: C:\Program Files\USDA\WinDAMv1.1.16

This directory contains:

- WinDAMC.exe – Graphical interface (C#, WinUI 3).
- WinDAMSim.exe – Dam breach-erosion simulator.
- SitesSim.exe – Auxiliary-spillway simulator.

The structure mirrors the legacy release, allowing existing workflows and scripts to operate unchanged. The WiX configuration performs file placement, Start-Menu integration, and clean uninstallation automatically, with no external dependencies or environment-variable edits

required. Each installation yields an identical directory layout, enabling consistent testing and long-term maintainability.

3.6.3 Consistency and Reproducibility

Running the ten Karl Test Series cases on both the 32-bit and 64-bit simulators confirmed that numerical results remained within the defined tolerance (± 0.05 relative).

Observed differences were consistent with expected floating-point precision limits rather than algorithmic errors. Peak discharge, breach timing, and erosion depth agreed across versions, demonstrating that modernization preserved the physical formulations and predictive reliability of the USDA-ARS models.

3.7 Summary

Modernization of WinDAM C produced a reproducible, verifiable workflow that maintains the scientific credibility of the USDA-ARS algorithms while adopting modern software engineering standards.

Key outcomes include:

- Input-file fidelity: The modern interface generates .WDC files identical to legacy VB6 outputs.
- Numerical consistency: 64-bit simulation results match the 32-bit version within machine-precision tolerances.
- Deterministic deployment: The WiX Toolset installer guarantees consistent installation paths and file structures.

- Data transparency: All scripts, inputs, and results are stored together for complete traceability.

These results confirm that modernization enhanced stability and maintainability without altering validated physical formulations. WinDAM C now functions as a sustainable research-grade platform that supports reproducible modeling, long-term maintenance, and future extensions such as GIS integration and accessibility improvements. The next chapter presents the detailed results of the 32-bit and 64-bit comparisons, including specific cases where small numerical deviations were observed.

Chapter 4 - Results and Discussion

Modernization of the WinDAM C system involved verifying that improvements in architecture and interface design did not alter the fundamental hydraulic and erosion algorithms originally developed by the USDA-ARS. The comparative evaluation described in this chapter was carried out in two parts: first, by confirming that the modern interface produces identical project input files to the legacy VB6 system, and second, by assessing whether the modern 64-bit simulator reproduces the results of the legacy 32-bit simulator within expected numerical precision. A supplemental visual comparison was also performed to document how the new WinUI 3 environment responds to accessibility settings such as large-text scaling and high-contrast color schemes. The objective throughout was not only to demonstrate reproducibility but also to establish the foundation for maintainable and accessible future development.

4.1 WDC File Comparison

The first stage of validation focused on the serialized project files used by the simulators—the .WDC files that encode all model inputs, including dam geometry, spillway parameters, and material properties. Because the legacy VB6 program wrote these files using fixed-width text formatting, small differences in spacing and padding were expected once the project data were exported through the modern interface, which uses standard text serialization.

Initial file comparisons confirmed that the numerical values matched but that the comparison tool reported multiple “shorter than expected” messages caused purely by differing numbers of blank spaces at the ends of lines. To isolate meaningful content, the comparisons were rerun using the `-b` flag in **numdiff**, which instructs the program to ignore whitespace differences. The zero-tolerance option (`-r 0.00`) was retained to enforce exact equality of all

numeric fields. Each test therefore compared only the substantive data—numeric entries and text tokens—while disregarding insignificant formatting variations.

Once whitespace was excluded, all ten Karl Test Series `.WDC` files produced by the modern interface matched their legacy counterparts exactly. The comparisons showed no discrepancies in any parameter field, indicating that the WinUI 3 version reproduces the legacy input schema in full: the same order of parameters, units, rounding precision, and default values. The outcome confirmed that the new C# serialization layer preserves the established USDA-ARS data structure without alteration. This step provided confidence that subsequent simulator comparisons would be valid, since both 32-bit and 64-bit executables would receive identical inputs.

```
@ Line 20 in file "KarlTestTen32/01-PSwDD.WDC" is shorter than expected!
----------------
##21            <==
##21      #>6   ==>
@ Line 21 in file "KarlTestTen32/01-PSwDD.WDC" is shorter than expected!
----------------
##22            <==
##22      #>6   ==>
@ Line 22 in file "KarlTestTen32/01-PSwDD.WDC" is shorter than expected!
----------------
##23            <==
##23      #>6   ==>
@ Line 23 in file "KarlTestTen32/01-PSwDD.WDC" is shorter than expected!
----------------
##24            <==
##24      #>6   ==>
@ Line 24 in file "KarlTestTen32/01-PSwDD.WDC" is shorter than expected!
----------------
##25            <==
##25      #>6   ==>
@ Line 25 in file "KarlTestTen32/01-PSwDD.WDC" is shorter than expected!
----------------
##26            <==
##26      #>6   ==>
@ Line 26 in file "KarlTestTen32/01-PSwDD.WDC" is shorter than expected!
----------------
##27            <==
##27      #>6   ==>
@ Line 27 in file "KarlTestTen32/01-PSwDD.WDC" is shorter than expected!
----------------
##28            <==
##28      #>6   ==>
@ Line 28 in file "KarlTestTen32/01-PSwDD.WDC" is shorter than expected!
----------------
##29            <==
##29      #>6   ==>
@ Line 29 in file "KarlTestTen32/01-PSwDD.WDC" is shorter than expected!
----------------
##30            <==
##30      #>6   ==>
@ Line 30 in file "KarlTestTen32/01-PSwDD.WDC" is shorter than expected!
```

Figure 7. Comparing .WDC files without ignoring white spaces

```
Testing
+++  File "KarlTestTen32/01-PSwDD.WDC" differs from file "ValidateWDC/01-PSwDD.WDC"
+++  File "KarlTestTen32/02-SDHstability.WDC" differs from file "ValidateWDC/02-SDHstability.WDC"
+++  File "KarlTestTen32/03-Overtop.WDC" differs from file "ValidateWDC/03-Overtop.WDC"
+++  File "KarlTestTen32/04-HansonOvertop.WDC" differs from file "ValidateWDC/04-HansonOvertop.WDC"
+++  File "KarlTestTen32/05-HROvertopBreach.WDC" differs from file "ValidateWDC/05-HROvertopBreach.WDC"
+++  File "KarlTestTen32/06-THOvertopBreach.WDC" differs from file "ValidateWDC/06-THOvertopBreach.WDC"
+++  File "KarlTestTen32/07-FBIntegrity.WDC" differs from file "ValidateWDC/07-FBIntegrity.WDC"
+++  File "KarlTestTen32/08-FBIntegritywBarrier.WDC" differs from file "ValidateWDC/08-FBIntegritywBarrier.WDC"
+++  File "KarlTestTen32/09-FBIntegrityNoBarrierSand.WDC" differs from file "ValidateWDC/09-FBIntegrityNoBarrierSand.WDC"
+++  File "KarlTestTen32/10-FBIntegrity2AuxSpwy.WDC" differs from file "ValidateWDC/10-FBIntegrity2AuxSpwy.WDC"
```

Figure 8. Comparing .WDC files after ignoring white spaces

4.2 Simulation Output Comparison

After verifying input equivalence, each of the ten Karl Test projects was executed using both the legacy 32-bit simulator and the modern 64-bit simulator that operates under the new interface. The resulting output files (`.OUT`) were then compared line-by-line with **numdiff**, applying a relative tolerance of 0.05 and an absolute tolerance of 0.01. These tolerances are appropriate for distinguishing physically meaningful differences from the minor deviations expected when migrating from single- to double-precision arithmetic.

Across all cases, the overall agreement between the two simulators was excellent. Seven of the ten benchmark projects produced identical output files, with every numeric field matching to the last decimal digit. The remaining three showed small, localized differences confined to sections of rapid hydraulic transition or very low discharge where floating-point precision has the greatest influence. None of the differences affected the shape of the breach hydrograph or the computed peak discharge to a degree that would alter engineering interpretation.

In the file ***02-SDHstability.OUT***, a single difference was detected at line 853, where the 32-bit result listed 15.02 and the 64-bit result listed 14.28, an absolute error of 0.74 and a relative difference of 0.052 percent. When the same case was rerun with a smaller simulation time step, the two results converged—14.60 for the 64-bit run and 14.56 for the 32-bit run—reducing the relative difference to 0.0027 percent. (Refer to Appendix A) behavior demonstrates that the apparent discrepancy stems from time-integration sensitivity rather than from a change in the algorithm itself. Both solvers approach the same limit as temporal resolution increases.

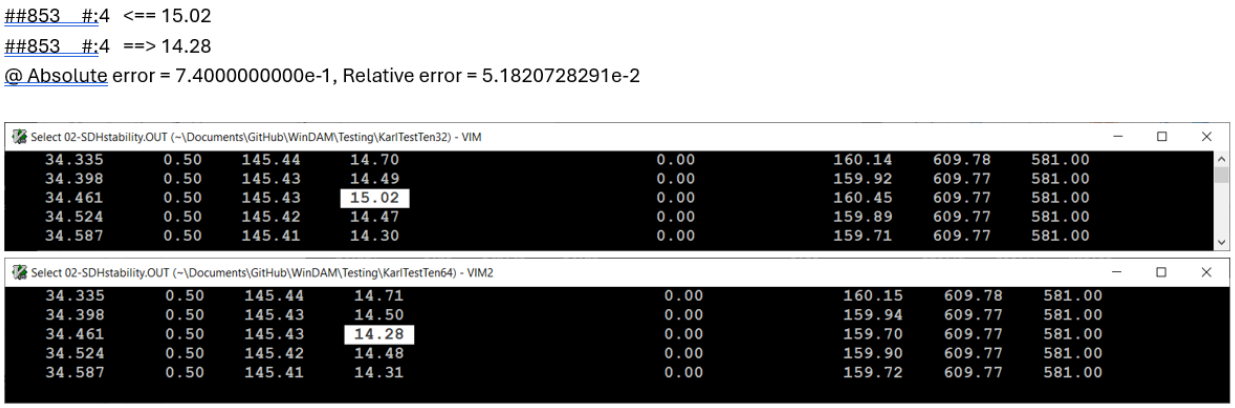


Figure 9. Comparing .OUT files of 02-SDHstability

In ***06-THOvertopBreach.OUT***, differences appeared beginning near line 2533 and again around line 3450, corresponding to the period when flow transitions from the principal spillway to overtopping. Individual lines differed by a few hundredths (for example, 0.25 versus 0.23 for the principal spillway outflow). The 64-bit version integrated one additional time step before termination, producing a slightly longer file. Such variations are typical of cumulative rounding behavior in iterative hydraulic solvers and are considered numerically insignificant. When run with smaller time steps, both outputs converged to nearly identical results.

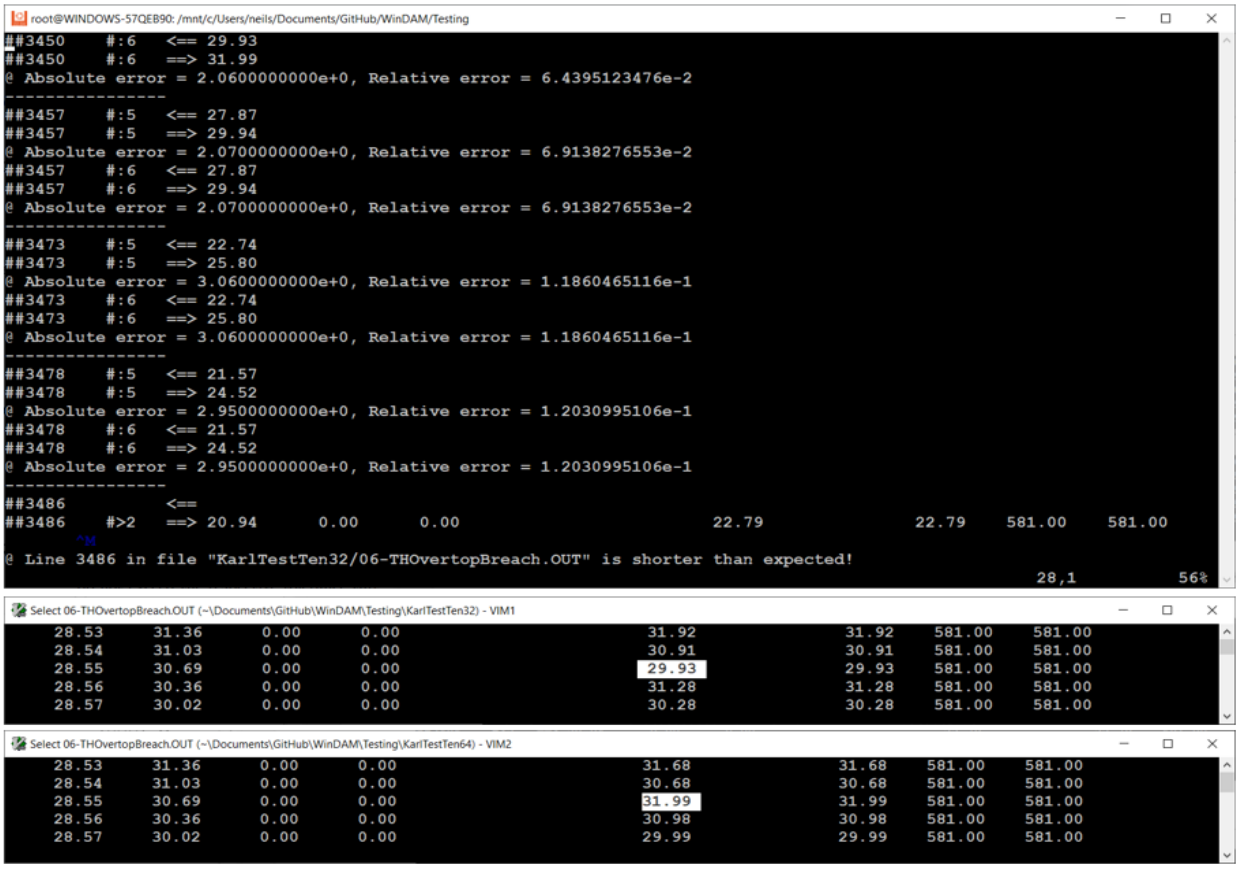


Figure 10. Comparing .OUT files of 06-THOvertopBreach

The final non-identical case, ***10-FBIntegrity2AuxSpwy.OUT***, exhibited two small deviations at line 680 in fields 4 and 5: 13.35 in the legacy output versus 12.65 in the modern version, corresponding to an absolute difference of 0.70 and a relative error of 5.5 percent. These fields represent auxiliary-spillway discharges under very low-flow conditions (less than 32 cubic feet per second). At such magnitudes, single-precision rounding dominates, and small changes in arithmetic precision can produce percent-level differences without affecting physical interpretation. The discrepancy therefore reflects numerical resolution limits rather than model divergence.

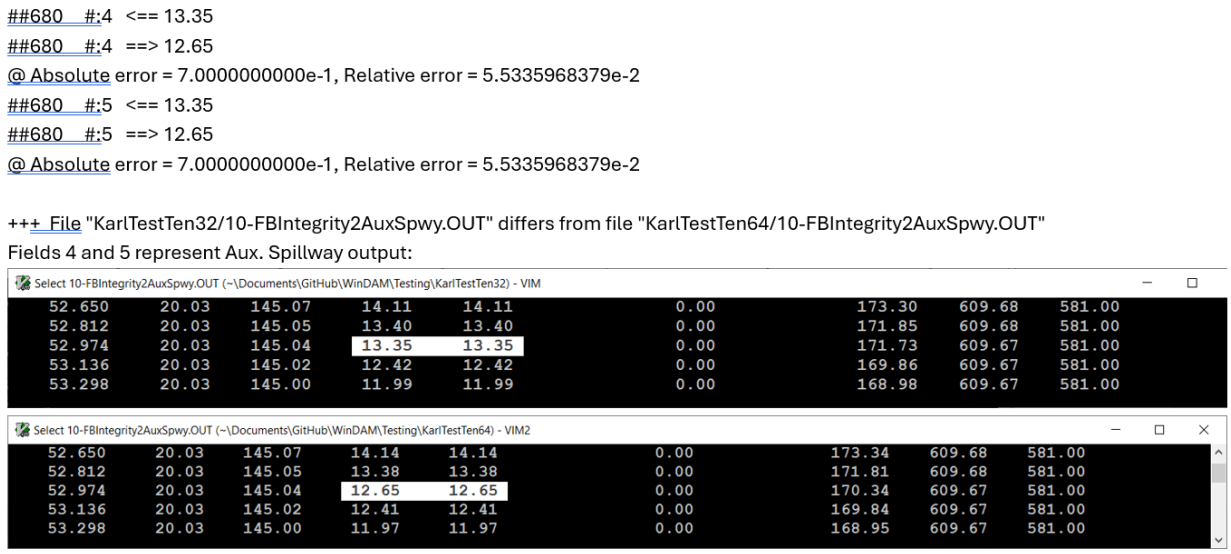


Figure 11. Comparing .OUT files of 10-FBIntegrity2AuxSpwy

Taken together, the results confirm that modernization preserved the physical and numerical integrity of the USDA-ARS breach and erosion formulations. Differences are confined to the level expected from transitioning between 32-bit and 64-bit floating-point computations and from the slightly altered rounding behavior of modern compilers. When integration resolution is refined, both solvers converge toward the same solution, demonstrating algorithmic equivalence.

4.3 Interpretation of Numerical Differences

The observed deviations illustrate fundamental properties of floating-point arithmetic rather than flaws in the modernization process. The 32-bit executable carries approximately seven significant digits of precision, whereas the 64-bit version carries about fifteen. Over thousands of iterative time-integration steps, the cumulative effect of rounding differences can shift termination conditions by one iteration or alter the least significant digits of intermediate variables. Because WinDAM C's algorithms are deterministic, even minute differences in

intermediate state variables can propagate through successive steps, leading to minor changes in printed output lines. However, such differences have no practical impact on predicted breach timing or discharge.

The sensitivity tests with halved time steps confirm this interpretation: both the legacy and modern simulators approach identical results as time resolution increases. These findings indicate that modernization affected only numerical precision, not the physical models or empirical relationships embedded within the code. From an engineering perspective, the two systems are therefore equivalent.

4.4 Summary of Numerical Validation

The comprehensive comparison between the legacy and modern WinDAM C systems demonstrates complete fidelity in input serialization and near-perfect agreement in numerical results. The modernization preserved all scientific formulations while improving computational stability and scalability through the adoption of 64-bit arithmetic. Any small discrepancies observed are well within the limits of floating-point round-off and vanish when simulation resolution is refined. The modernization can thus be regarded as numerically validated: the 2025 WinUI 3 implementation produces results indistinguishable, for all practical purposes, from those of the original USDA-ARS code base.

4.5 Preliminary Accessibility Evaluation

Although accessibility was not a primary focus of this phase of modernization, the transition from VB6 to WinUI 3 provided an opportunity to observe how the new framework inherently supports system-level accessibility features. The evaluation centered on two

fundamental aspects of visual usability: text-size scaling and high-contrast theme adaptation. The intention was not to achieve full compliance with Section 508 or WCAG 2.1 standards but to understand the baseline capabilities of the new environment and how it positions the application for future enhancement.

When Windows text size was increased to 200 percent, the legacy VB6 interface ignored the system setting entirely. All text retained its original pixel size, resulting in labels, buttons, and data fields that appeared disproportionately small within the enlarged window. The interface therefore remained unchanged which is not the expected behavior and it would have been difficult to interpret the text for a user who wishes to use enlarged text system wide. This limitation is inherent to VB6's fixed-font rendering and cannot be corrected without rewriting the interface.

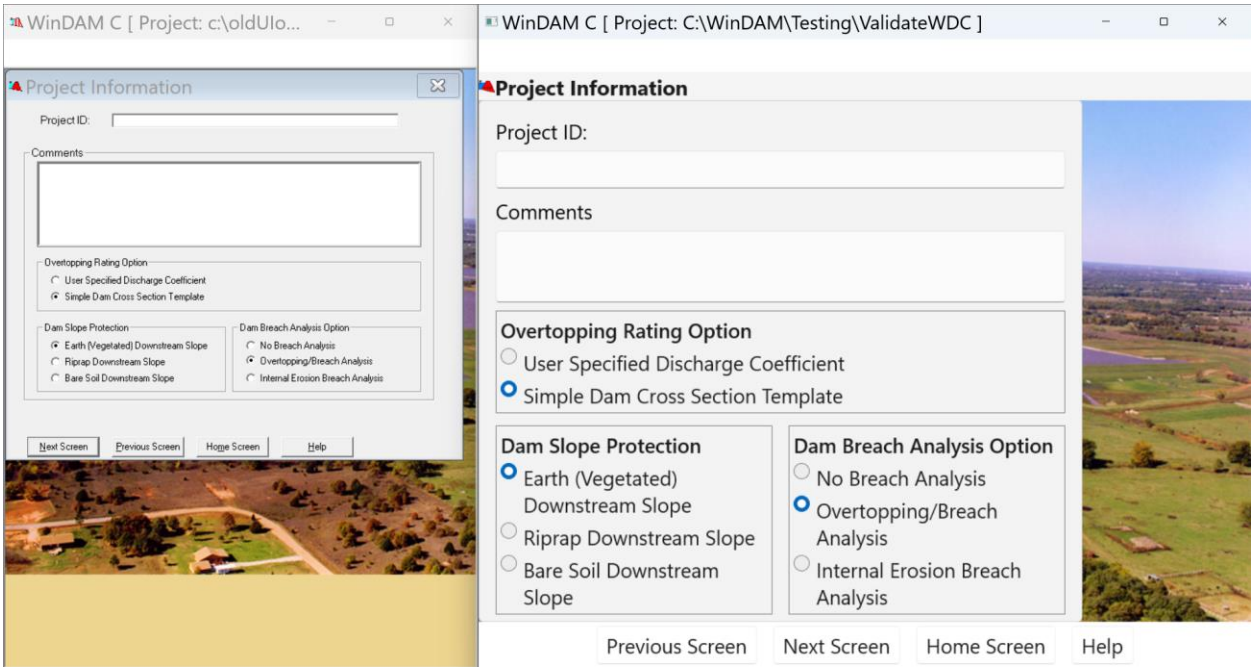


Figure 12. Comparing the Font Size of the Input Screens, when System Text Size is set to 200%.

In contrast, the WinUI 3 implementation responded dynamically. At the same 200 percent scaling level, all labels, text fields, and table rows expanded proportionally, and control spacing adjusted automatically. The behavior required no special code within WinDAM C; it is a standard feature of the WinUI typography and layout system. This alone represents a substantial improvement in readability for users working on high-DPI or magnified displays.

Similar differences were observed when the **Windows High-Contrast – Aquatic Dark** theme was enabled. In the legacy interface, only the window frame and text adopted the high-contrast palette dynamically; interior tables remained white. As a result, the tables displayed **white text on white backgrounds**, rendering data virtually invisible. By comparison, the modern interface automatically adapted to the high-contrast palette through its use of dynamic system color resources. All text and borders switched to theme-appropriate brushes, producing **white text on black backgrounds** with accent borders that clearly delineated focus and selection. Again, these improvements were achieved without custom accessibility code—they are inherent to the WinUI 3 framework.

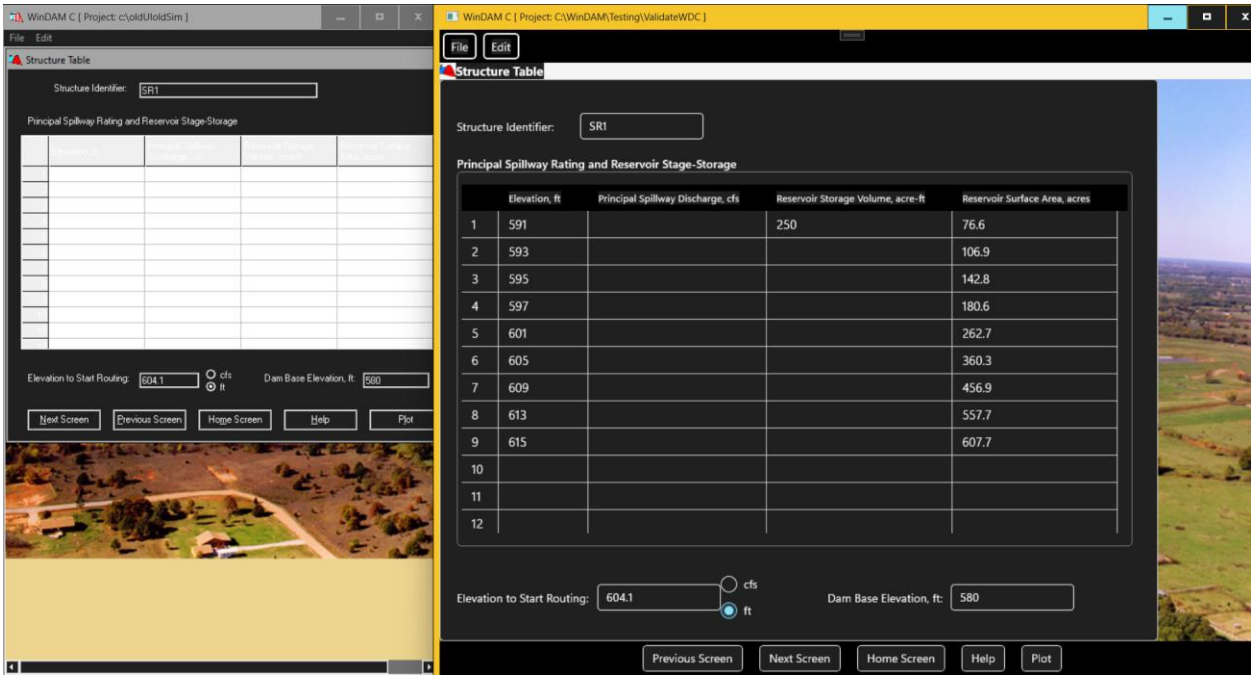


Figure 13. Comparing if the UI updates dynamically with high contrast changes

While these results do not make WinDAM C fully accessible, they show that modernization has eliminated several fundamental barriers present in the legacy design. The new interface respects system text-size settings, supports theme-driven color inversion, and maintains layout stability under scaling. These behaviors form the groundwork for future accessibility development. Features such as narrator support, keyboard focus management, and enhanced color-contrast checking can now be added incrementally using the accessibility APIs built into WinUI 3. Implementing comparable capabilities in the VB6 environment would have required extensive rewriting and external dependencies.

From an engineering perspective, the accessibility evaluation demonstrates that the modernization has taken the first practical steps toward an inclusive interface. The application is not yet compliant with formal standards, but its architecture now aligns with modern accessibility infrastructure. The ability to inherit system color themes, rescale typography

dynamically, and respond to user preferences represents a foundational improvement that will facilitate continued enhancement in subsequent development phases.

4.6 Summary

The results presented in this chapter confirm that the modernization of WinDAM C successfully preserved the scientific accuracy of the USDA-ARS breach and erosion algorithms while introducing an architecture that supports long-term maintainability and potential accessibility expansion. The `.WDC` file comparisons verified identical input serialization once formatting differences were excluded. The numerical analyses demonstrated that the modern 64-bit simulator reproduces the legacy 32-bit results within expected floating-point precision limits, and time-step refinement confirmed convergence toward identical solutions. Finally, the preliminary accessibility evaluation showed that, although WinDAM C is not yet fully accessible, migration to the WinUI 3 framework provides automatic text-scaling and high-contrast capabilities that the legacy system entirely lacked. These inherent features establish a strong technical foundation for future compliance work.

In summary, modernization achieved its principal objectives of reproducibility and stability while positioning the application for progressive enhancement. The next chapter presents the conclusions drawn from this research and outlines specific recommendations for extending accessibility, usability, and integration capabilities in future versions of WinDAM C.

Chapter 5 - Conclusions and Future Work

5.1 Overview

This chapter interprets the broader meaning of the modernization effort undertaken for the Windows Dam Analysis Model for Cohesive Embankments (WinDAM C). The previous chapters documented the engineering context, the modernization process, and the quantitative validation of numerical consistency. Here the emphasis shifts from individual results to their overall significance: what the modernization demonstrates about sustaining scientific software, what constraints remain, and how the system can evolve further. The discussion therefore focuses on implications, lessons learned, and recommended directions for continued development.

5.2 Broader Significance of Modernization

Re-engineering WinDAM C demonstrated that modern software architecture can extend the useful life of specialized scientific models without compromising their verified behavior. The migration from Visual Basic 6 to C# with WinUI 3 transformed the program from a platform-dependent desktop utility into a maintainable, version-controlled research application. The modernization also confirmed that scientific reproducibility can coexist with contemporary software engineering practices such as modular design, 64-bit computation, and deterministic installation.

A key outcome of this work is the proof that modernization can be approached incrementally: the physical formulations, long trusted within USDA-ARS research, were left untouched, while the interface, data handling, and deployment infrastructure were rebuilt. This

separation of scientific and software concerns provides a sustainable pattern for other legacy analytical tools whose original codes remain scientifically valid but technologically outdated.

5.3 Lessons Learned

Modernizing legacy scientific software required balancing historical consistency with new technical conventions. Three main lessons emerged.

First, validation must be designed in parallel with modernization. Establishing reproducibility criteria at the beginning—through structured file comparisons and tolerance thresholds—prevented ambiguity about whether observed differences were numerical or procedural. This approach transformed validation from a one-time quality check into a reusable verification process that now accompanies each build of WinDAM C.

Second, framework selection influences long-term adaptability as much as programming language choice. Many improvements observed after modernization, including dynamic text scaling and theme awareness, were not custom features but intrinsic behaviors of the WinUI 3 framework. Selecting a framework with built-in accessibility and automation support reduces future development effort and ensures that the software will remain compatible with evolving operating-system standards.

Third, incremental modernization preserves institutional knowledge. Because the project retained the USDA-ARS algorithms unchanged, existing calibration data, documentation, and training materials remain valid. This continuity avoids the typical disruption associated with replacing legacy tools wholesale and shows that modernization can be evolutionary rather than revolutionary.

5.4 Current Limitations

Despite these successes, several limitations constrain the present release. The modernization focused on interface redesign and reproducibility; it did not expand the scientific scope of the model. Simulation workflows remain file-based, and no graphical visualization or parameter-sensitivity tools are yet included. Although the new interface inherits basic system-level behaviors such as text scaling and color inversion, comprehensive accessibility support—including screen-reader narration, keyboard navigation, and formal compliance testing—has not been implemented.

Performance testing was limited to the Karl Test Series datasets. While these represent realistic scenarios, additional validation against extreme cases or very fine temporal resolutions would strengthen confidence in large-scale applications. Furthermore, the reproducibility scripts rely on local execution; integration with automated build or continuous-integration systems is a logical next step but lies outside the current scope.

5.5 Future Development Directions

The architecture produced through this modernization creates opportunities for several lines of future work.

Enhanced Accessibility: The most immediate area for improvement is formal accessibility implementation. Because WinUI 3 already exposes Microsoft UI Automation interfaces, future versions can add narrator announcements, descriptive automation peers for data tables, and full keyboard-navigation pathways without altering the application’s structure. Establishing a small accessibility-testing suite using tools such as Accessibility Insights would enable systematic evaluation against WCAG 2.1 and Section 508 criteria.

Interactive Visualization and Analysis: The current workflow remains text-file oriented. The next logical step is to integrate lightweight plotting of breach hydrographs, reservoir levels, and erosion depth in real time. WinUI 3 and .NET 6 provide sufficient graphics performance to support this without external dependencies. Visualization would help users interpret results more intuitively and detect anomalies that text files may obscure.

Automated Regression and Continuous Validation: The existing comparison scripts can be incorporated into a continuous-integration pipeline so that every software update automatically re-executes the standard Karl Test Series. Any numerical deviation beyond defined tolerances would be flagged immediately, ensuring that scientific fidelity is maintained across releases.

Interoperability and Data Exchange: Because the modern architecture already supports structured data formats such as JSON or XML, it can be extended to import or export data compatible with other USDA and NRCS hydrologic tools (for example, TR-20 or HEC-RAS). Standardized data exchange would reduce manual preprocessing and broaden the applicability of WinDAM C.

User-Experience Refinement: Future development could focus on refining window layouts, keyboard shortcuts, and error reporting to make the program more approachable for new users. These changes, while minor technically, will further align the application with modern usability expectations in research and agency environments.

5.6 Implications for Scientific Software Sustainability

This work illustrates a generalizable strategy for sustaining computational tools that underpin long-term research programs. Many agencies rely on models created decades ago in languages or frameworks now considered obsolete. Complete rewrites risk altering verified algorithms, while ignoring modernization leads to gradual obsolescence. The WinDAM C experience demonstrates that a middle path is possible: maintain the proven scientific core, replace the surrounding infrastructure, and document equivalence quantitatively. The result is software that continues to produce trusted results but can evolve with hardware, operating systems, and accessibility expectations.

By documenting both the process and the validation methodology, this project provides a template for future modernization efforts within USDA-ARS or similar organizations. The emphasis on reproducibility, transparency, and incremental enhancement ensures that modernization strengthens scientific credibility rather than disrupting it.

5.7 Concluding Remarks

Modernization of WinDAM C achieved its central goal: preserving the validated USDA-ARS breach and erosion algorithms while rebuilding the surrounding software in a maintainable, extensible, and forward-compatible environment. The transition to C# and WinUI 3 resolved dependency and deployment issues, introduced deterministic installation, and enabled accurate comparison between 32-bit and 64-bit simulations. The program now operates on a foundation that can support both scientific expansion and progressive accessibility improvement.

The work presented here should be viewed as the beginning of a long-term modernization process rather than its completion. The current version provides a stable and verifiable platform upon which future teams can build additional functionality, visualization, and accessibility. By

combining rigorous validation with thoughtful architectural renewal, this project demonstrates how legacy analytical models can continue to serve the research community effectively in modern computing environments.

Bibliography

[1] Wu, W. (2013). Simplified Physically Based Model of Earthen Embankment Breaching. Journal of Hydraulic Engineering, 139(8), 837–851. https://doi.org/10.1061/(ASCE)HY.1943-7900.0000741

[2] Wu, W. (2016). Introduction to DLBreach: A Simplified Physically-Based Dam/Levee Breach Model. Technical report, Department of Civil and Environmental Engineering, Clarkson University.

[3] Hunt, S. L., Temple, D. M., Neilsen, M. L., Ali, A. A., & Tejral, R. D. (2021). WinDAM C: Analysis Tool for Predicting Breach Erosion Processes of Embankment Dams Due to Overtopping or Internal Erosion. Applied Engineering in Agriculture, 37(3), 523–534. https://doi.org/10.13031/aea.14334

[4] ICOLD. (n.d.). World Register of Dams. International Commission on Large Dams. https://www.icold-cigb.org/GB/world_register/general_synthesis.asp (accessed Oct 2025)

[5] USACE. (n.d.). National Inventory of Dams. U.S. Army Corps of Engineers. https://nid.sec.usace.army.mil/#/ (accessed Oct 2025)

[6] Wahl, T. L., et al. (1998). Prediction of embankment dam breach parameters: A literature review and needs assessment (DSO-98-004). U.S. Bureau of Reclamation, Denver, CO.

[7] Thornton, C. I., Pierce, M. W., & Abt, S. R. (2011). Enhanced Predictions for Peak Outflow from Breached Embankment Dams. Journal of Hydrologic Engineering, 16(1), 81–88. https://doi.org/10.1061/(ASCE)HE.1943-5584.0000288

[8] Fread, D. L. (1984). DAMBRK: The NWS Dam-Break Flood Forecasting Model (Report HRL-256). National Weather Service, Silver Spring, MD.

[9] Fread, D. L. (1988). BREACH: An Erosion Model for Earthen Dam Failures (Revised 1991). National Weather Service, Silver Spring, MD.

[10] Hassan, M. A. A. M., Morris, M. W., Hanson, G. J., & Lakhal, K. (2004). Breach formation: Laboratory and numerical modelling of breach formation. In Proceedings of ASDSO Annual Conference, Phoenix, AZ.

[11] Wahl, T. L., et al. (2011). Physical hydraulic modeling of canal breaches (Report HL-2011-02). U.S. Bureau of Reclamation, Denver, CO.

[12] Al-Riffai, M. (2014). Experimental Study of Breach Mechanics in Overtopped Non-Cohesive Earthen Embankments (Ph.D. dissertation). University of Ottawa, Ottawa, Canada.

[13] Ashraf, M., Soliman, A. H., El-Ghorab, E., & El-Zawahry, A. (2018). Assessment of Embankment Dams Breaching Using Large-Scale Physical Modeling and Statistical Methods. Water Science, 32(2), 362–379. https://doi.org/10.1016/j.wsj.2018.05.002

[14] Alvarez, T., et al. (2019). Sensitivity Analysis to the Pilot Channel Geometry in Dam Breach by Overtopping. In Proceedings of the 38th IAHR World Congress, Panama City, Panama.

[15] Temple, D. M., et al. (2005). Simplified breach analysis model for homogeneous embankments: Part I, Background and model components. In Proceedings of 25th USSD Annual Conference, Salt Lake City, UT.

[16] Hanson, G. J., Tejral, R. D., Hunt, S. L., & Temple, D. M. (2010). Internal Erosion and Impact of Erosion Resistance. U.S. Society on Dams – Technologies to Enhance Dam Safety and the Environment, 773–784, Westminster, CO.

[17] Vaskinn, K. A., et al. (2004). Physical modeling of breach formation: Large scale field tests. In Proceedings of ASDSO Annual Conference, Phoenix, AZ.

[18] Hanson, G. J., & Cook, K. R. (2004). Apparatus, Test Procedures, and Analytical Methods to Measure Soil Erodibility in Situ. Applied Engineering in Agriculture, 20(4), 455–462. https://doi.org/10.13031/2013.16492

[19] Hanson, G. J., & Cook, K. R. (2004). Determination of material rate parameters for headcut migration of compacted earthen materials. In Proceedings of ASDSO Annual Conference, Phoenix, AZ.

[20] Hanson, G. J., & Hunt, S. L. (2007). Lessons Learned Using Laboratory Jet Method to Measure Soil Erodibility of Compacted Soils. Applied Engineering in Agriculture, 23(3), 305–312. https://doi.org/10.13031/2013.22686

[21] ASCE/EWRI Task Committee on Dam/Levee Breaching. (2011). Earthen Embankment Breaching. Journal of Hydraulic Engineering, 137(12), 1549–1564. https://doi.org/10.1061/(ASCE)HY.1943-7900.0000498

[22] CEATI International. (2017). Evaluation of numerical models for simulating embankment dam erosion and breach processes (Report No. DSO-2017-02). U.S. Bureau of Reclamation, Denver, CO.

[23] Zhong, Q., Wu, W., Chen, S., & Wang, M. (2016). Comparison of Simplified Physically Based Dam Breach Models. Natural Hazards, 84(2), 1385–1418. https://doi.org/10.1007/s11069-016-2492-9

[24] West, M., Morris, M., & Hassan, M. (2018). A Guide to Breach Prediction. Dams and Reservoirs, 28(4), 150–152. https://doi.org/10.1680/jdare.18.00031

[25] Zhong, Q. M., Chen, S. S., Deng, Z., & Mei, S. A. (2019). Prediction of Overtopping-Induced Breach Process of Cohesive Dams. Journal of Geotechnical and Geoenvironmental Engineering, 145(5). https://doi.org/10.1061/(ASCE)GT.1943-5606.0002035

[26] Yochum, S. E., Goertz, L. A., & Jones, P. H. (2008). Case Study of the Big Bay Dam Failure: Accuracy and Comparison of Breach Predictions. Journal of Hydraulic Engineering, 134(9), 1285–1293. https://doi.org/10.1061/(ASCE)0733-9429(2008)134:9(1285)

[27] Adams, B. M., et al. (2021). Dakota: A multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis (Version 6.15 User's Manual) (SAND2020-12495). Sandia National Laboratories, Albuquerque, NM.

[28] Neilsen, M. L., & Cao, C. (2017). Coupled dam safety analysis using BREACH and WinDAM. In Proceedings of the 15th International Conference on Scientific Computing, Las Vegas, NV.

[29] Temple, D. M., et al. (2006). Analysis of spillway erosion rate parameters. U.S. Department of Agriculture, Agricultural Research Service, Stillwater, OK.

[30] Atkinson, A., & Neilsen, M. (2024). Analysis of uncertainty in internal erosion simulations for DLBreach and WinDAM C. GeoHazards, 5(2), 350–363. https://doi.org/10.3390/geohazards5020018

[31] Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. Linux Journal, 2014(239), Article 2.

[32] Assunção, M. D., Da Silva, C. B., & Buyya, R. (2024). Technical Debt and Software Sustainability in Legacy Systems. Journal of Systems and Software, 205, 111842.

[33] Bisbal, J., Lawless, D., Wu, B., & Grimson, J. (1999). Legacy Information Systems: Issues and Directions. IEEE Software, 16(5), 103–111.

[34] Microsoft Corporation. (2008). Support statement for Visual Basic 6.0 on Windows Vista and Windows Server 2008. MSDN. https://msdn.microsoft.com/en-us/vbrun/ms788708.aspx

[35] USDA-NRCS. (2022). National Engineering Handbook, Part 650, Chapter 2: Estimating runoff and peak discharge. U.S. Department of Agriculture, Washington, DC.

[36] Microsoft Corporation. (n.d.). Model-View-ViewModel. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

[37] Santamarina, J. C., Torres-Cruz, L. A., & Bachus, R. C. (2020). Why coal ash and tailings dam disasters occur. Science, 364(6440), 526–528. https://doi.org/10.1126/science.aax1927

[38] USBR. (2014). Climate change and water resources management: A federal perspective. U.S. Bureau of Reclamation, Denver, CO.

[39] IPCC. (2022). Climate change 2022: Impacts, adaptation, and vulnerability. Cambridge University Press.

WinDAM C – UI Screenshots (64-bit)

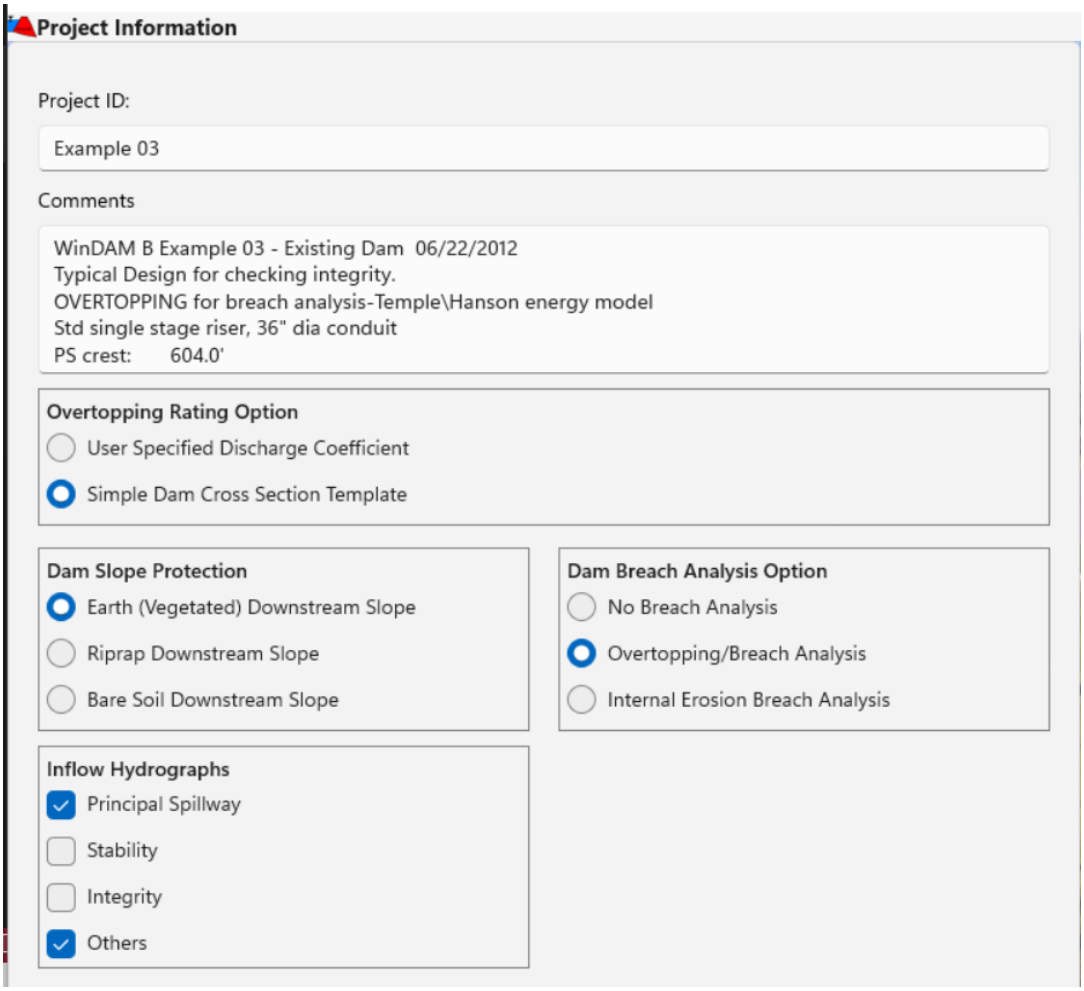


Figure 14. Project Information Screen

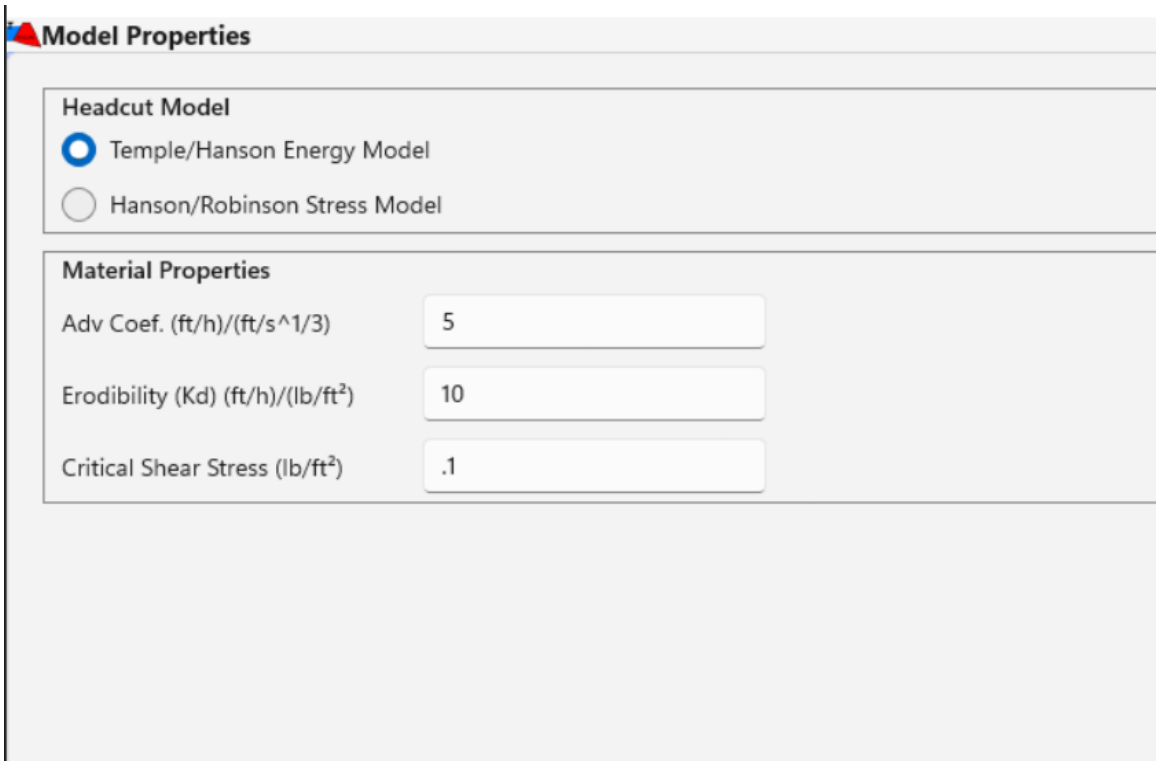


Figure 15. Model Properties Screen

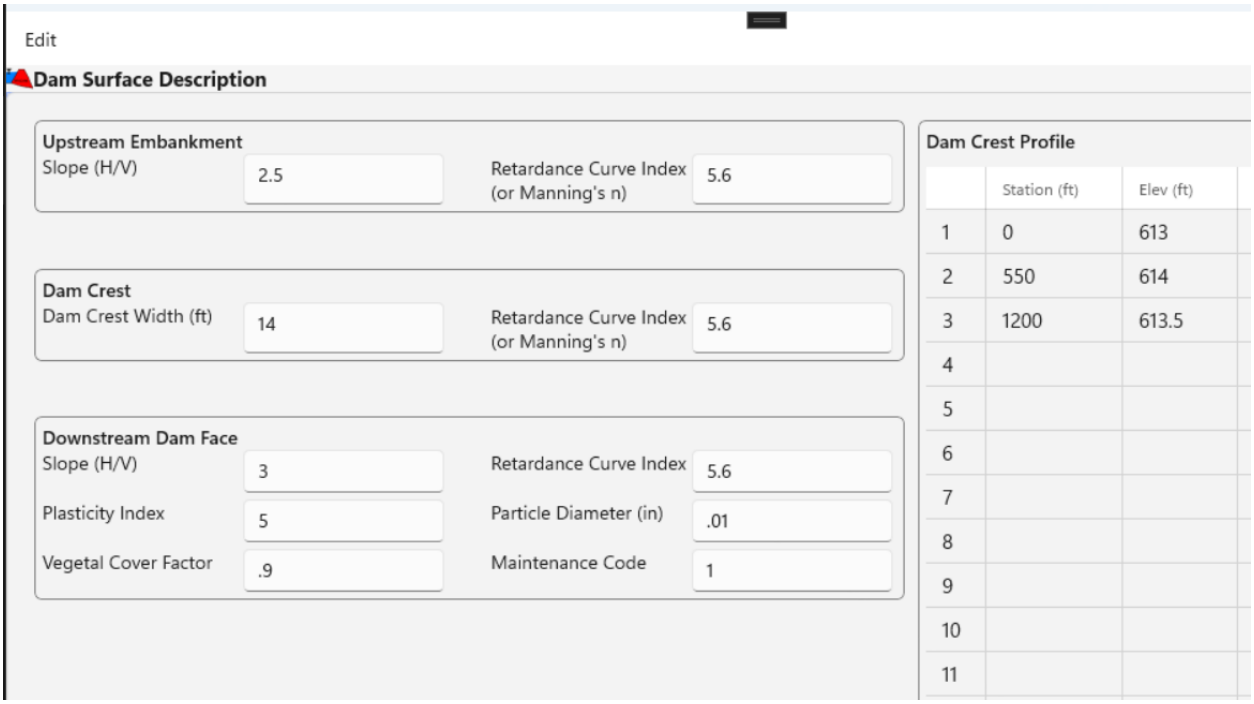


Figure 16. Dam Surface Description Screen

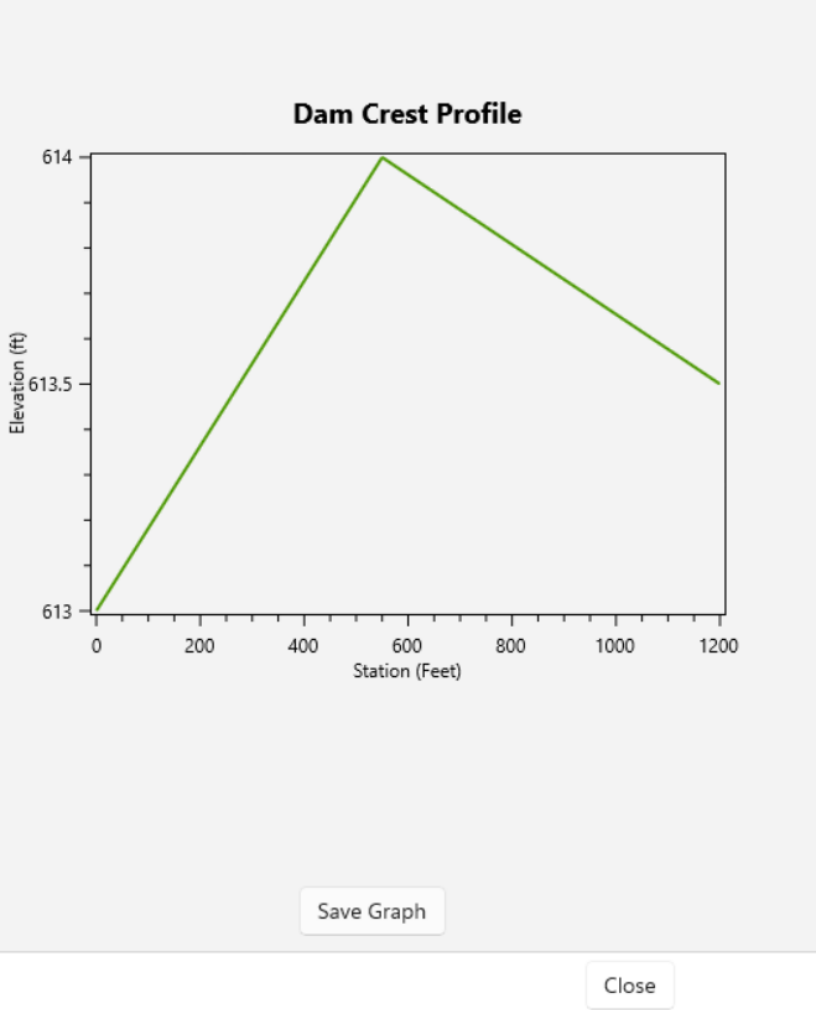


Figure 17. Plot of Dam Crest Profile

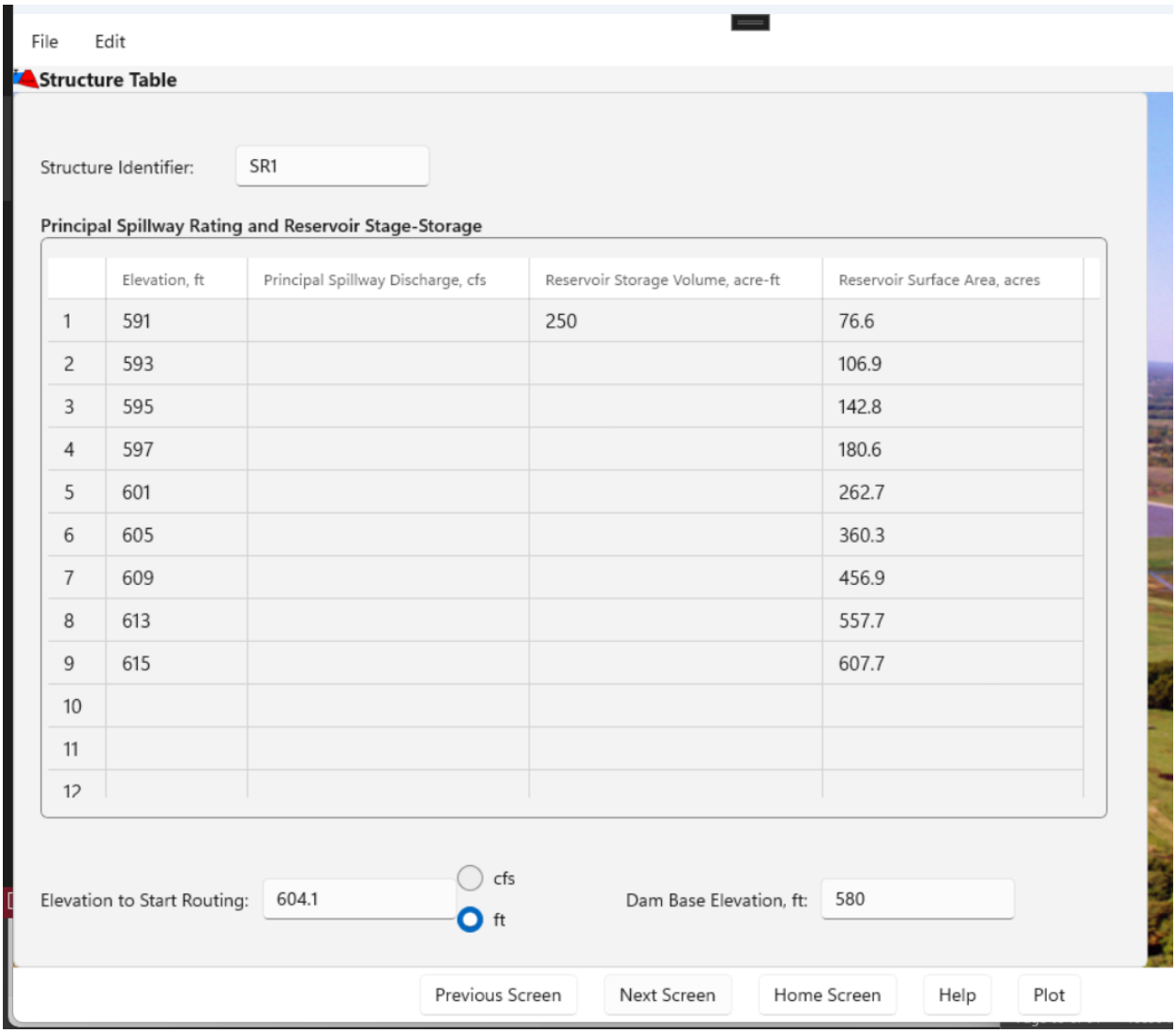


Figure 18. Structure Table Screen

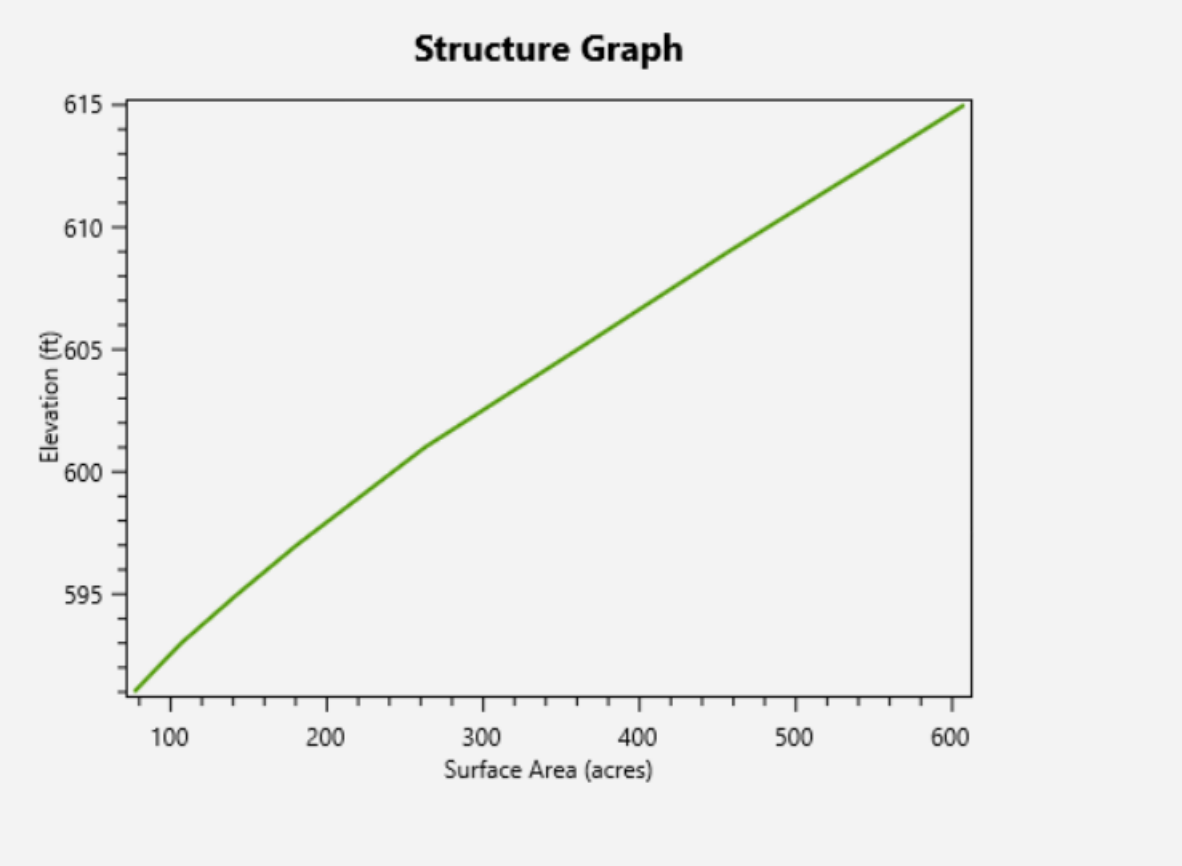


Figure 19. Elevation vs. Surface Area Plot

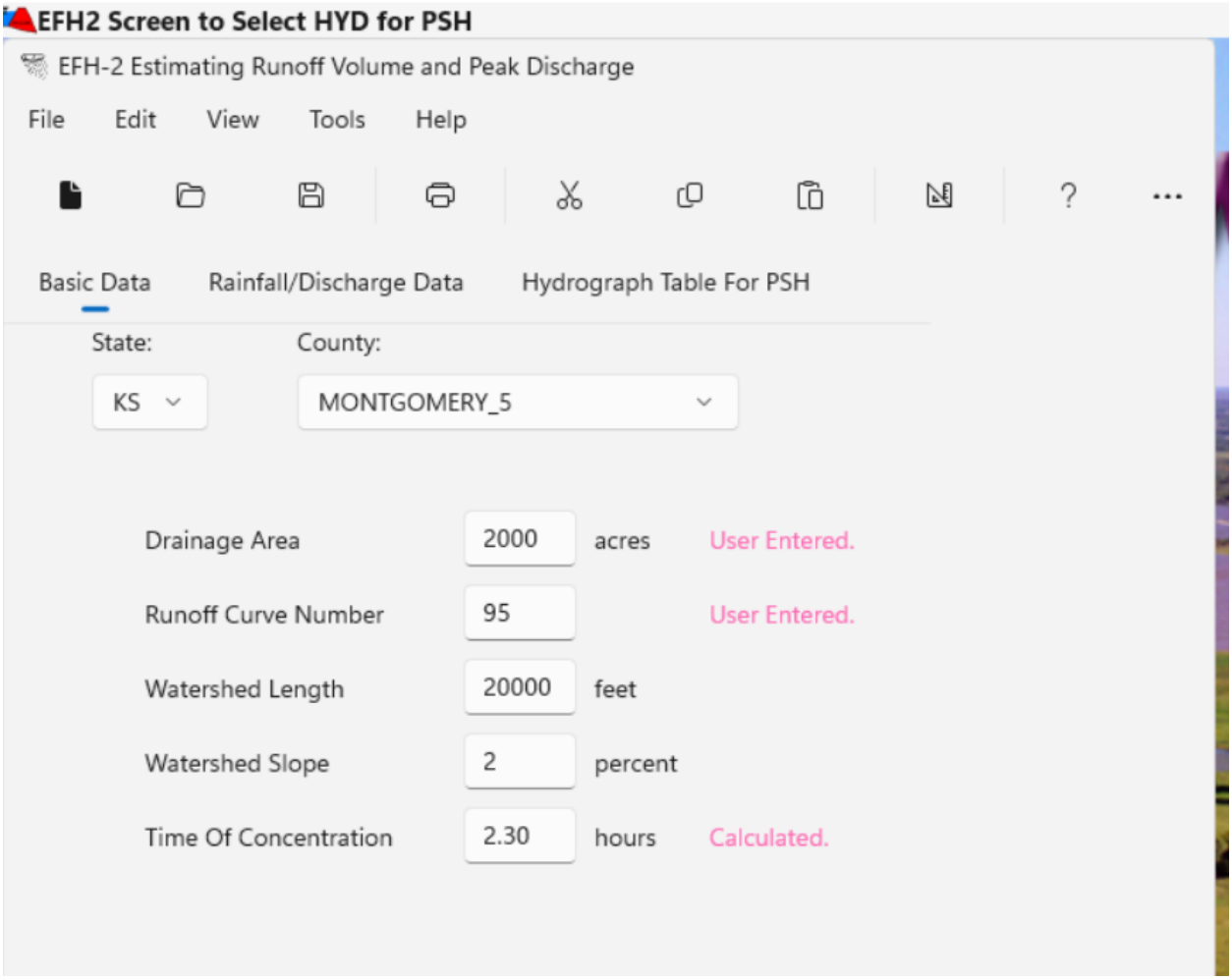


Figure 20. EFH-2 Screen for PSH

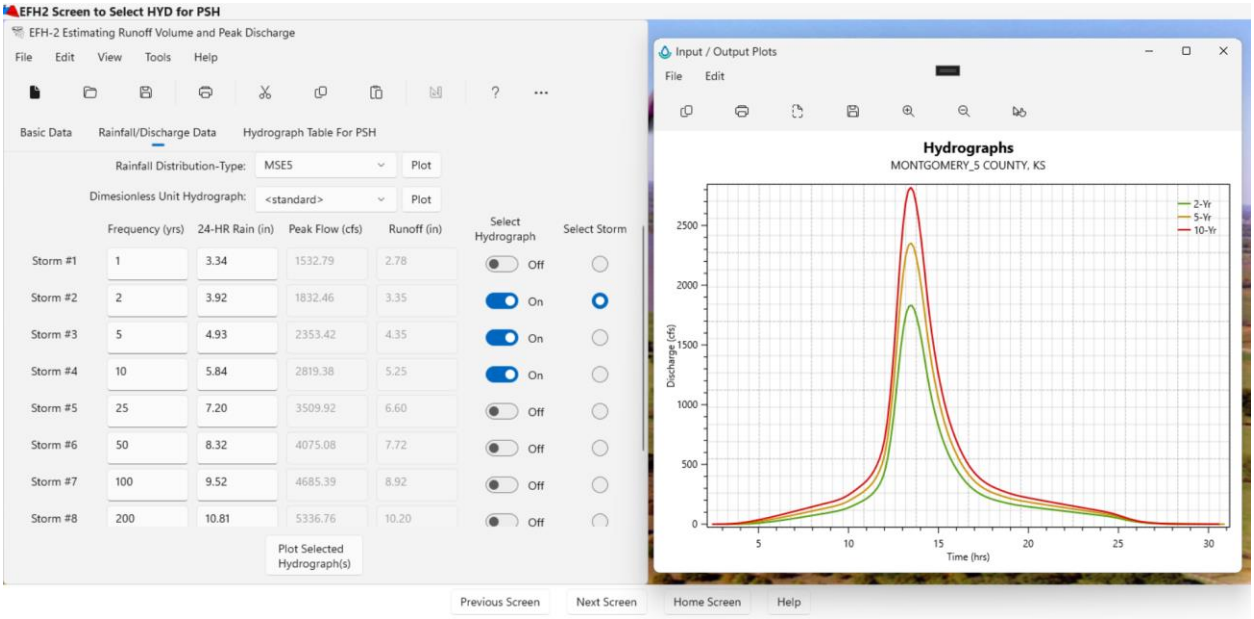


Figure 21. EFH-2 Screen for PSH

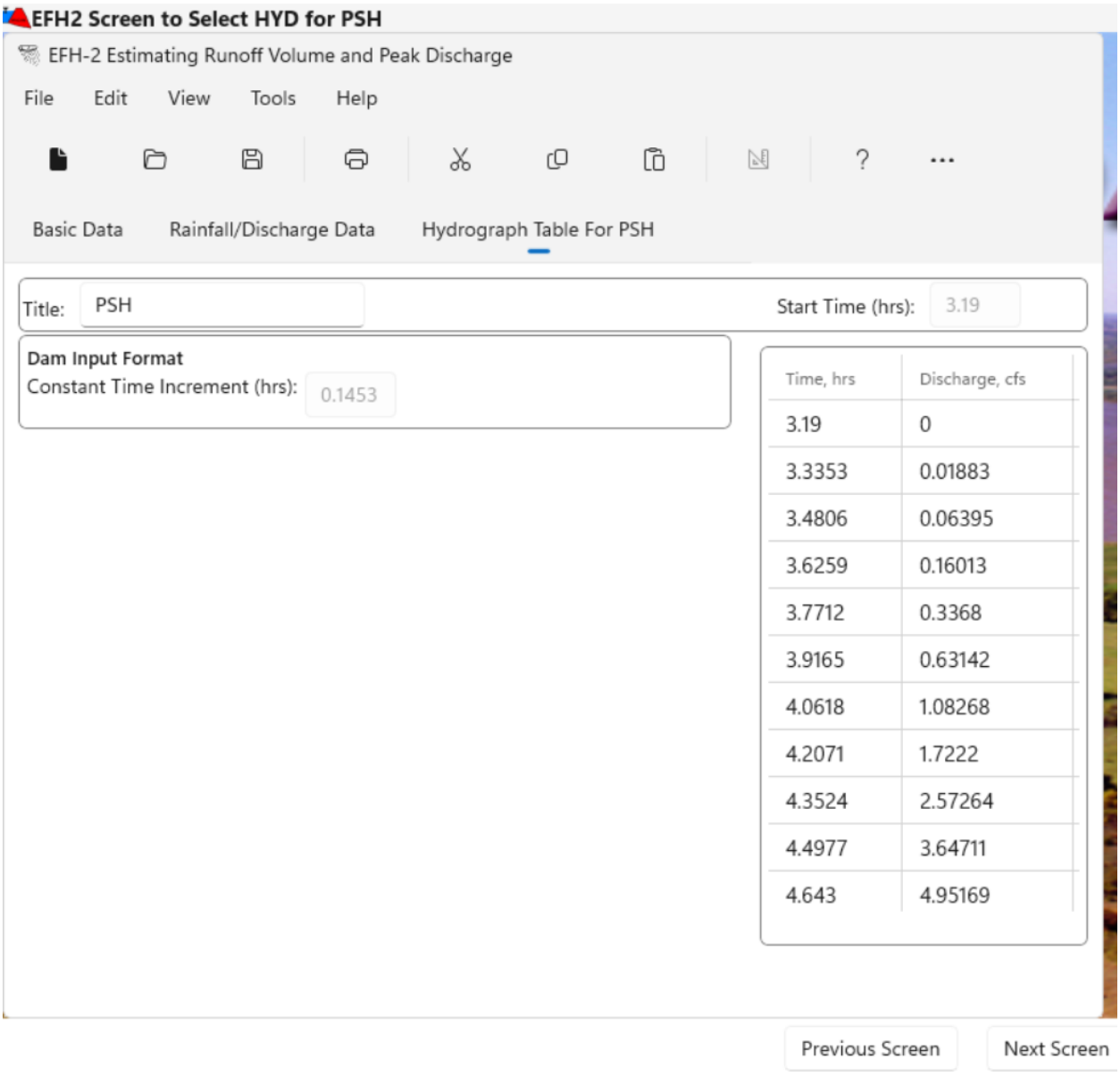


Figure 22. EFH-2 Screen for PSH

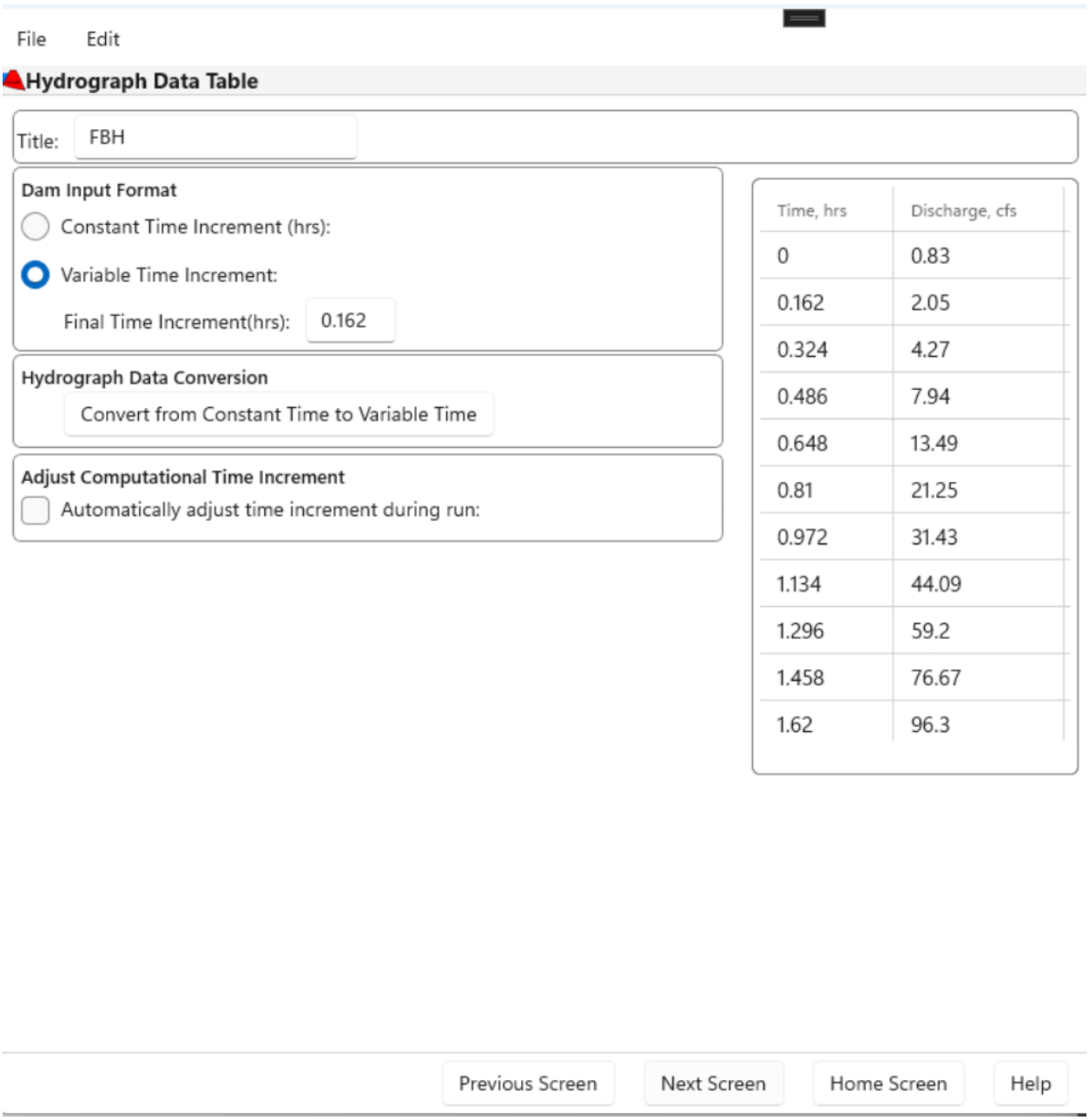


Figure 23. Hydrograph Data Table Screen

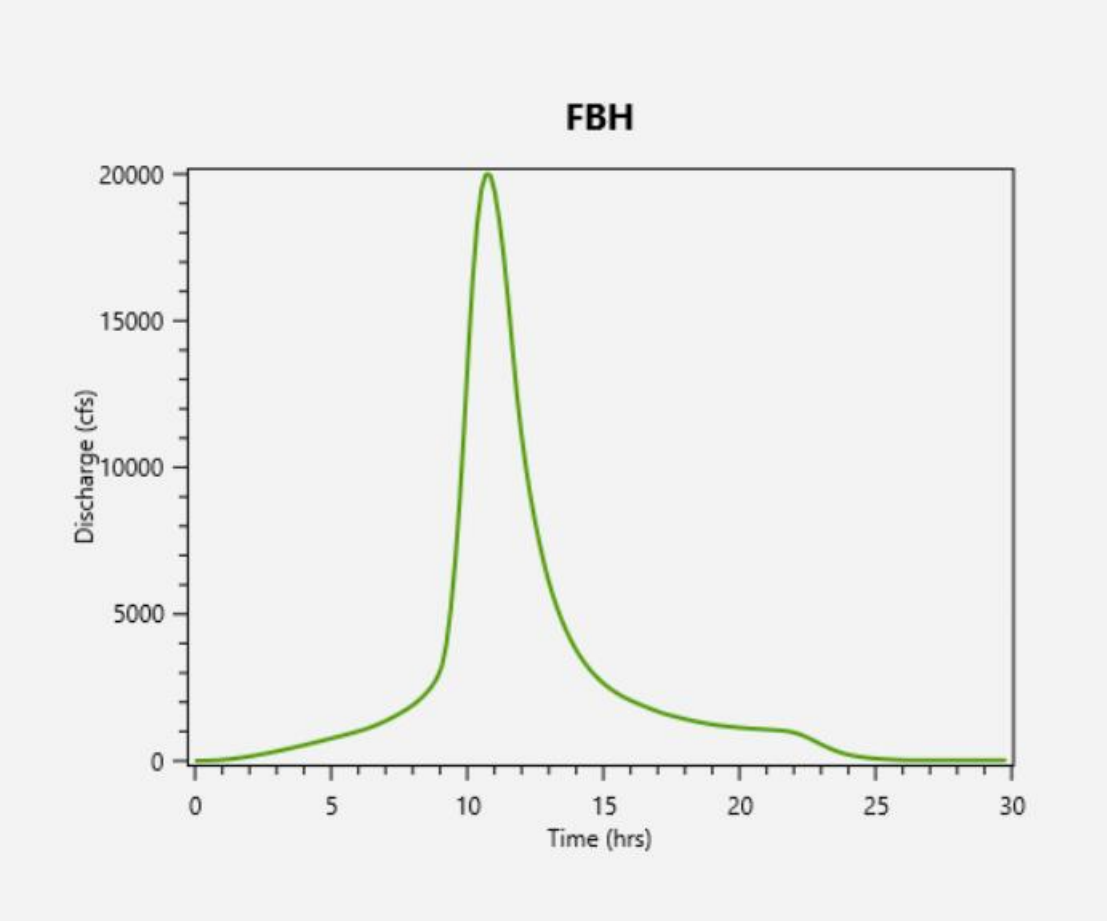


Figure 24. Hydrograph Plot

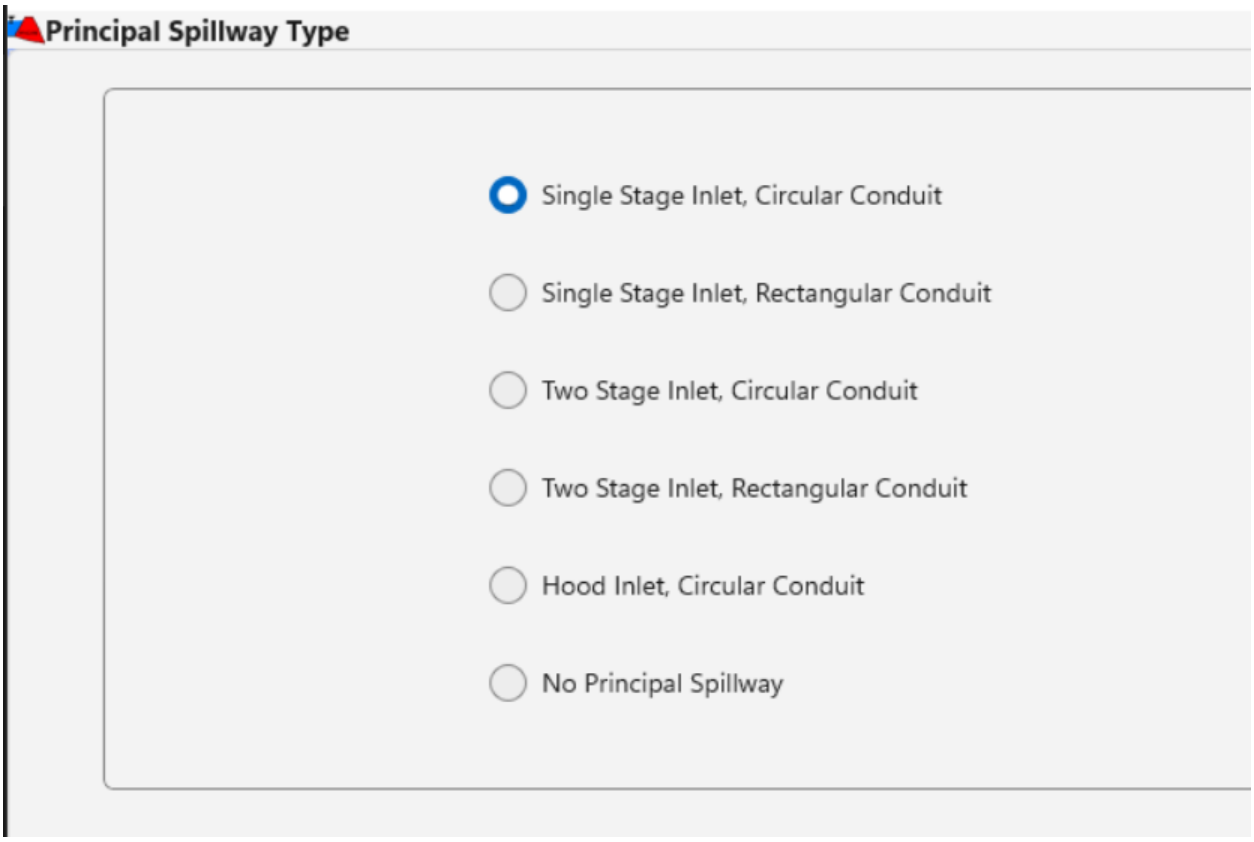


Figure 25. Principal Spillway Type Screen

Figure 26. Principal Spillway Details Screen

Figure 27. Tailwater Rating Table Screen

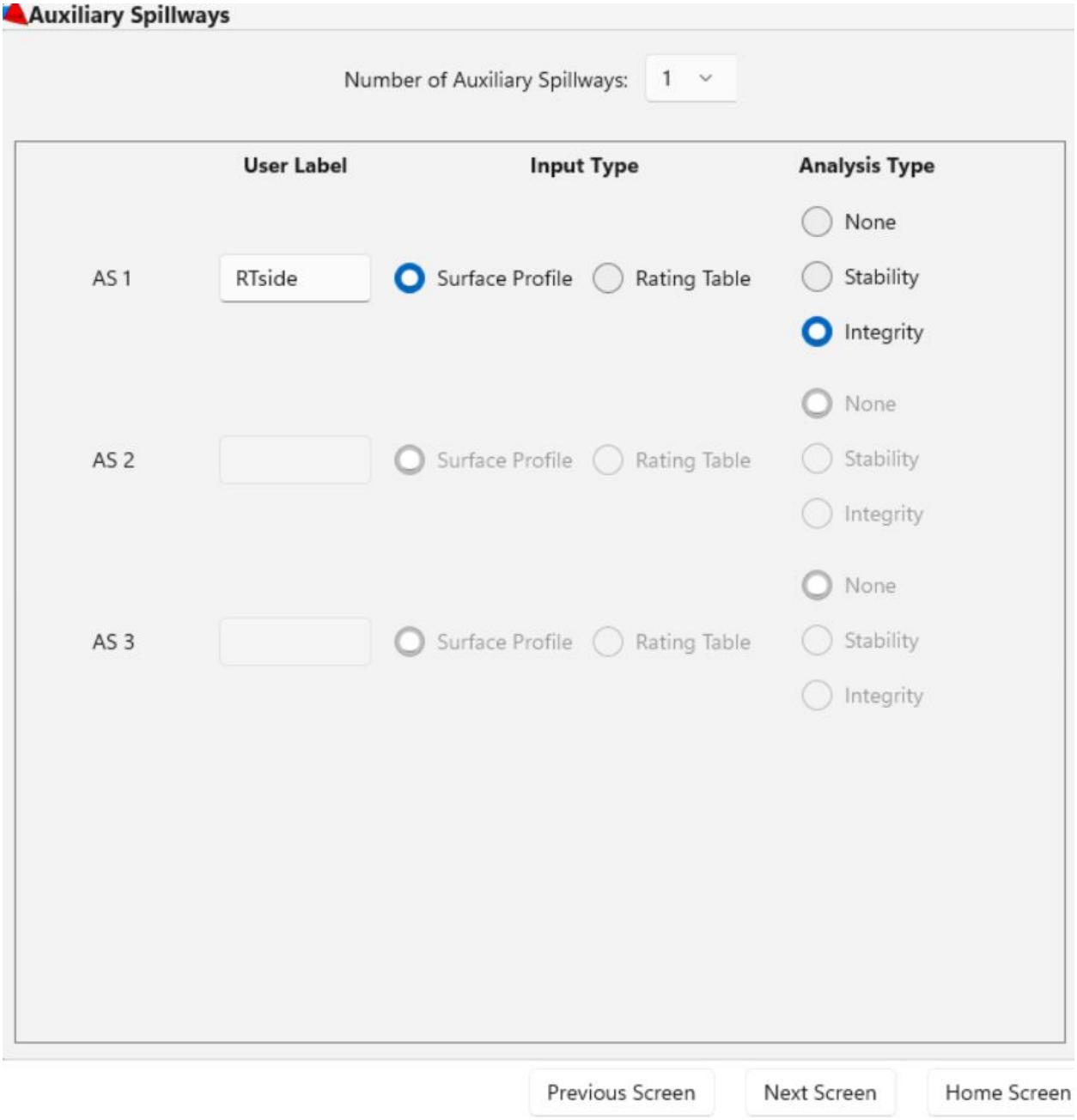
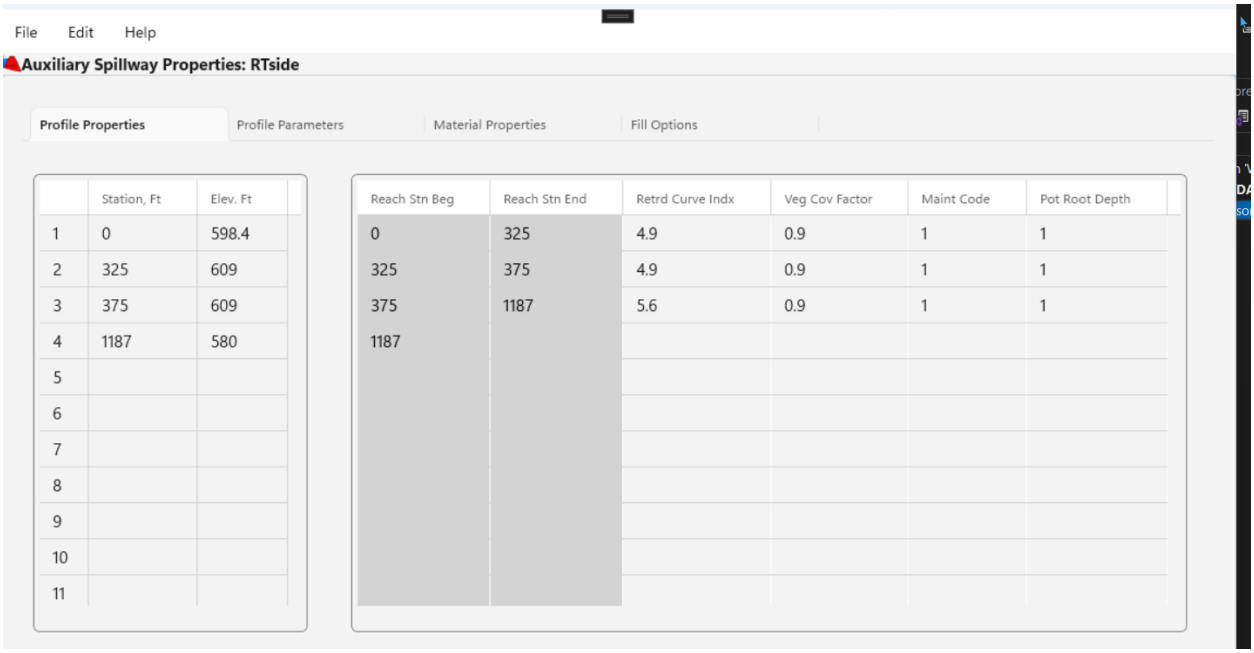


Figure 28. Aux. Spillway Info Input Screen



	Station, Ft	Elev. Ft
1	0	598.4
2	325	609
3	375	609
4	1187	580
5		
6		
7		
8		
9		
10		
11		

Reach Stn Beg	Reach Stn End	Retrd Curve Indx	Veg Cov Factor	Maint Code	Pot Root Depth
0	325	4.9	0.9	1	1
325	375	4.9	0.9	1	1
375	1187	5.6	0.9	1	1
1187					

Figure 29. Aux. Spillway Properties - Profile Properties Screen

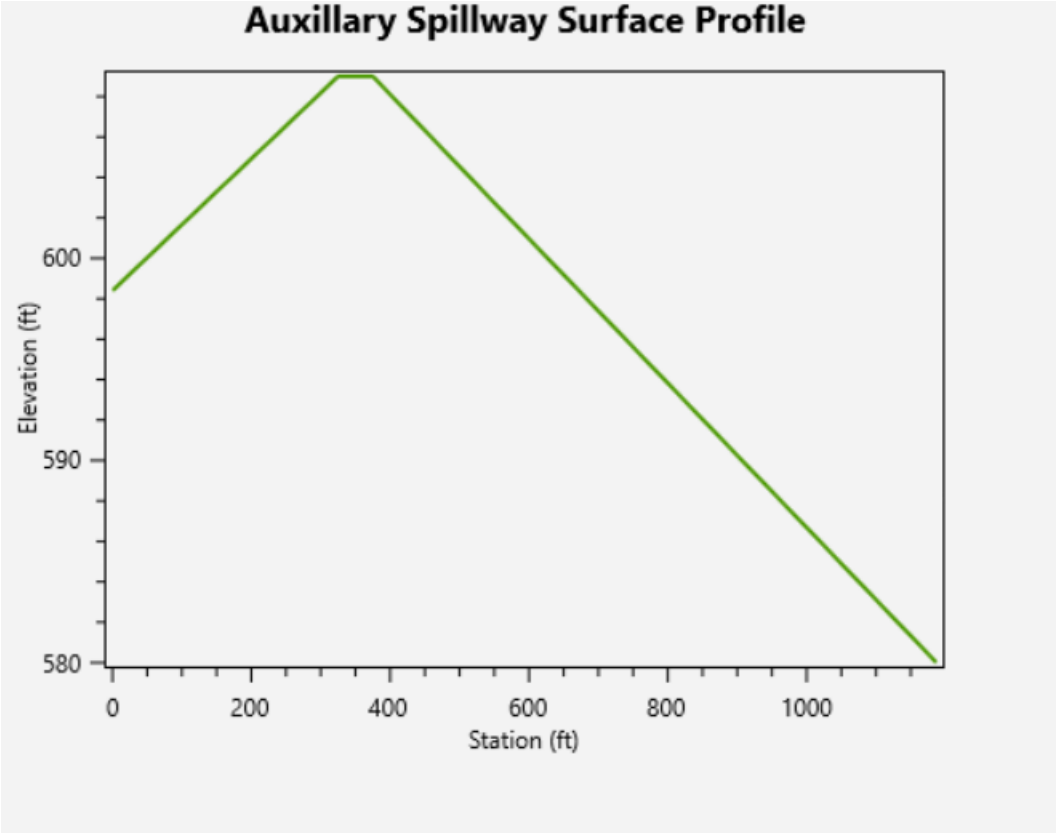


Figure 30. Aux. Spillway Surface Profile Plot

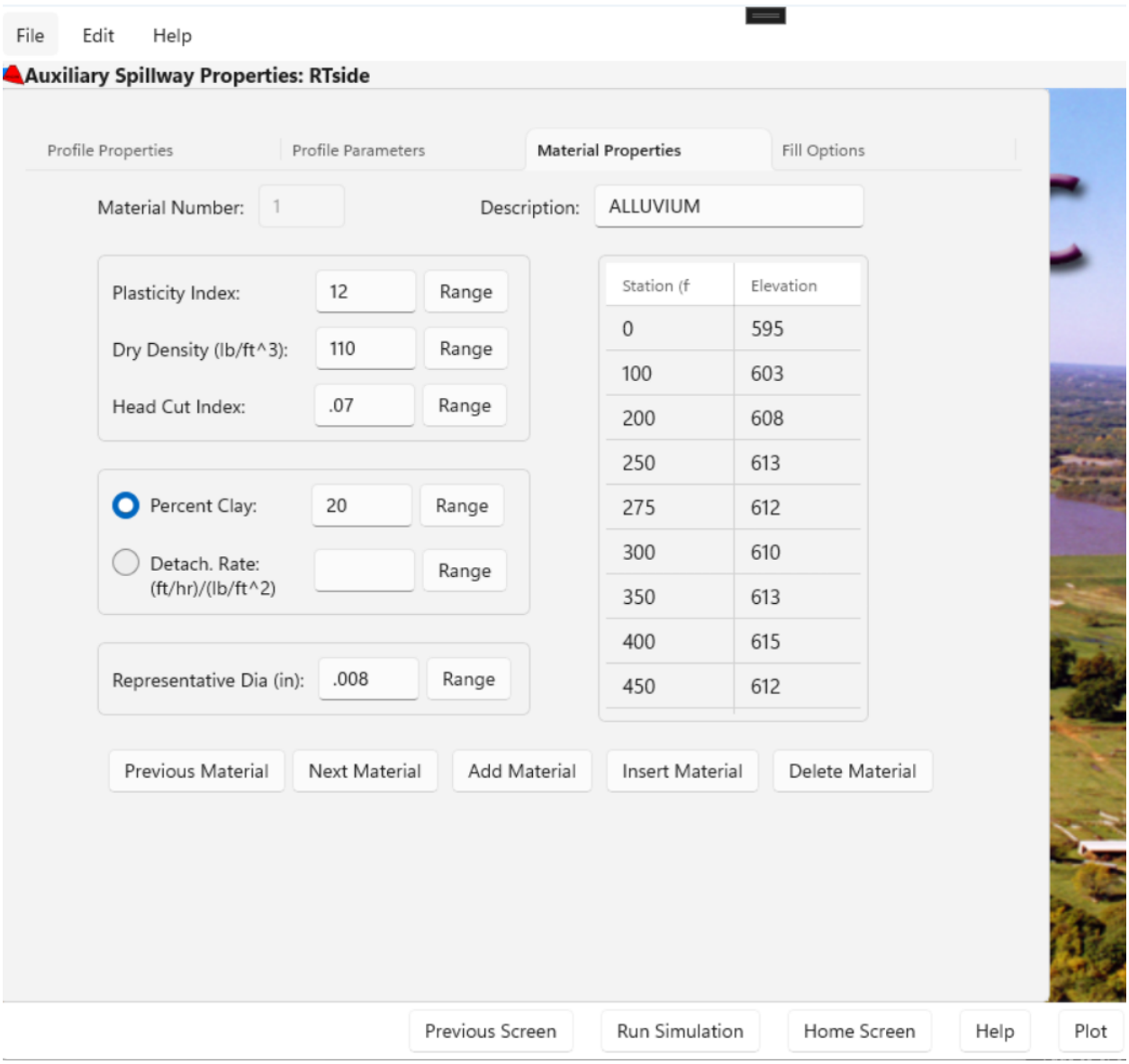


Figure 31. Aux. Spillway Properties - Material Properties Screen

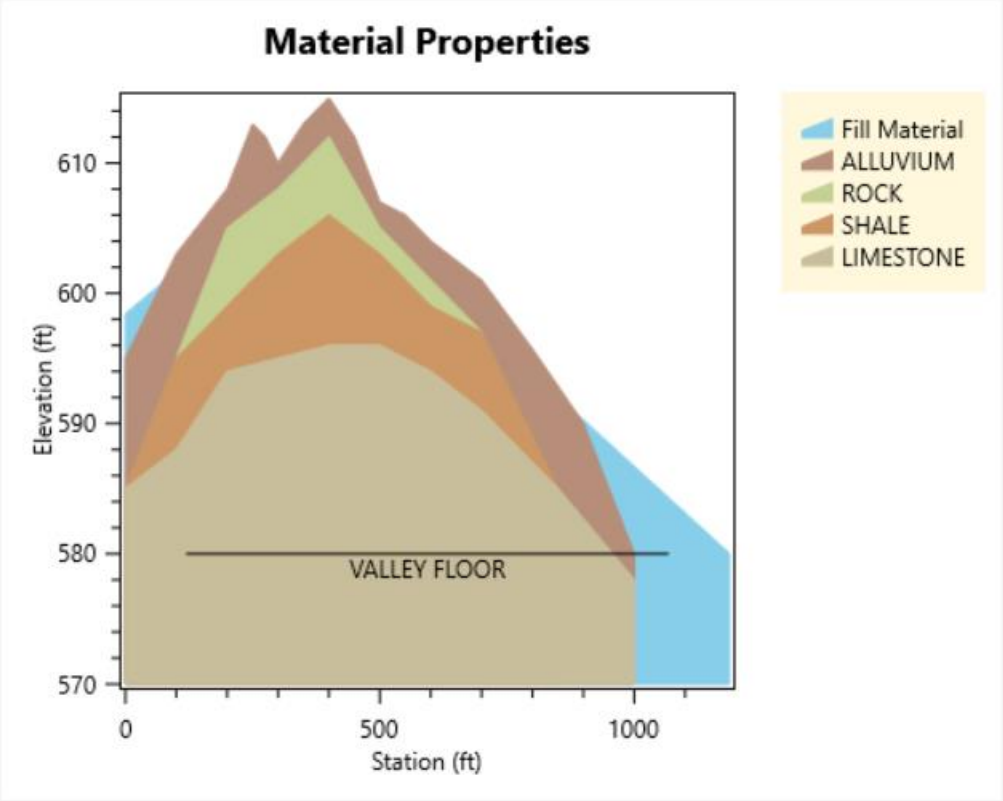


Figure 32. Aux. Spillway Properties - Material Properties Plot

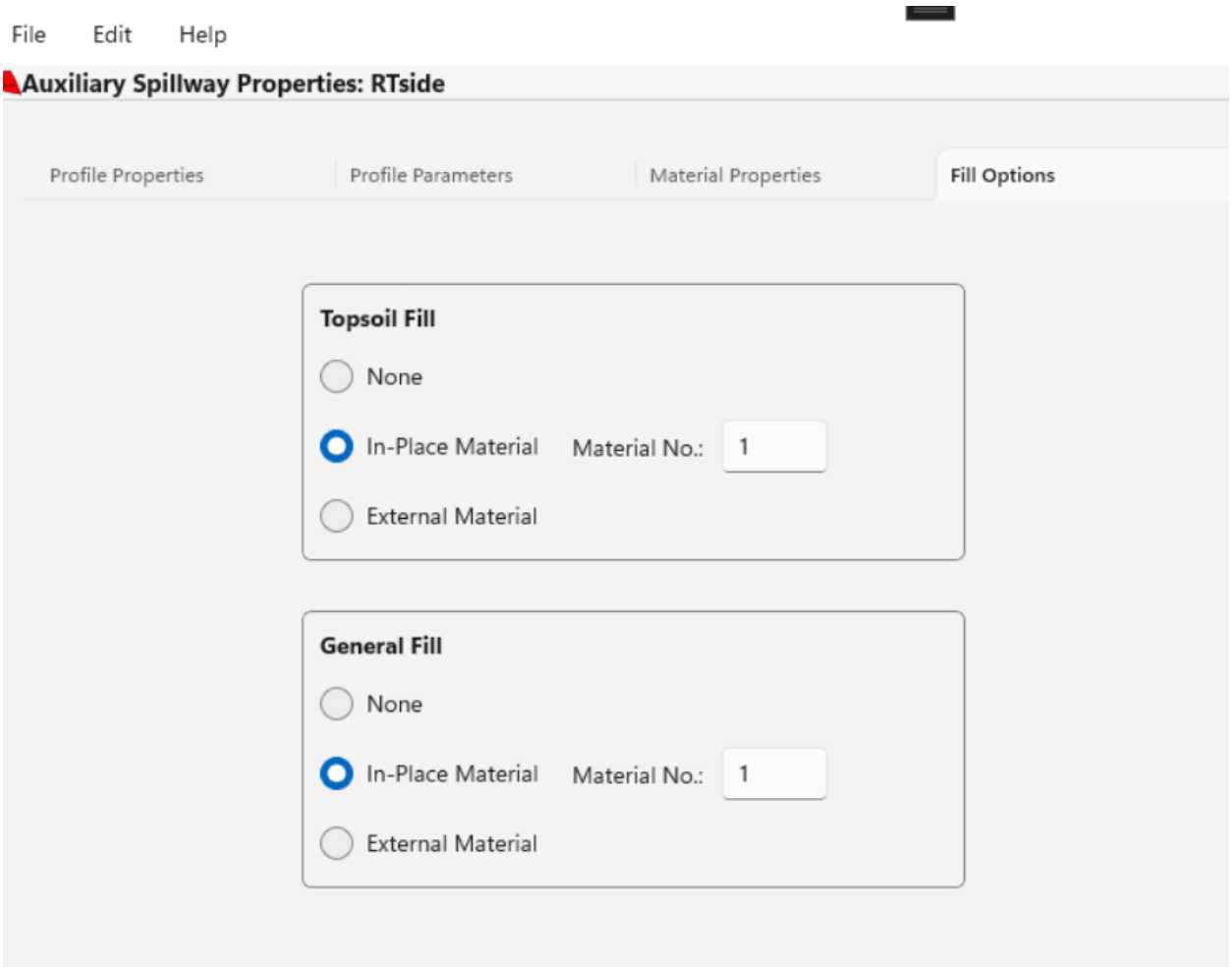


Figure 33. Aux. Spillway Properties - Fill Options Screen

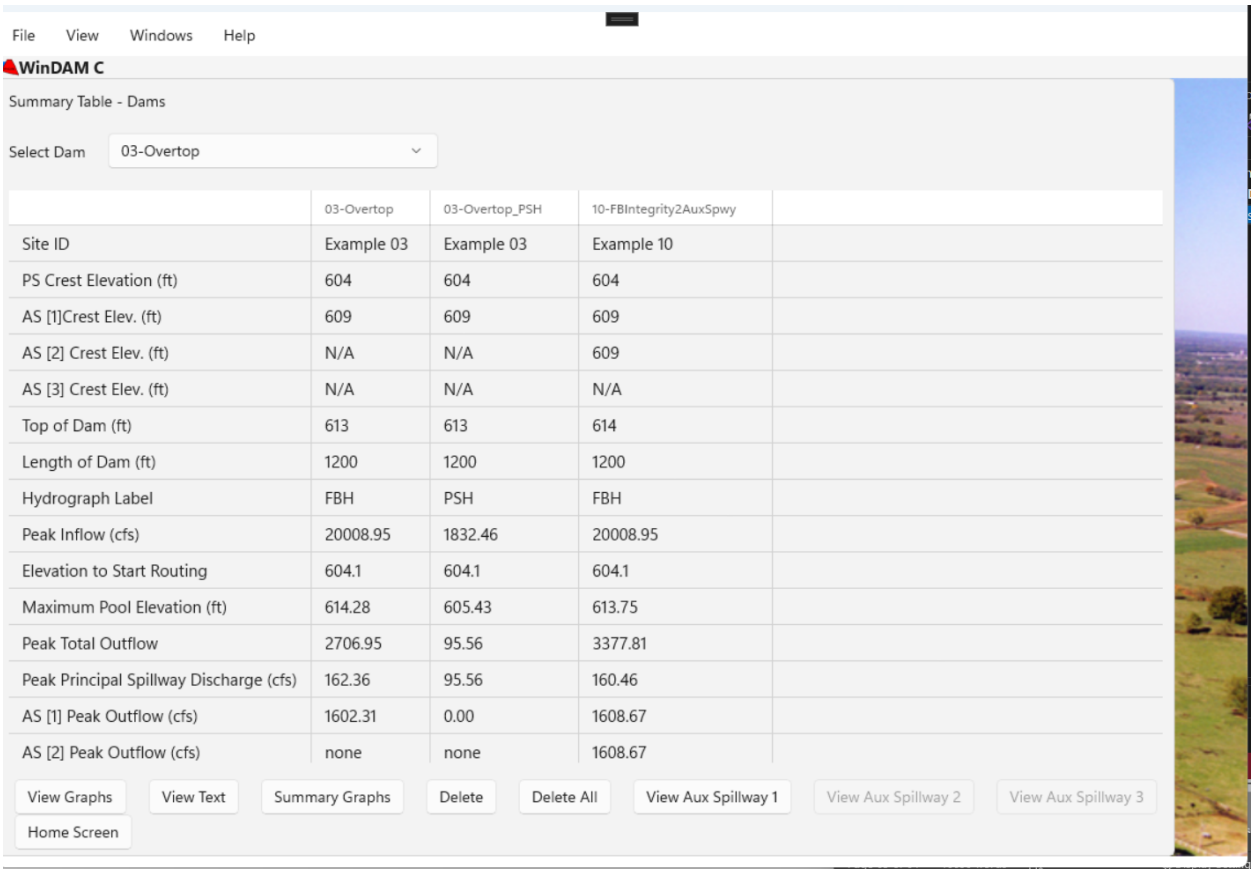


Figure 34. Summary Table – Dam Level

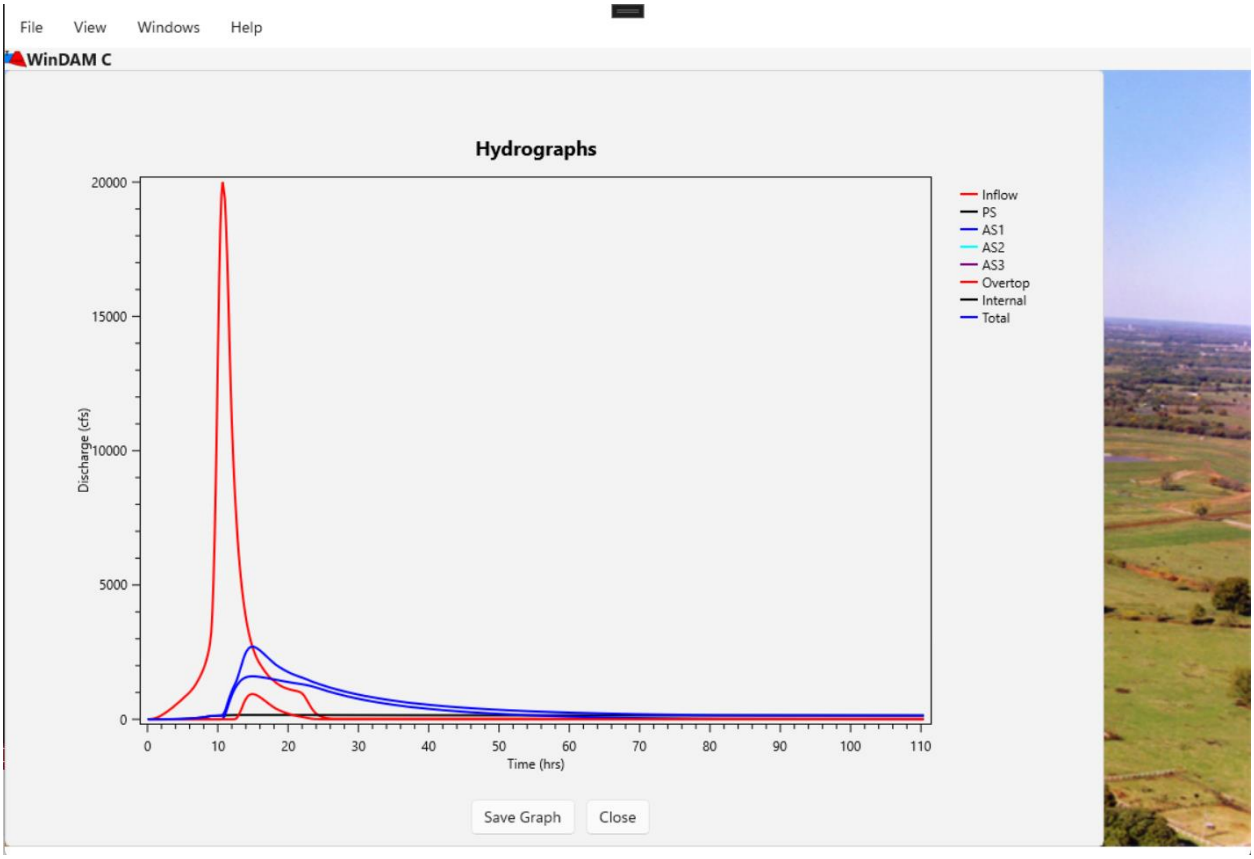


Figure 35. Summary Table - Dam - View Graphs Screen

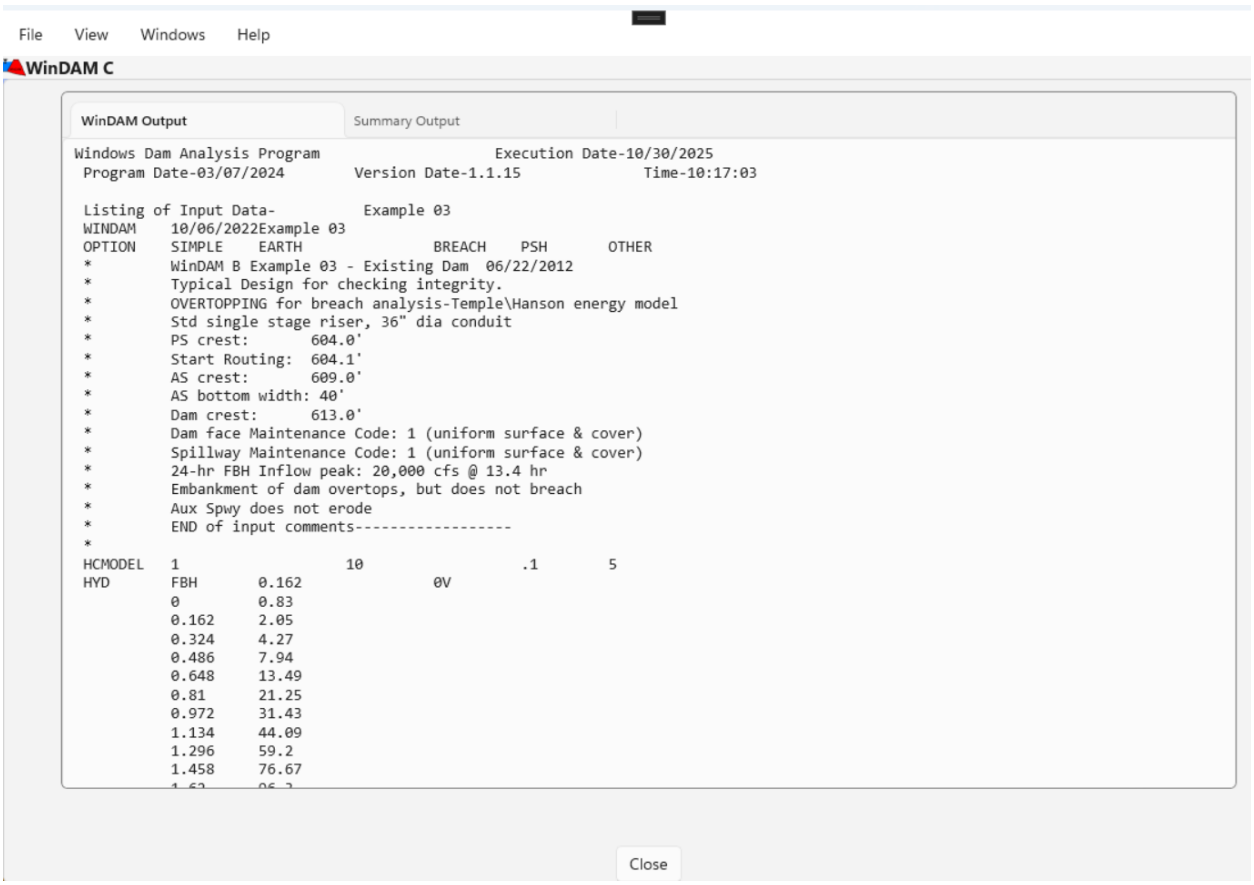


Figure 36. Summary Table - Dam - View Text Screen

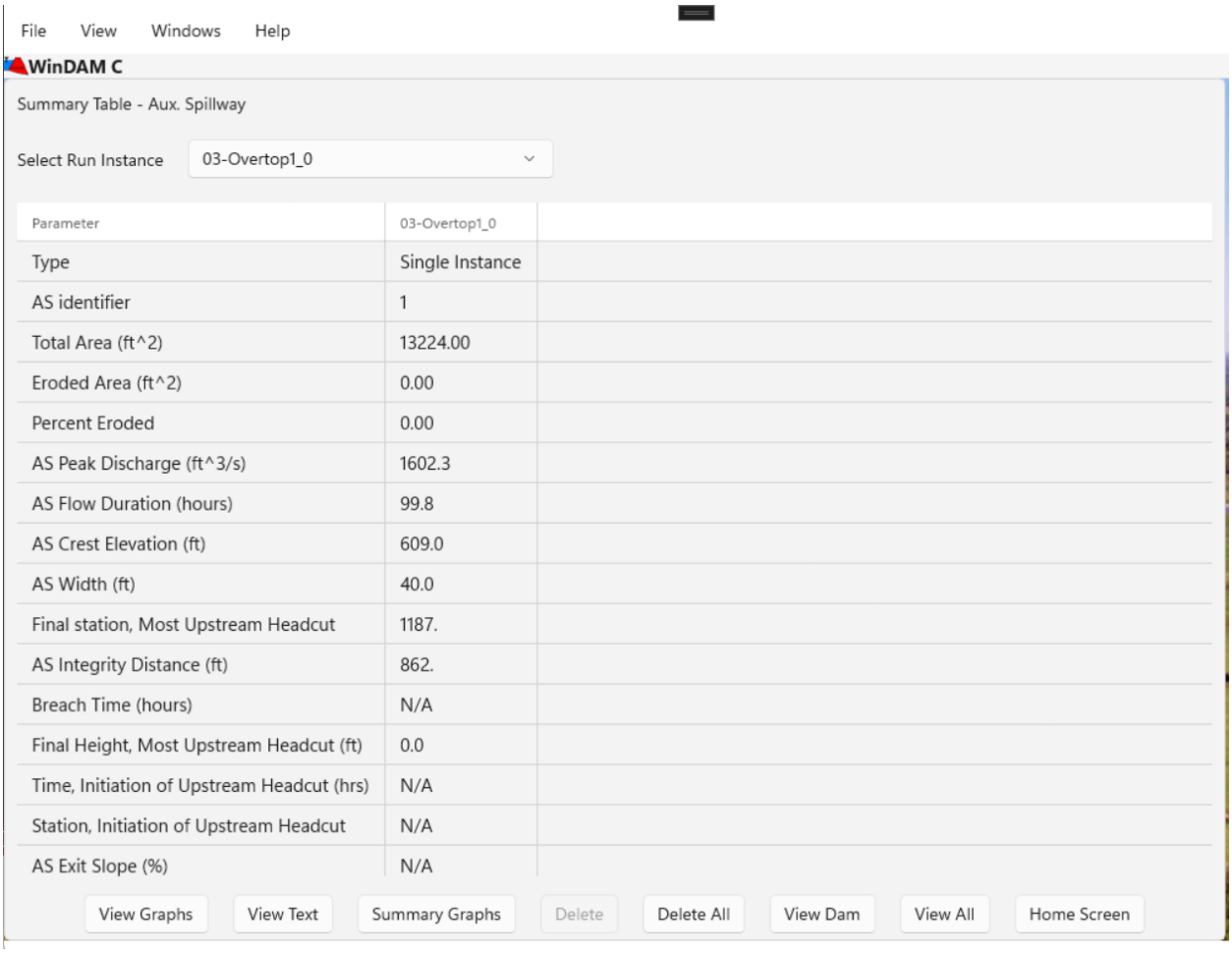


Figure 37. Summary Table - Aux Spillway

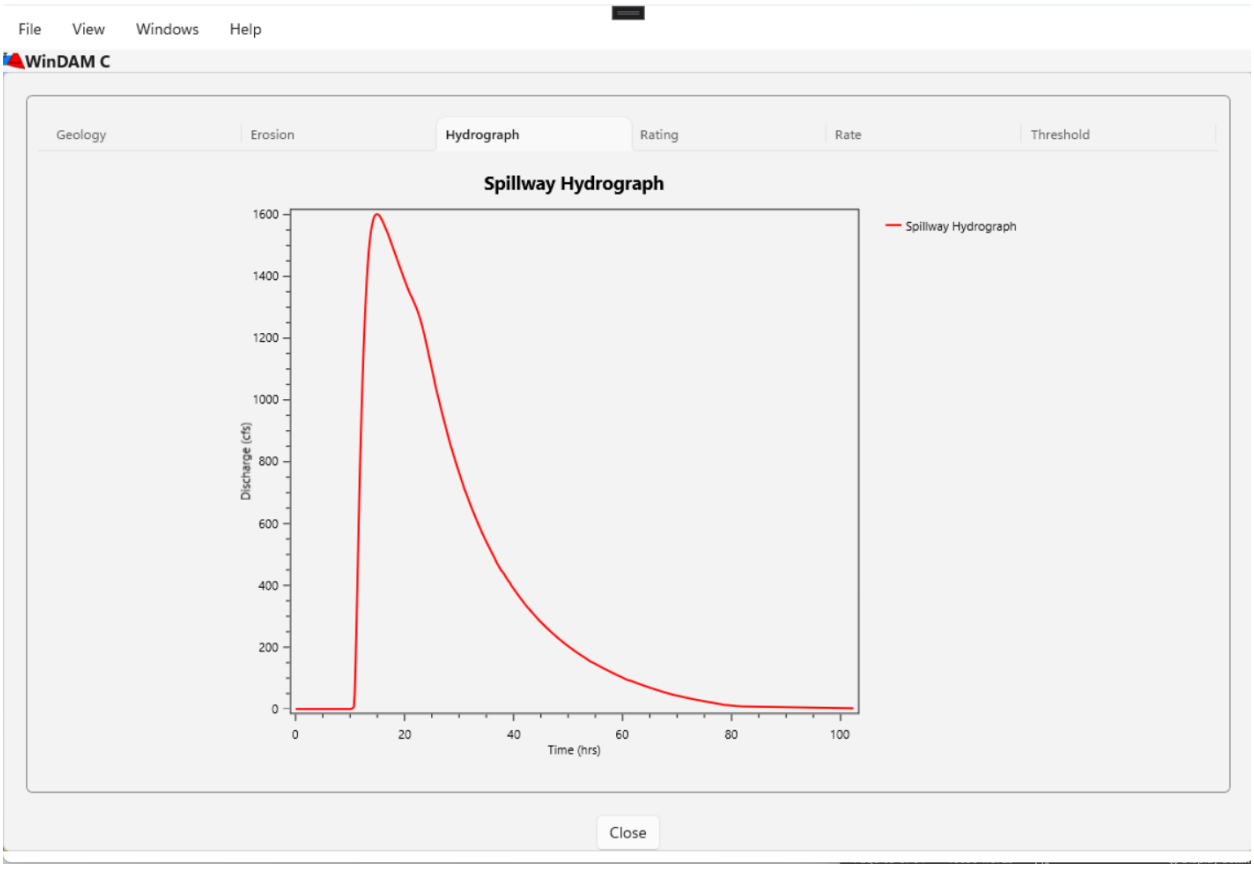


Figure 38. Summary Table - Aux. Spillway - View Graphs Screen

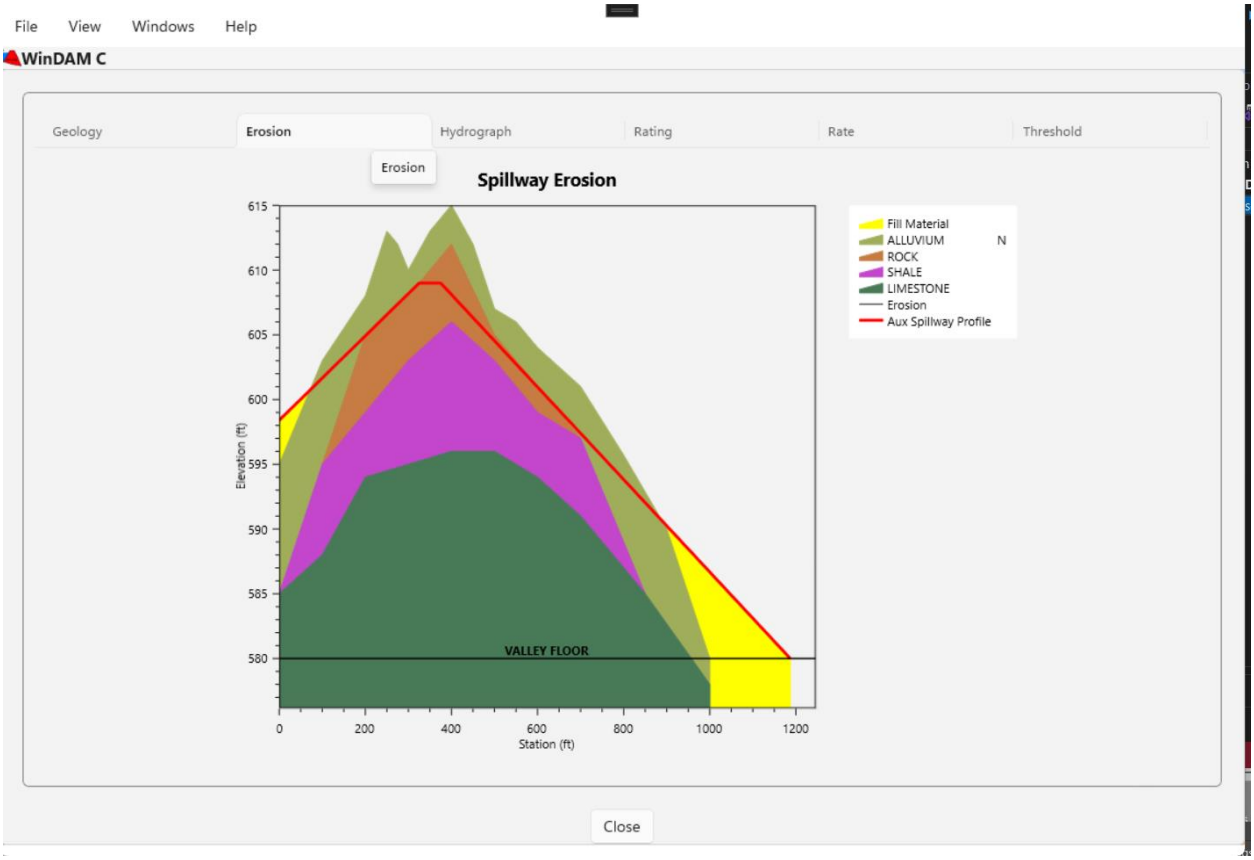


Figure 39. Summary Table – Aux. Spillway - View Erosion Screen

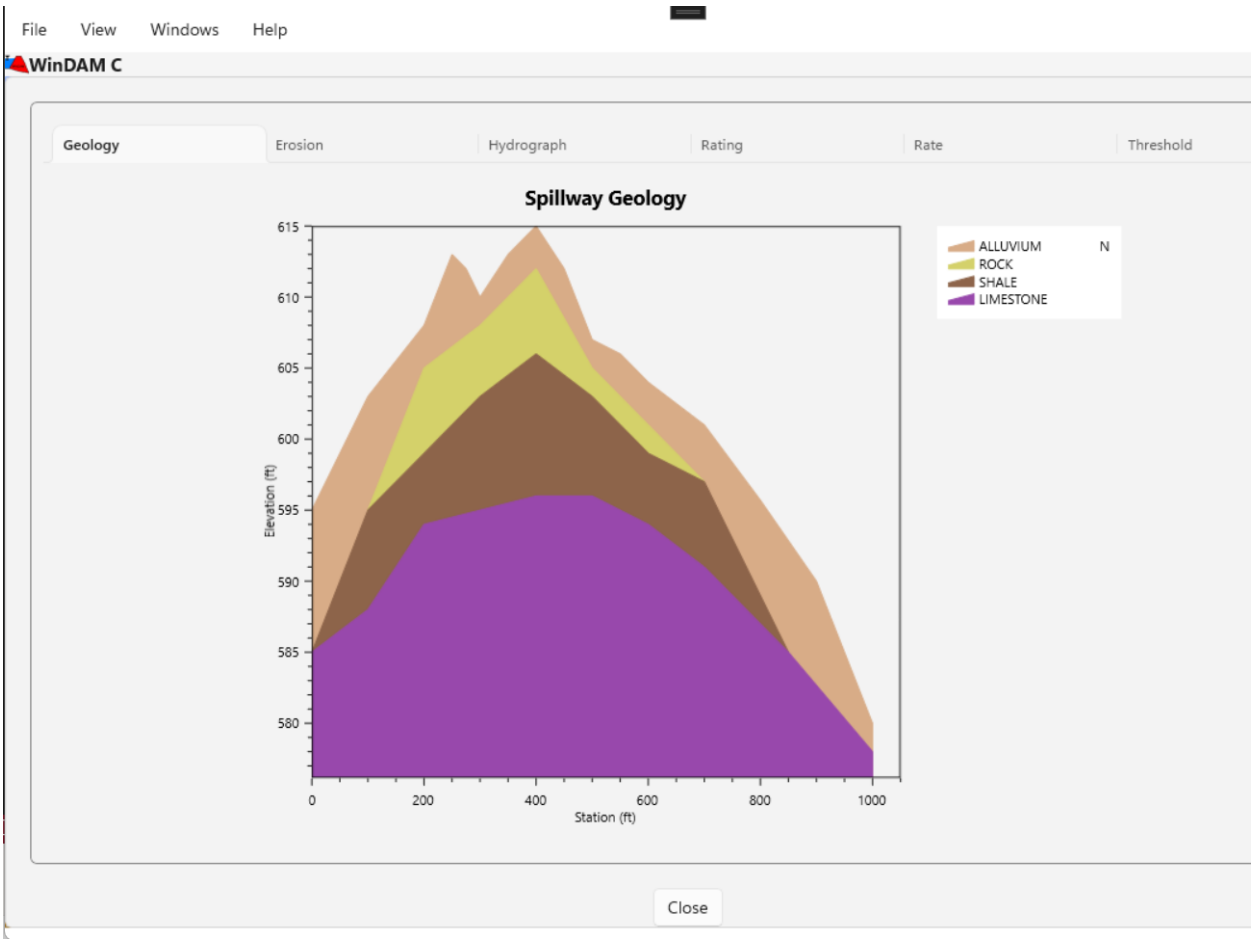


Figure 40. Summary Table - Aux. Spillway - View Geology Screen

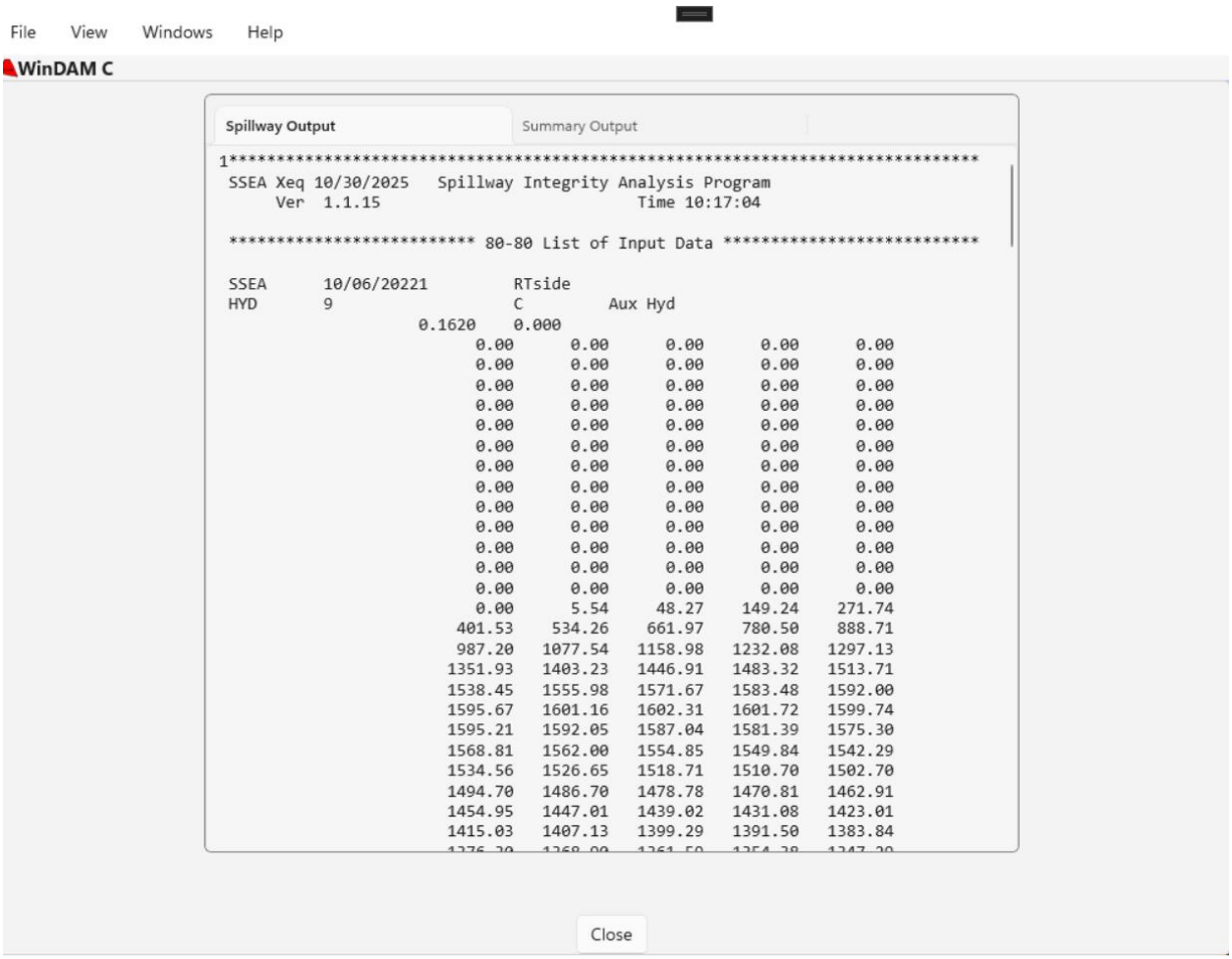


Figure 41. Summary Table - Aux. Spillway - View Text Screen