FOSSEE Semester Long Internship Report

On

Development of Parametric 2D/3D CAD Visualization Framework for OsdagBridge

Submitted by

Faizan Khan

3rd Year B.Tech Student, Department of Computer Engineering

Aligarh Muslim University

Aligarh, UP

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentors:

Nidhi Khare

Aditya Donde

May 23, 2026

Acknowledgments

I express my sincere gratitude to all who supported me during this internship at FOSSEE, IIT Bombay. I am deeply thankful to **Prof. Siddhartha Ghosh**, Principal Investigator of the Osdag project, Department of Civil Engineering, IIT Bombay, for providing this invaluable opportunity to work on cutting-edge structural engineering software development.

I am immensely grateful to my mentors **Nidhi Khare** and **Aditya Donde** for their patient guidance, technical expertise, and constant support. Their insights into pythonOCC, 3D modeling, and software development were invaluable in completing this project successfully. I also thank my fellow interns at FOSSEE for their collaboration and shared learning experiences.

I acknowledge the **National Mission on Education through ICT, Ministry of Education, Government of India** for funding this project. Finally, I am grateful to **Zakir Husain College of Engineering and Technology** and the **Department of Computer Engineering** for their encouragement in pursuing this internship opportunity.

Faizan Khan

Department of Computer Engineering

Aligarh Muslim University

Contents

Chapter 1

Introduction

1.1 National Mission in Education through ICT

The National Mission on Education through ICT (NMEICT) is a scheme under the Department of Higher Education, Ministry of Education, Government of India. It aims to leverage the potential of ICT to enhance teaching and learning in Higher Education Institutions in an anytime-anywhere mode.

The mission aligns with the three cardinal principles of the Education Policy—**access, equity, and quality**—by:

- Providing connectivity and affordable access devices for learners and institutions.
- Generating high-quality e-content free of cost.

NMEICT seeks to bridge the digital divide by empowering learners and teachers in urban and rural areas, fostering inclusivity in the knowledge economy. Key focus areas include:

- Development of e-learning pedagogies and virtual laboratories.
- Online testing, certification, and mentorship through accessible platforms like EduSAT and DTH.
- Training and empowering teachers to adopt ICT-based teaching methods.

For further details, visit the official website: www.nmeict.ac.in.

1.1.1 ICT Initiatives of MoE

The Ministry of Education (MoE) has launched several ICT initiatives aimed at students, researchers, and institutions. The table below summarizes the key details:

No.	Resource	For Students/Researchers	For Institutions
Audio-Video e-content			
1	SWAYAM	Earn credit via online courses	Develop and host courses; accept credits
2	SWAYAMPRABHA	Access 24x7 TV programs	Enable SWAYAMPRABHA viewing facilities
Digital Content Access			
3	National Digital Library	Access e-content in multiple disciplines	List e-content; form NDL Clubs
4	e-PG Pathshala	Access free books and e-content	Host e-books
5	Shodhganga	Access Indian research theses	List institutional theses
6	e-ShodhSindhu	Access full-text e-resources	Access e-resources for institutions
Hands-on Learning			
7	e-Yantra	Hands-on embedded systems training	Create e-Yantra labs with IIT Bombay
8	FOSSEE	Volunteer for open-source software	Run labs with open-source software
9	Spoken Tutorial	Learn IT skills via tutorials	Provide self-learning IT content
10	Virtual Labs	Perform online experiments	Develop curriculum-based experiments
E-Governance			
11	SAMARTH ERP	Manage student lifecycle digitally	Enable institutional e-governance
Tracking and Research Tools			
12	VIDWAN	Register and access experts	Monitor faculty research outcomes
13	Shodh Shuddhi	Ensure plagiarism-free work	Improve research quality and reputation
14	Academic Bank of Credits	Store and transfer credits	Facilitate credit redemption

Table 1.1: Summary of ICT Initiatives by the Ministry of Education

1.2 FOSSEE Project

The FOSSEE (Free/Libre and Open Source Software for Education) project promotes the use of FLOSS tools in academia and research. It is part of the National Mission on Education through Information and Communication Technology (NMEICT), Ministry of Education (MoE), Government of India.

1.2.1 Projects and Activities

The FOSSEE Project supports the use of various FLOSS tools to enhance education and research. Key activities include:

- **Textbook Companion**: Porting solved examples from textbooks using FLOSS.
- **Lab Migration**: Facilitating the migration of proprietary labs to FLOSS alternatives.
- **Niche Software Activities**: Specialized activities to promote niche software tools.
- **Forums**: Providing a collaborative space for users.
- **Workshops and Conferences**: Organizing events to train and inform users.

1.2.2 Fellowships

FOSSEE offers various internship and fellowship opportunities for students:

- Winter Internship
- Summer Fellowship
- Semester-Long Internship

Students from any degree and academic stage can apply for these internships. Selection is based on the completion of screening tasks involving programming, scientific computing, or data collection that benefit the FLOSS community. These tasks are designed to be completed within a week.

For more details, visit the official FOSSEE website.

Figure 1.1: FOSSEE Projects and Activities

1.3 Osdag Software

Osdag (Open steel design and graphics) is a cross-platform, free/libre and open-source software designed for the detailing and design of steel structures based on the Indian Standard IS 800:2007. It allows users to design steel connections, members, and systems through an interactive graphical user interface (GUI) and provides 3D visualizations of designed components. The software enables easy export of CAD models to drafting tools for construction/fabrication drawings, with optimized designs following industry best practices [1, 2, 3]. Built on Python and several Python-based FLOSS tools (e.g., PyQt and PythonOCC), Osdag is licensed under the GNU Lesser General Public License (LGPL) Version 3.

1.3.1 Osdag GUI

The Osdag GUI is designed to be user-friendly and interactive. It consists of

- **Input Dock**: Collects and validates user inputs.
- **Output Dock**: Displays design results after validation.
- **CAD Window**: Displays the 3D CAD model, where users can pan, zoom, and rotate the design.
- **Message Log**: Shows errors, warnings, and suggestions based on design checks.

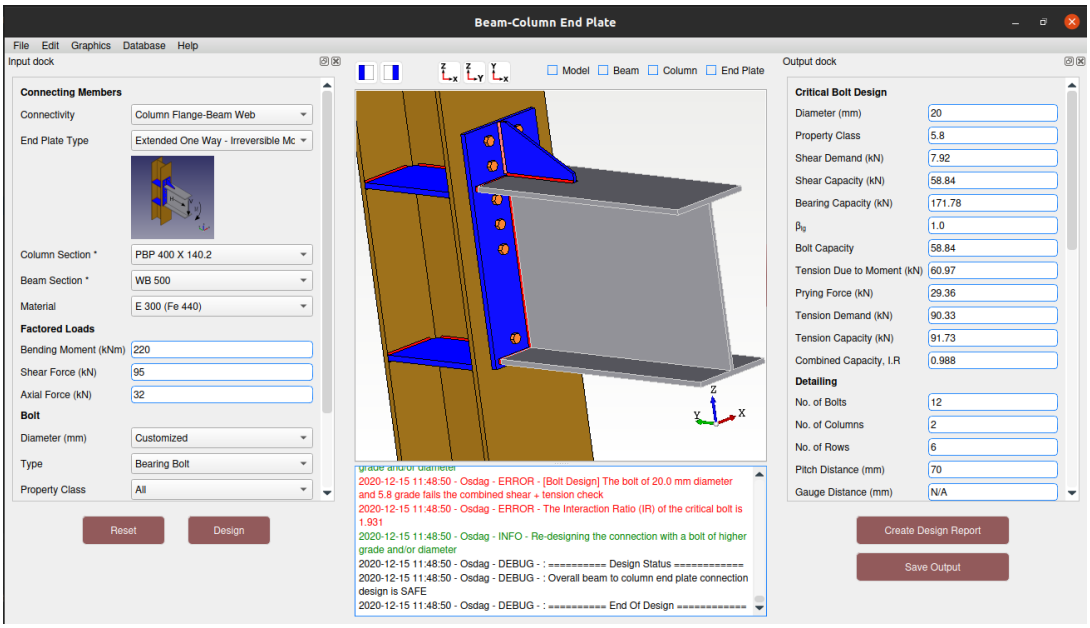

Figure 1.2: Osdag GUI

1.3.2 Features

- **CAD Model**: The 3D CAD model is color-coded and can be saved in multiple formats such as IGS, STL, and STEP.
- **Design Preferences**: Customizes the design process, with advanced users able to set preferences for bolts, welds, and detailing.
- **Design Report**: Creates a detailed report in PDF format, summarizing all checks, calculations, and design details, including any discrepancies.

For more details, visit the official Osdag website.

Chapter 2

Previous Work and Scope of Continuation

This chapter presents the continuation and extension of the bridge CAD development work initiated during the FOSSEE winter internship. The previous internship established the initial parametric three-dimensional CAD framework for steel girder bridge superstructures using pythonOCC and OpenCascade. Building upon this foundation, the semester-long internship focused on transforming the earlier standalone bridge modeling prototype into a significantly more advanced synchronized engineering visualization framework integrated within the OsdagBridge ecosystem.

Unlike the winter internship, which primarily focused on modular three-dimensional CAD generation of bridge components, the semester internship emphasized subsystem synchronization, interactive visualization architectures, centralized parameter management, drafting-oriented rendering workflows, and deeper integration between bridge geometry generation, graphical user interface operations, and engineering visualization pipelines.

The continuation work expanded the earlier bridge modeling framework through development of advanced parametric detailing systems, synchronized visualization architectures, interactive CAD workflows, and drafting-oriented sectional visualization systems integrated with bridge geometry and graphical user interface operations.

2.1 Previous Internship Work

The previous winter internship primarily focused on the development of a parametric three-dimensional CAD modeling framework for steel girder bridge superstructures. The implemented system introduced modular geometry generation strategies for major bridge components including plate girders, deck slabs, cross bracings, crash barriers, medians, and railing systems.

All bridge components were generated using parameter-driven CAD workflows to support configurable bridge geometry and rapid model regeneration. The framework followed a component-based CAD generation architecture in which individual bridge components were modeled independently and later assembled into a unified bridge superstructure model.

Reusable geometric utilities were implemented for plate generation, coordinate transformations, skew-aware placement, and assembly-level positioning operations. These utilities established the foundation for scalable bridge geometry generation within the OsdagBridge environment.

The complete bridge superstructure generated during the winter internship is shown in Figure 2.1. The generated model integrated structural and safety components within a common coordinate system while maintaining parametric adaptability for geometric modifications.

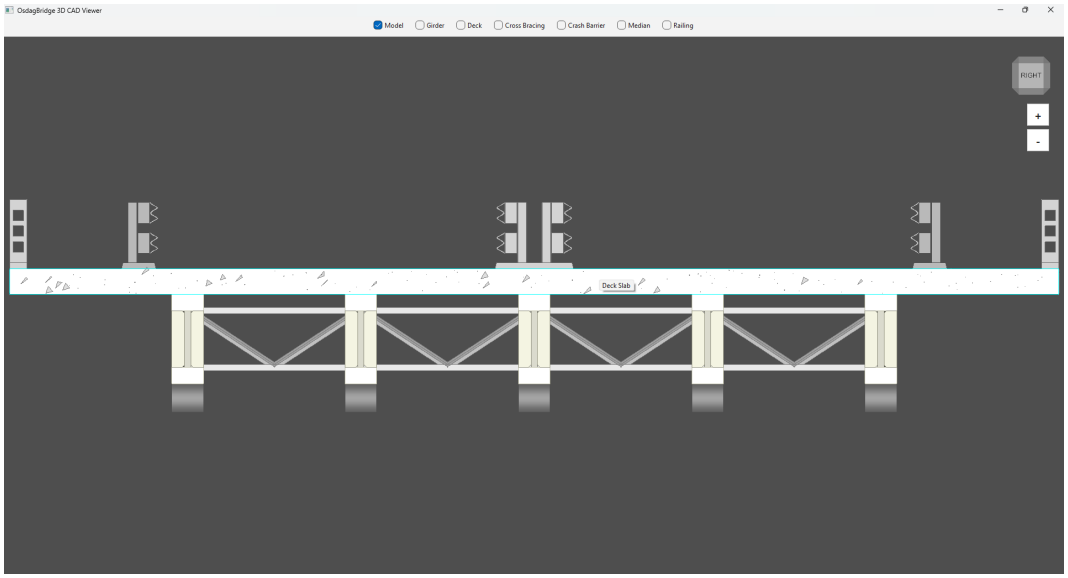

Figure 2.1: Parametric steel girder bridge superstructure developed during the winter internship

The winter internship also established the initial OpenCascade-based visualization environment for bridge inspection and interaction. Interactive operations including pan, zoom, rotation, component coloring, and CAD export support were integrated within the visualization system.

Special emphasis was placed on maintaining parametric flexibility across the bridge assembly environment. Geometric parameters such as girder spacing, skew angle, deck dimensions, stiffener spacing, and cross-bracing configurations could be modified dynamically through parameter updates, enabling rapid generation of multiple bridge configurations.

The plate girder system formed the primary structural component within the bridge superstructure. Modular plate-based modeling strategies were used for generating web plates, flange plates, stiffeners, and support components before integrating them into a complete girder assembly.

Figure 2.2 shows the generated plate girder assembly developed during the winter internship.

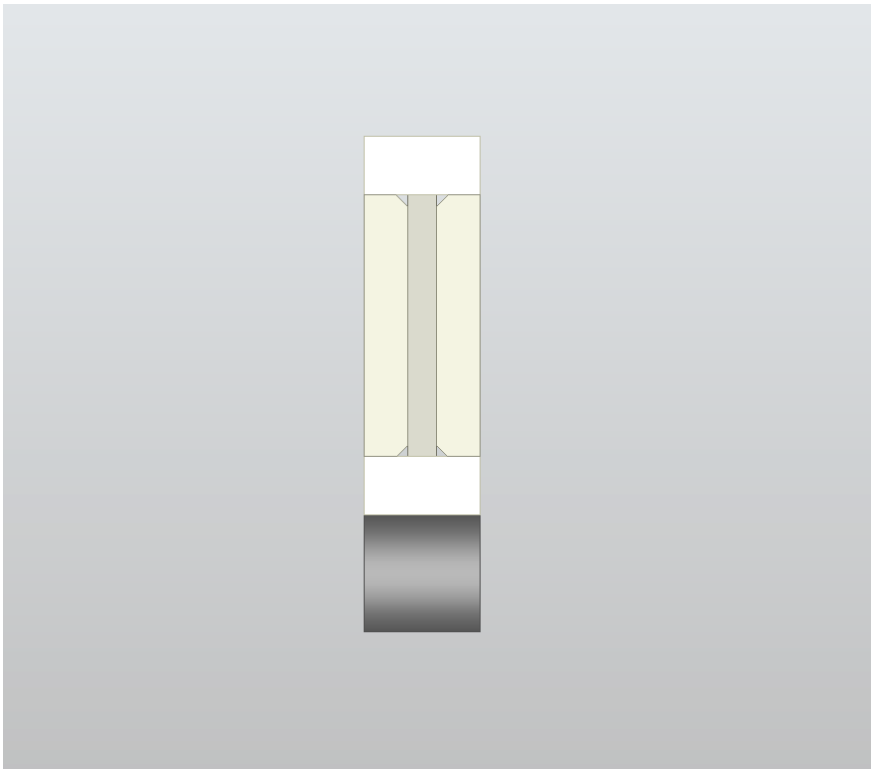

Figure 2.2: Plate girder assembly generated using the modular CAD generation framework

The framework also supported integration of deck slabs and bridge safety systems

within the overall bridge assembly environment. Figure 2.3 shows the integrated deck-girder bridge assembly generated using the earlier CAD framework.

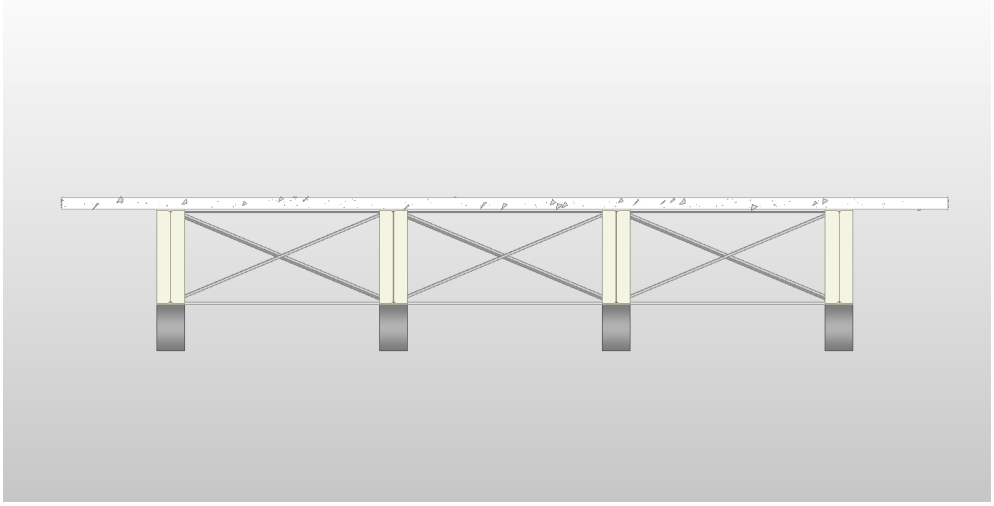

Figure 2.3: Integrated deck and girder bridge assembly generated during the winter internship

Although the winter internship successfully established the initial bridge CAD generation framework, the work primarily focused on standalone three-dimensional CAD generation and visualization operations. Advanced bridge detailing systems, synchronized CAD architectures, interactive rendering mechanisms, and drafting-oriented cross-sectional visualization environments were outside the scope of the earlier implementation.

The earlier framework extensively utilized OpenCascade geometric modeling workflows through pythonOCC bindings for topology generation, geometric transformations, bridge assembly construction, and interactive CAD visualization. These OpenCascade-based geometric foundations established the basis for the advanced synchronized rendering and visualization systems developed during the semester internship.

2.2 Scope of Continuation

The semester-long internship continued the development of the bridge CAD framework with the objective of extending the earlier bridge modeling environment into a significantly more advanced and synchronized engineering visualization system integrated within OsdagBridge.

The continuation work focused on two major development directions:

- Extension of the parametric 3D CAD framework with advanced bridge detailing systems, modular assembly architectures, synchronized rendering pipelines, and interactive visualization capabilities.

- Development of a dedicated drafting-oriented 2D CAD cross-section visualization subsystem synchronized with bridge geometry generation and graphical user interface workflows.

The advanced 3D CAD development work involved implementation of enhanced bridge detailing systems including advanced stiffener configurations, shear stud integration, configurable end-bearing systems, IRC-compliant crash barrier systems, multiple median configurations, railing systems, improved cross-bracing environments, and synchronized end diaphragm generation workflows.

The semester-long internship also established advanced interactive CAD visualization features including hover labels, component isolation, multi-component selection, synchronized rendering operations, viewport-aware scaling mechanisms, zoom-control systems, and CAD export functionality integrated within the visualization environment.

Unlike the winter internship, which focused primarily on three-dimensional bridge assembly generation, the semester-long internship introduced a complete drafting-oriented two-dimensional CAD visualization subsystem integrated directly with bridge geometry generation, rendering synchronization pipelines, and graphical user interface workflows. The developed framework established synchronized interaction between engineering parameters, sectional visualization, viewport-aware rendering operations, and dynamic drafting utilities within the OsdagBridge environment.

The developed 2D CAD environment supported synchronized sectional rendering, dynamic dimension visualization, drafting-oriented annotation systems, viewport-aware scaling operations, fit-to-screen rendering mechanisms, and real-time synchronization between graphical user interface operations and CAD visualization systems.

The developed framework enabled automatic rendering of bridge cross-sections directly from engineering parameters such as carriageway width, footpath configuration, crash barrier type, median configuration, railing systems, and deck geometry. Dynamic scaling and centering mechanisms were also incorporated to maintain drafting consistency across multiple bridge configurations.

The semester-long internship additionally emphasized deeper integration between CAD generation systems and the OsdagBridge software architecture. Modular builder-based modeling pipelines, centralized parameter synchronization mechanisms, reusable rendering utilities, synchronized viewer architectures, and extensible visualization subsystems were implemented to improve maintainability, scalability, and future extensibility of the bridge engineering framework.

Table 2.1: Evolution of the OsdagBridge CAD framework from the winter internship to the semester internship

Aspect	Winter Internship	Semester Internship
Primary Focus	Parametric 3D bridge CAD generation	Integrated engineering visualization framework
Visualization Scope	Standalone 3D visualization	Synchronized 2D and 3D CAD visualization
Architecture Style	Component-oriented CAD modules	Builder-based synchronized architecture
Parameter Management	Local parameter handling	Centralized DTO-based synchronization
Bridge Detailing	Basic structural components	Advanced detailing systems and IRC integration
Cross-Bracing System	Fixed geometric workflows	Depth-aware synchronized bracing workflows
Stiffener System	Intermediate stiffeners	Advanced end, longitudinal, and outstand stiffeners
Interaction Features	Basic visualization controls	Hover interaction, isolation, multi-selection, zoom systems
CAD Environment	3D bridge assembly generation	Interactive engineering visualization environment
Rendering Workflow	Independent rendering operations	Viewport-aware synchronized rendering architecture
Software Integration	Standalone CAD workflows	GUI-integrated synchronized engineering framework
Extensibility	Component-level extensibility	Scalable subsystem-level extensibility

Figure illustrates the evolution of the bridge CAD framework from the winter internship to the semester-long internship.

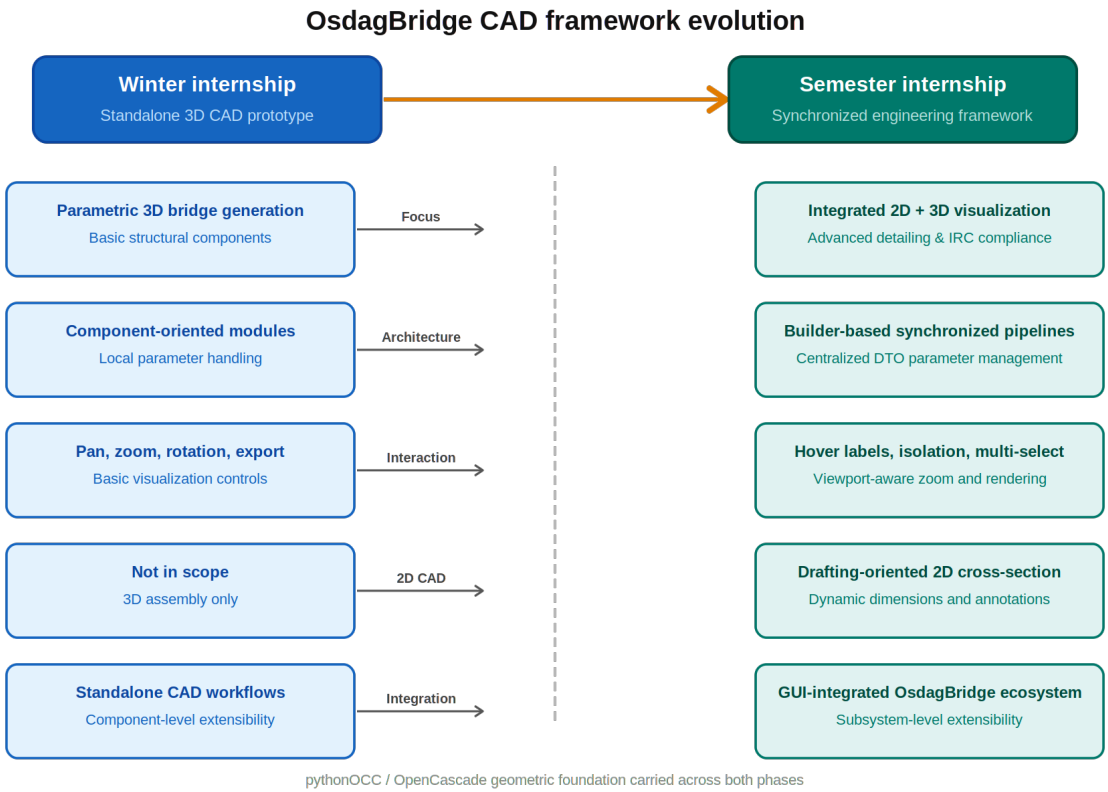

Figure 2.4: Workflow of the Parametric Bridge CAD System

Overall, the continuation work significantly extended the earlier bridge CAD framework into a more advanced engineering visualization environment capable of supporting synchronized 2D and 3D CAD systems, advanced bridge detailing workflows, interactive rendering operations, drafting-oriented visualization systems, and deeper integration within the OsdagBridge ecosystem.

The semester-long internship transformed the earlier bridge CAD prototype into a more scalable engineering framework capable of supporting synchronized visualization pipelines, interactive CAD workflows, centralized parameter management, extensible rendering architectures, and advanced drafting-oriented bridge representation systems within OsdagBridge.

Chapter 3

Advanced Parametric 3D CAD Development Results

3.1 Problem Statement

The bridge CAD framework developed during the winter internship successfully established the initial workflow for parametric three-dimensional modeling of steel girder bridge superstructures using pythonOCC and OpenCascade. The earlier implementation introduced modular CAD generation workflows for bridge components such as plate girders, deck slabs, crash barriers, railings, and cross-bracing systems. Although the framework successfully demonstrated parametric bridge generation capabilities, the overall system remained primarily focused on standalone CAD generation and visualization workflows.

The earlier implementation lacked several advanced engineering and software integration capabilities required for a scalable bridge engineering framework. The system did not support advanced bridge detailing workflows such as configurable end stiffeners, longitudinal stiffeners, shear stud systems, configurable support systems, or extensible end diaphragm environments. Similarly, synchronization between bridge geometry generation and visualization pipelines remained limited, restricting scalability of the overall CAD framework.

The visualization environment also required major architectural improvements to support interactive CAD workflows including hover-based component identification, component isolation, multi-selection, synchronized rendering, zoom-control systems, and viewer lifecycle management integrated directly within the graphical user interface environment.

Another major limitation of the earlier framework was the absence of a synchronized drafting-oriented two-dimensional CAD visualization system. The winter internship focused primarily on three-dimensional bridge generation and did not include cross-sectional rendering systems, dynamic scaling workflows, viewport-aware rendering pipelines, annotation systems, dimension management, or synchronized drafting utilities.

Furthermore, the earlier framework relied primarily on localized parameter handling and direct propagation of configuration values across geometry generation workflows. This architecture introduced synchronization limitations and increased coupling between graphical user interface operations, geometry generation systems, and visualization workflows.

Therefore, the semester-long internship focused on transforming the earlier bridge CAD prototype into a significantly more advanced engineering visualization framework capable of supporting synchronized 2D and 3D CAD environments, advanced bridge detailing systems, interactive visualization workflows, centralized parameter synchronization, modular subsystem integration, and scalable engineering extensibility within the OsdagBridge ecosystem.

3.2 Objectives

The primary objective of the semester-long internship was to extend the previously developed bridge CAD framework into a more advanced and extensible engineering visualization system integrated within OsdagBridge.

The major objectives of the internship are summarized below:

- To extend the parametric 3D CAD framework with advanced bridge detailing and assembly capabilities.
- To improve the modular plate girder generation workflow with advanced geometric synchronization and detailing systems.
- To develop configurable stiffener systems including intermediate stiffeners, end stiffeners, longitudinal stiffeners, and customizable outstand configurations.
- To integrate shear stud generation and synchronized placement workflows within the bridge superstructure assembly.

- To implement IRC-compliant crash barrier systems, median systems, and railing configurations.
- To improve cross-bracing and end diaphragm generation workflows using configurable member sections and skew-aware placement strategies.
- To develop an advanced interactive 3D CAD visualization environment integrated directly within the OsdagBridge graphical user interface.
- To implement hover-based interaction, multi-selection workflows, component isolation systems, zoom-control systems, and synchronized viewer interaction mechanisms.
- To integrate CAD export functionality within the synchronized visualization environment.
- To develop a drafting-oriented 2D CAD cross-section visualization subsystem integrated with bridge geometry and graphical user interface workflows.
- To establish centralized DTO-based synchronization workflows for bridge parameter management.
- To improve maintainability, scalability, and extensibility of the engineering framework using modular builder-oriented software architecture.

3.3 Advanced Parametric CAD Framework

The semester-long internship significantly extended the earlier bridge CAD framework by introducing a more advanced and synchronized engineering visualization architecture for bridge generation, detailing, and rendering workflows.

The developed framework extensively utilized OpenCascade geometric modeling workflows through pythonOCC bindings for topology construction, geometric transformations, AIS-based visualization, compound assembly management, and synchronized rendering operations.

Unlike the earlier implementation, which primarily focused on standalone bridge geometry generation, the semester internship emphasized synchronization between geome-

try generation systems, graphical user interface operations, rendering architectures, and drafting-oriented visualization pipelines.

The framework followed a modular builder-oriented CAD generation strategy in which bridge subsystems were generated independently and later integrated into a synchronized bridge assembly environment. Separate generation workflows were implemented for:

- plate girders,
- stiffener systems,
- deck slabs,
- shear studs,
- cross bracings,
- end diaphragms,
- crash barriers,
- medians,
- railing systems,
- support components.

This modular organization established clear separation between geometry generation, visualization management, parameter synchronization, and application-level interaction workflows.

The implemented architecture significantly improved maintainability and enabled scalable integration of additional engineering workflows within OsdagBridge.

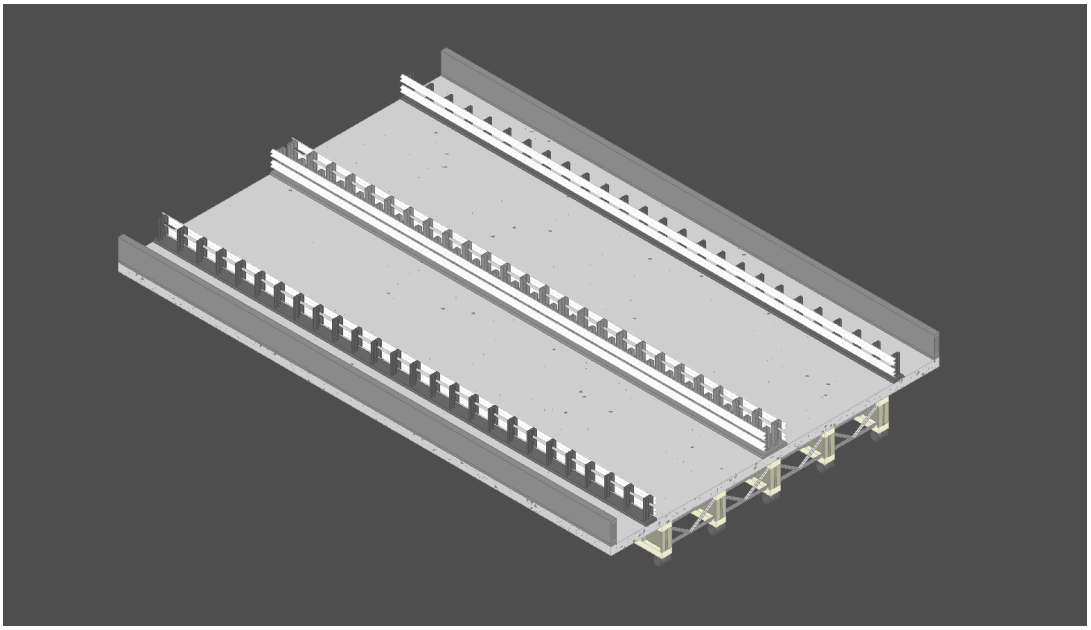

Figure 3.1: Complete parametric bridge CAD assembly generated within the synchronized OsdagBridge framework

3.4 Modular CAD Generation Pipeline

One of the major architectural improvements introduced during the semester internship was the development of a modular CAD generation pipeline for synchronized bridge assembly generation.

The developed framework separated geometry generation, parameter synchronization, rendering management, and visualization workflows into independent but coordinated engineering subsystems.

Dedicated geometry builders were implemented for generating raw OpenCascade `TopoDS_Shape` objects corresponding to individual bridge components. These geometry builders were coordinated through a centralized CAD generation pipeline responsible for parameter synchronization, bridge assembly orchestration, and rendering management.

Listing 3.1: Modular bridge assembly generation workflow

```
pg = build_plate_girder_geometry(
    D=self.girder_section_d,
    tw=self.girder_section_tw,
    length=self.span_length_L,
    segments=current_segments,
```

```
    include_intermediate_stiffeners=
        self.include_intermediate_stiffeners,
    include_longitudinal_stiffeners=
        self.include_longitudinal_stiffeners
)
```

The implemented workflow significantly improved subsystem independence and simplified integration of advanced bridge detailing systems.

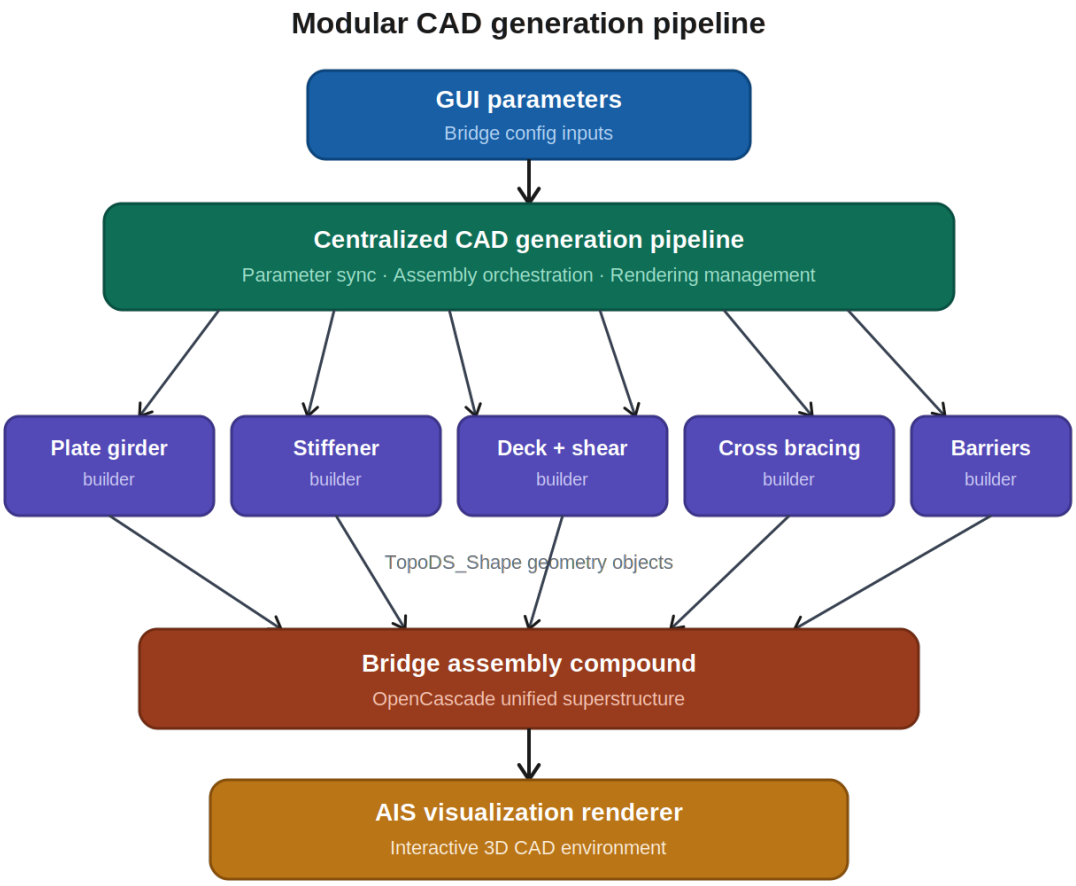

Figure 3.2: Modular CAD generation and synchronized assembly workflow

3.5 DTO-Based Parameter Synchronization Architecture

One of the most important software-engineering improvements introduced during the semester internship was the implementation of a centralized DTO-based parameter syn-

chronization framework.

The earlier CAD implementation relied primarily on localized parameter handling and direct propagation of configuration values across geometry generation functions. Although this approach was sufficient for smaller standalone workflows, it introduced synchronization limitations and scalability challenges during integration of advanced detailing systems and visualization pipelines.

To address these limitations, structured Data Transfer Object (DTO) workflows were implemented to establish centralized synchronization between graphical user interface operations, CAD generation systems, rendering pipelines, and visualization architectures.

The implemented DTO framework encapsulated bridge parameters into dedicated dataclass-based engineering configuration models responsible for maintaining synchronized bridge-state representation across all subsystems.

Listing 3.2: DTO-based bridge parameter synchronization

```
@dataclass
class BridgeParametersDTO:

    steel_grade: str
    concrete_grade: str

    span_length_L: float
    girder_section_d: float

    num_girders: int
    girder_spacing: float

    skew_angle: float

    barrier_type: str
    median_type: str

    bracing_type: str
```

The DTO architecture significantly reduced coupling between graphical user interface

workflows and geometry generation systems while improving maintainability, synchronization consistency, and subsystem extensibility.

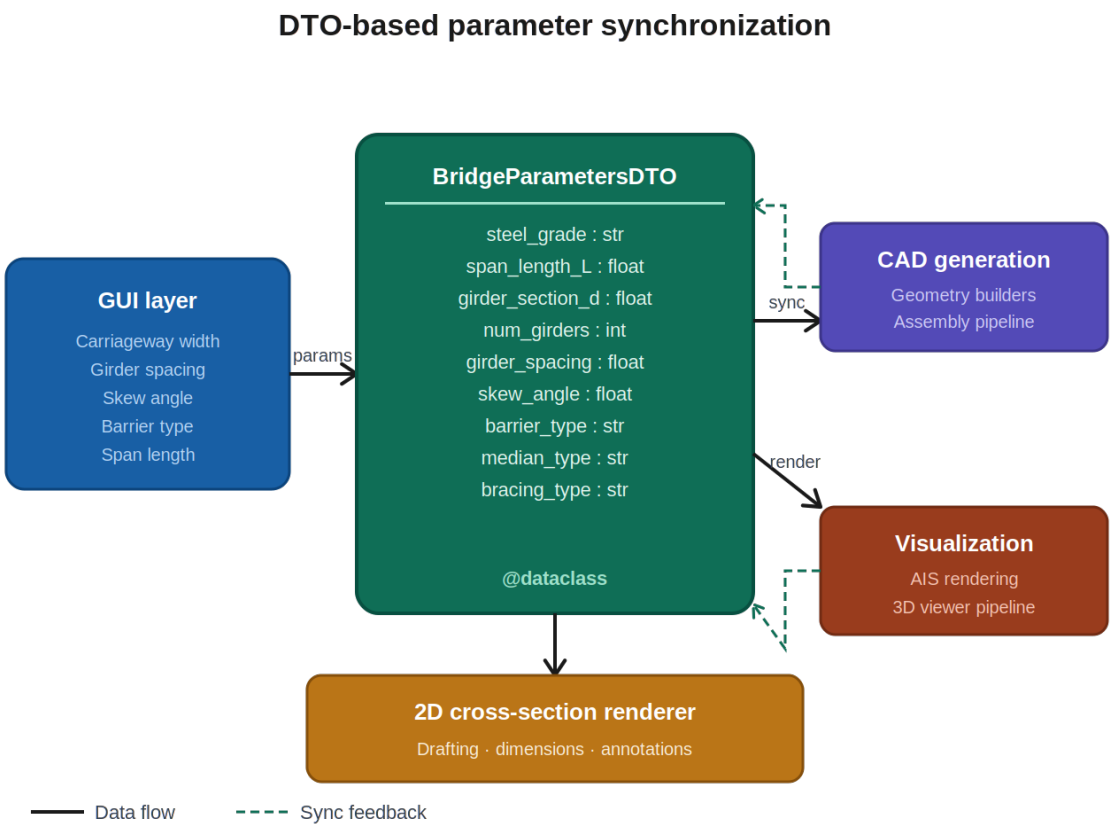

Figure 3.3: DTO-based synchronization architecture between GUI workflows, CAD generation systems, and visualization pipelines

3.6 Advanced Stiffener and Shear Stud Systems

The semester internship significantly extended the earlier detailing framework through development of configurable stiffener and shear stud systems integrated directly within the plate girder assembly environment.

The developed framework supported:

- intermediate stiffeners,
- end stiffener pairs,
- longitudinal stiffeners,

- configurable outstand options,
- customizable spacing configurations,
- synchronized shear stud placement.

Longitudinal stiffeners were integrated to improve engineering realism of the plate girder assemblies while configurable outstand workflows enabled flexible geometric control for detailing alignment.

The implemented shear stud workflow additionally supported configurable stud dimensions, pitch spacing, transverse spacing, and section-wise placement synchronization.

Listing 3.3: Configurable shear stud generation workflow

```
create_shear_stud(
    base_dia,
    base_height,
    top_dia,
    top_height
)
```

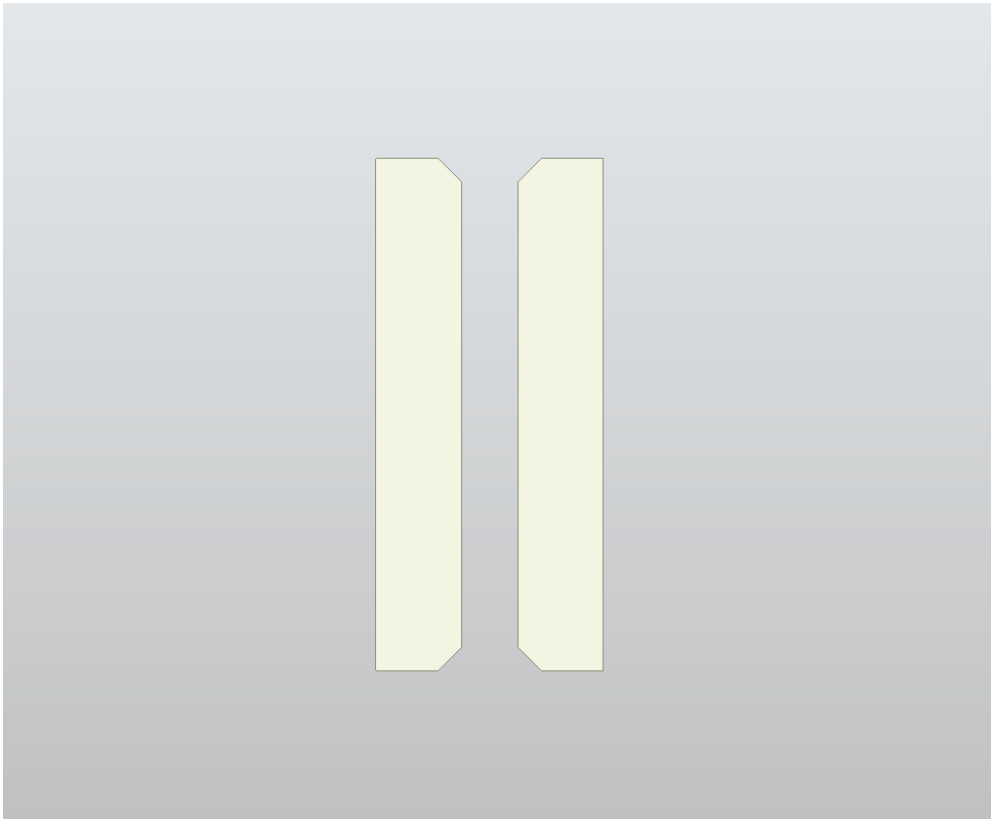

Figure 3.4: Advanced stiffener integration within the plate girder framework

3.7 Cross-Bracing and End Diaphragm System

The semester internship also introduced a significantly more advanced cross-bracing and diaphragm generation framework supporting configurable member sections, skew-aware placement logic, and synchronized assembly integration.

The developed framework supported multiple section configurations including:

- angle sections,
- channel sections,
- double-angle sections,
- double-channel sections,
- I-sections.

The framework additionally supported:

- X-bracing systems,
- K-bracing systems,
- configurable bracket options,
- skew-aware bridge configurations,
- synchronized end diaphragm generation.

Listing 3.4: Cross-bracing section generation workflow

```
def _create_angle_section(
    length,
    leg_h,
    leg_w,
    thickness
):
```

Dynamic synchronization between girder geometry and bracing placement significantly improved realism and geometric consistency across bridge assemblies.

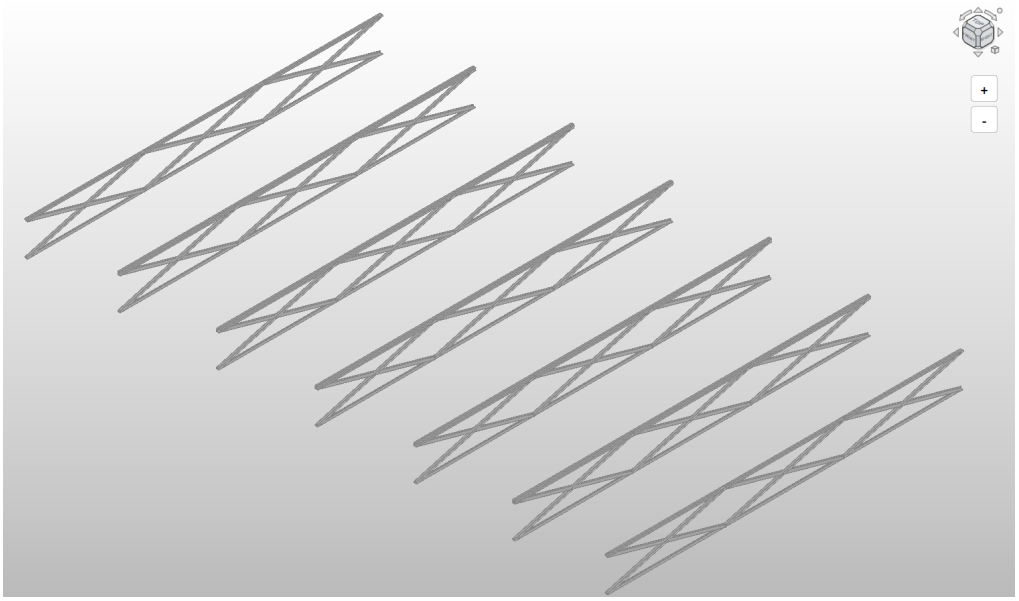

Figure 3.5: Advanced cross-bracing and diaphragm integration workflow

3.8 Interactive 3D CAD Visualization System

One of the most significant developments carried out during the semester internship was the implementation of a synchronized interactive three-dimensional CAD visualization framework integrated directly within the OsdagBridge application environment.

The developed visualization architecture utilized OpenCascade AIS (Application Interactive Services) workflows through pythonOCC bindings for rendering management, object registration, interaction control, hover synchronization, and component-level visualization.

The visualization framework was implemented using a dedicated `CAD3DWindow` architecture integrated with a customized OpenCascade viewer environment.

Listing 3.5: 3D CAD viewer initialization workflow

```
self.viewer = CustomViewer3d(self)

self.viewer.setMouseTracking(True)

self.layout.addWidget(self.viewer)
```

```
QTimer.singleShot(
    0,
    self._deferred_init_driver
)
```

The implemented viewer lifecycle architecture additionally incorporated deferred initialization workflows and controlled rendering synchronization mechanisms to ensure stable startup behavior and safe CAD initialization.

3.8.1 AIS-Based Rendering Architecture

Bridge subsystems were registered independently as AIS objects within the OpenCascade interactive context to support synchronized interaction and subsystem-level rendering management.

Listing 3.6: AIS-based rendering workflow

```
ais = display.DisplayShape(
    shp,
    color=color,
    transparency=transparency,
    update=False
)

context.Activate(ais, 0)
```

This workflow enabled:

- hover-based interaction,
- component identification,
- subsystem isolation,
- multi-selection workflows,
- synchronized rendering updates.

3.8.2 Component-Oriented Rendering Management

Separate rendering workflows were implemented for:

- girder webs,
- flange systems,
- stiffeners,
- deck systems,
- cross bracings,
- supports,
- crash barriers,
- medians,
- railing systems.

Distinct rendering styles and color-coded visualization strategies were implemented to improve engineering readability within complex bridge assemblies.

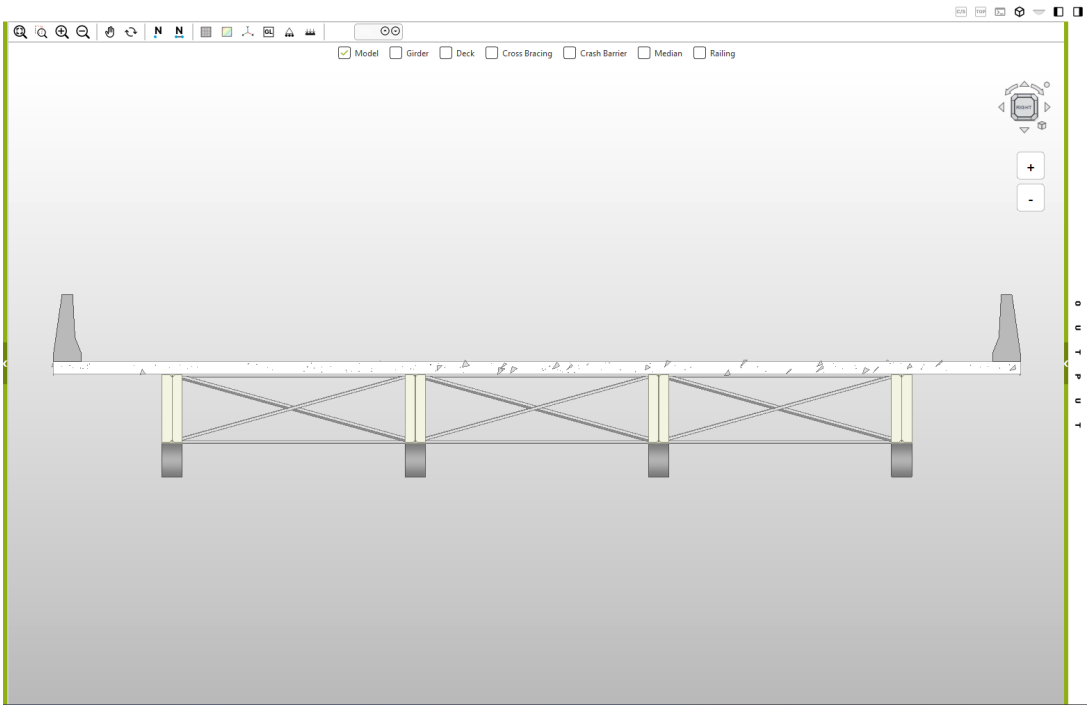

Figure 3.6: Interactive OpenCascade-based visualization environment integrated within OsdagBridge

3.8.3 Hover Interaction and Multi-Selection

The semester internship introduced synchronized hover interaction and multi-selection workflows for improved bridge inspection and subsystem-level visualization.

AIS objects were mapped to corresponding subsystem labels through centralized registration dictionaries, enabling synchronized hover-based component identification during viewer interaction.

Listing 3.7: Hover synchronization workflow

```
self.viewer.model_ais_objects[key] = ais_list


self.viewer.model_hover_labels[key] = label
```

The developed framework additionally supported simultaneous interaction with multiple bridge subsystems through synchronized selection management.

3.8.4 Zoom Controls and Viewport Synchronization

Integrated zoom-control systems were implemented directly within the visualization environment to improve accessibility of interactive navigation workflows.

Listing 3.8: Zoom-control integration workflow

```
self.zoom_in_btn.clicked.connect(
    lambda: self.display.ZoomFactor(1.1)
)


self.zoom_out_btn.clicked.connect(
    lambda: self.display.ZoomFactor(0.9)
)
```

Viewport-aware rendering operations and synchronized isometric initialization workflows were also incorporated to improve stability during viewer interaction.

3.8.5 Rendering Synchronization and Viewer Lifecycle Management

The semester internship also addressed synchronization challenges associated with repeated bridge regeneration and interactive rendering operations.

Controlled cleanup workflows and AIS re-registration pipelines were implemented before regeneration of bridge assemblies to maintain rendering stability and synchronization consistency.

Listing 3.9: Synchronized viewer cleanup workflow

```
display.EraseAll()


self.viewer.model_ais_objects = {}


self.viewer.model_hover_labels = {}
```

These workflows significantly improved reliability of interactive regeneration operations within the CAD environment.

3.9 Integrated CAD Export Workflow

The semester internship additionally integrated CAD export functionality directly within the synchronized visualization environment.

The implemented workflow enabled complete bridge assemblies to be exported from the interactive CAD environment while maintaining synchronization between generated geometry and consolidated OpenCascade assembly structures.

The export pipeline improved practical usability of the bridge engineering framework and supported interoperability with external CAD systems and engineering visualization environments.

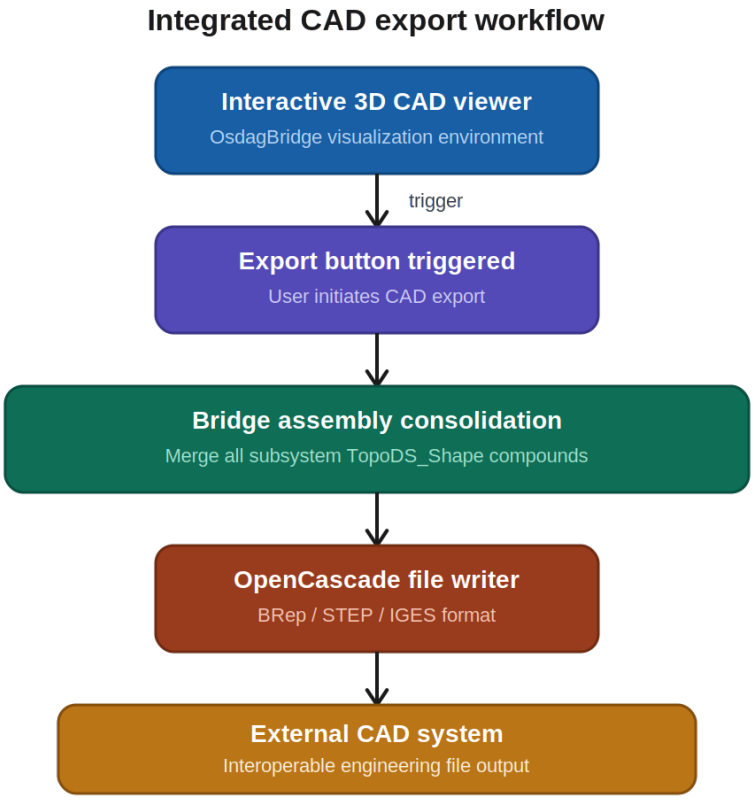

Figure 3.7: Integrated CAD export workflow within the synchronized visualization framework

3.10 Software Engineering Advantages

The semester internship significantly improved the maintainability, scalability, and extensibility of the OsdagBridge bridge engineering framework.

The implemented architecture introduced:

- modular subsystem separation,
- centralized synchronization workflows,
- reusable geometry utilities,
- scalable visualization pipelines,
- synchronized rendering management,
- extensible detailing architectures,

- maintainable OpenCascade integration workflows.

The builder-oriented architecture additionally simplified future integration of new bridge subsystems, analysis workflows, and visualization features within the OsdagBridge environment.

Overall, the semester internship transformed the earlier standalone bridge CAD prototype into a significantly more advanced synchronized engineering visualization framework capable of supporting interactive CAD workflows, advanced detailing systems, drafting-oriented visualization environments, centralized synchronization architectures, and scalable engineering extensibility within OsdagBridge.

Chapter 4

2D CAD Cross-Section Visualization System

4.1 Introduction

The semester-long internship involved the complete development and integration of an advanced two-dimensional CAD cross-section visualization framework within the OsdagBridge environment. The implemented system established a synchronized drafting-oriented visualization workflow capable of dynamically generating bridge sectional representations directly from user-defined bridge parameters.

The developed framework integrated parameter-driven geometry generation, viewport-aware rendering, interactive CAD navigation, dynamic dimension visualization, and synchronized graphical user interface workflows within a unified visualization environment.

The implementation focused on creating a maintainable and extensible CAD rendering architecture capable of supporting multiple bridge configurations, IRC-compliant safety systems, and synchronized interaction between bridge geometry generation and sectional visualization workflows.

The developed system established a fully interactive drafting-oriented CAD environment integrated with bridge geometry, rendering utilities, scaling workflows, annotation systems, and responsive user interaction mechanisms within OsdagBridge.

4.2 Objectives

The major objectives of the 2D CAD cross-section visualization development are summarized below:

- To develop a complete parameter-driven 2D CAD cross-section visualization framework for bridge systems within OsdagBridge.
- To establish synchronized rendering workflows integrated with bridge geometry generation and graphical user interface operations.
- To implement modular CAD rendering utilities for drafting-oriented visualization.
- To support dynamic dimension rendering and annotation workflows for bridge components.
- To implement viewport-aware scaling and fit-to-screen visualization operations.
- To integrate crash barriers, medians, railings, footpaths, and deck systems within a unified sectional rendering workflow.
- To implement interactive CAD navigation including zoom controls, hover interaction, and viewport synchronization.
- To establish maintainable and extensible visualization architecture for future bridge component integration.
- To improve drafting-oriented usability and real-time visualization capabilities within the OsdagBridge environment.

4.3 Architecture of the 2D CAD Visualization Framework

The developed visualization framework adopted a widget-based CAD rendering architecture using PySide6 and QPainter-based rendering workflows. Separate visualization widgets were implemented for sectional and top-view rendering operations, enabling organized separation of visualization responsibilities within the CAD environment.

The framework maintained centralized parameter-driven geometry synchronization between bridge inputs, rendering operations, and graphical user interface workflows. All bridge geometry and sectional visualization operations were dynamically generated from synchronized CAD state variables.

The visualization architecture also integrated viewport-aware rendering, drafting utilities, annotation systems, interaction workflows, and scaling mechanisms within the CAD rendering pipeline.

Figure 4.1 illustrates the architecture of the developed 2D CAD visualization framework.

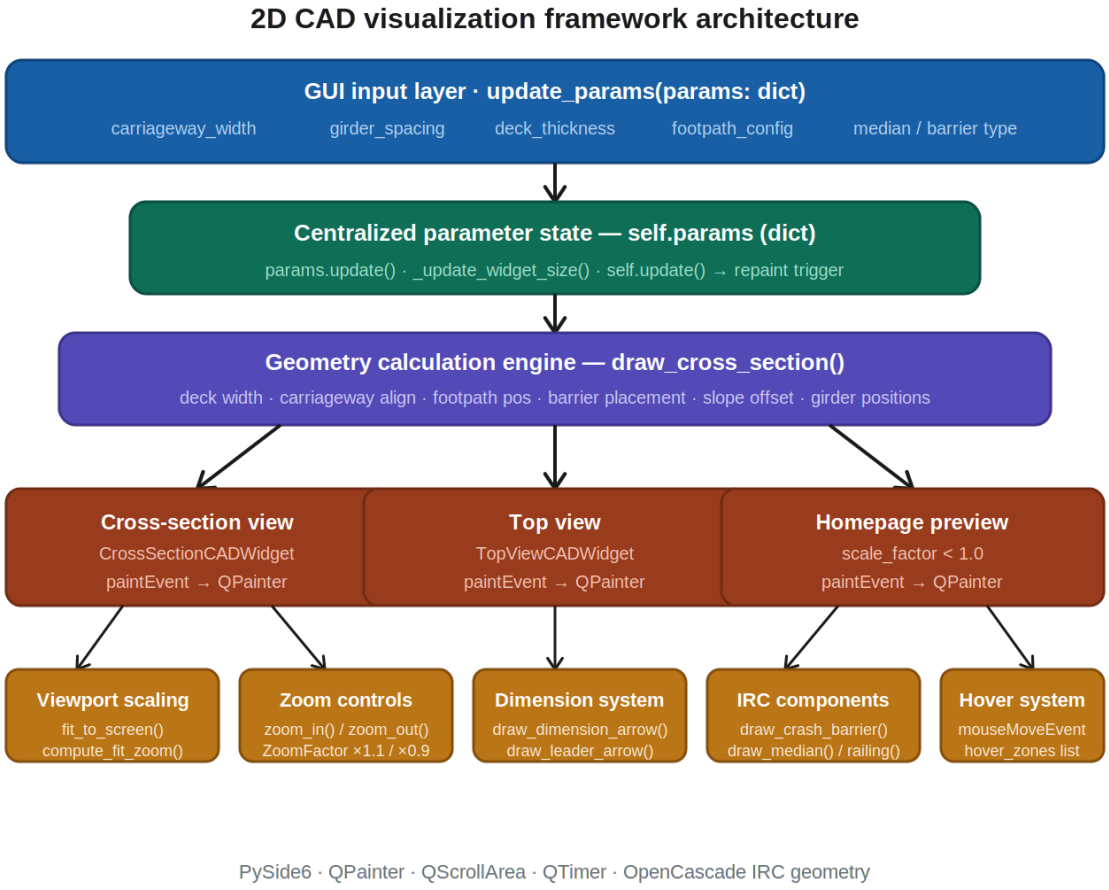

Figure 4.1: Architecture of the developed 2D CAD visualization framework within OsdagBridge

A centralized parameter-driven state management workflow was implemented to maintain synchronization between bridge geometry and rendering operations.

Listing 4.1: Centralized parameter-driven CAD state management

```
self.params = {
    'span_length': 35000,
    'num_girders': 4,
    'girder_spacing': 2750,
    'carriageway_width': 10500,
    'deck_thickness': 200,
    'footpath_width': 1500,
    'median_present': False,
    'median_width': 1200,
}
```

The centralized state management workflow enabled synchronized updates between graphical user interface operations and CAD rendering workflows while maintaining consistency across bridge geometry generation operations.

4.4 Cross-Section Rendering Workflow

The developed rendering workflow dynamically generated bridge cross-sections using parameter-driven geometric calculations integrated with drafting-oriented visualization operations.

The rendering pipeline computed bridge deck width, carriageway alignment, footpath positioning, crash barrier placement, railing offsets, median geometry, and bridge component alignment before generating the final sectional representation.

Centralized geometric calculations were implemented to maintain consistency across multiple bridge configurations while ensuring synchronized placement of deck systems, footpaths, railings, crash barriers, and medians.

The rendering workflow also integrated dynamic annotation systems, drafting utilities, scaling operations, and viewport synchronization within the sectional CAD environment.

Figure 4.2 illustrates the generated bridge cross-section visualization.

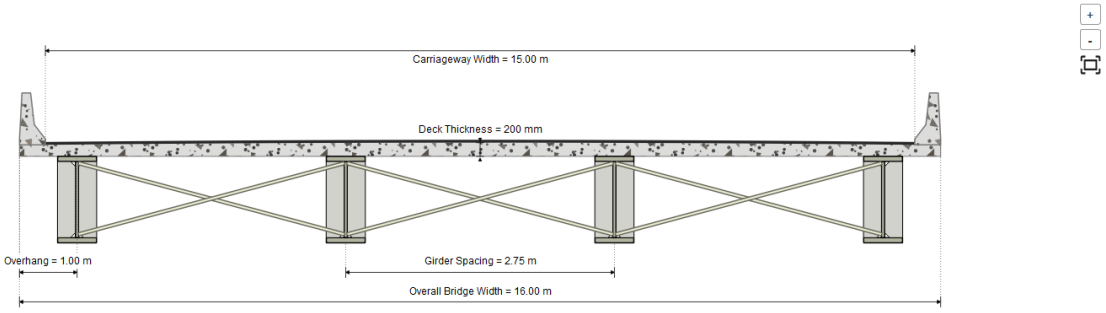

Figure 4.2: Bridge cross-section visualization generated using the developed 2D CAD rendering workflow

The implemented workflow established a fully synchronized and parameter-driven sectional CAD visualization environment within OsdagBridge.

4.5 Viewport Scaling and Fit-to-Screen Workflow

One of the major technical requirements of the CAD system involved maintaining visualization consistency across bridge configurations with varying deck widths, girder spacings, and sectional layouts.

To address this requirement, a viewport-aware fit-to-screen scaling workflow was implemented within the rendering environment. The system dynamically computed horizontal and vertical scaling factors based on viewport dimensions and bridge geometry extents.

Listing 4.2: Automatic fit-to-screen scaling workflow

```
scale_x = avail_w / total_deck_width
scale_y = avail_h / model_h
scale = min(scale_x, scale_y)


return target_scale / base_scale
```

The implemented workflow preserved aspect ratios while automatically adapting sectional visualization to different bridge sizes and viewport configurations. Dynamic centering operations were also integrated to maintain proper alignment during rendering updates and zoom operations.

The viewport-aware scaling system significantly improved visualization consistency, drafting readability, and user interaction within the CAD environment.

4.6 Interactive CAD Navigation and Zoom Controls

Interactive CAD navigation workflows were implemented within the visualization environment to support dynamic zooming, fit-to-screen operations, and viewport-aware interaction.

Zoom controls were integrated directly within the CAD widgets while maintaining synchronization with scroll areas and viewport positioning workflows.

Listing 4.3: Viewport-aware zoom control positioning

```
viewport = self.scroll_area.viewport()


x = viewport.width() - 50
y = margin


self.zoom_in_btn.move(x + 10, y)
self.zoom_out_btn.move(x + 10, y + 30)
```

The implemented workflow ensured that zoom controls remained fixed relative to the visible viewport during scrolling and resizing operations. Additional viewport synchronization logic maintained visualization centering during zoom updates and parameter modifications.

Figure 4.3 illustrates the integrated CAD navigation and zoom control workflow.

Figure 4.3: Interactive zoom controls and viewport-aware CAD navigation workflow

The implemented navigation workflow established a significantly improved interactive drafting-oriented CAD environment within OsdagBridge.

4.7 Dynamic Synchronization Between GUI Inputs and CAD Rendering

The developed framework established synchronized interaction between graphical user interface operations and CAD rendering workflows.

Bridge parameters modified through the graphical user interface were dynamically propagated to the visualization environment, enabling real-time sectional updates and synchronized rendering operations.

Listing 4.4: Dynamic synchronization of GUI parameters with CAD rendering

```
def update_params(self, params: dict):
    self.params.update(params)
    self._update_widget_size()
    self.update()
```

The synchronization workflow ensured that bridge geometry modifications were immediately reflected within the CAD environment while maintaining rendering consistency and responsive visualization behavior.

The implementation also integrated synchronized updates between sectional views, top-view rendering, homepage previews, and additional bridge input dialogs.

Figure 4.4 illustrates synchronization between graphical user interface operations and CAD visualization workflows.

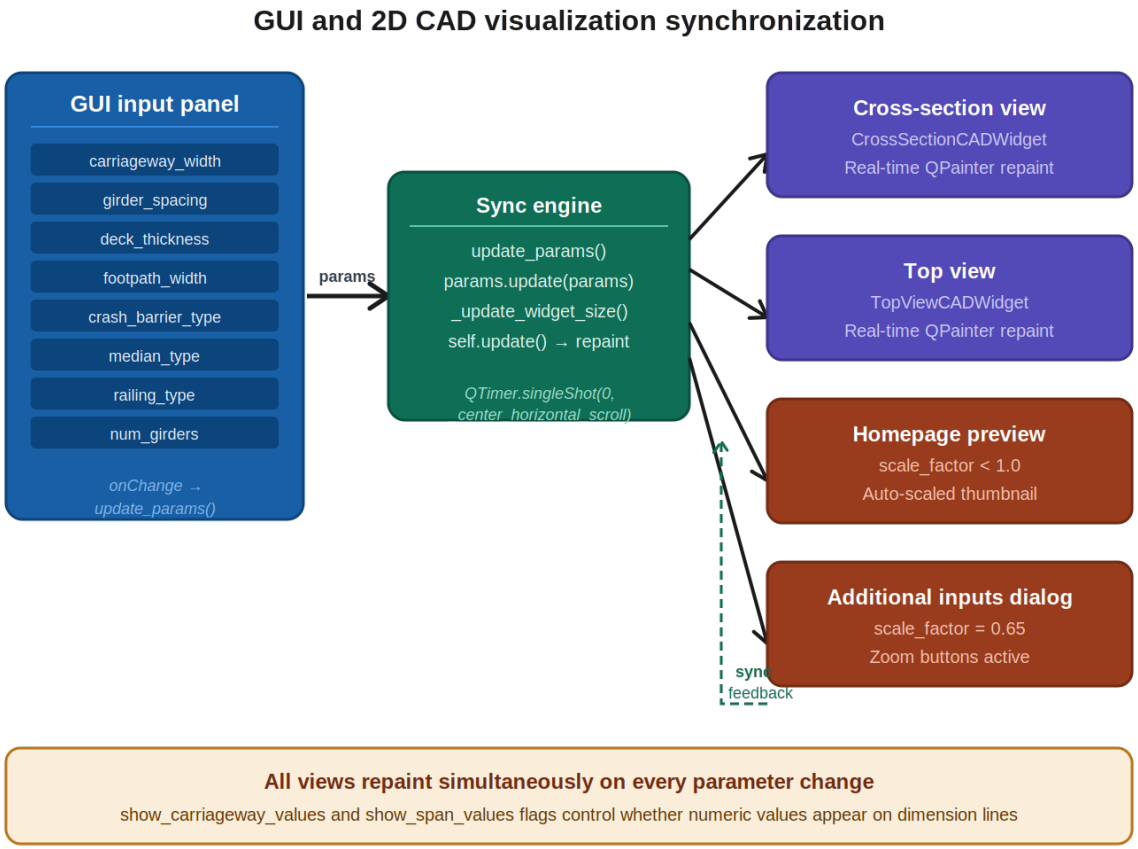

Figure 4.4: Synchronization between graphical user interface workflows and 2D CAD visualization

The implemented synchronization system established a responsive and dynamically updated CAD visualization environment within OsdagBridge.

4.8 Interactive Hover and Visualization Feedback System

The CAD visualization framework also incorporated interactive hover-based visualization workflows to improve interpretability and user interaction.

Mouse-tracking systems, hover regions, dynamic highlighting, and interactive compo-

nent feedback workflows were implemented within the rendering environment.

The hover interaction system enabled bridge components to be dynamically identified during user interaction, thereby improving sectional interpretation and visualization usability.

These interactive visualization features established a more responsive drafting-oriented CAD environment while improving user interaction within OsdagBridge.

4.9 Integration of IRC Geometry Abstractions

The developed framework integrated modular geometry abstractions for IRC-compliant bridge safety systems including crash barriers, railings, and medians.

Separate geometry workflows were implemented for different safety system configurations, enabling synchronized generation and visualization of multiple IRC-compliant bridge components.

The rendering system supported different crash barrier systems including RCC barriers and W-beam configurations while also supporting multiple median arrangements and railing systems.

Figure 4.5 illustrates sectional visualization of IRC-compliant safety systems integrated within the CAD workflow.

Figure 4.5: 2D CAD visualization of crash barriers, medians, and railing systems

The modular geometry abstraction workflow improved maintainability and enabled

future IRC-compliant bridge components to be integrated efficiently within OsdagBridge.

4.10 Challenges and Solutions

4.10.1 Challenge 1: Maintaining Synchronized Rendering Workflows

Maintaining deterministic synchronization between graphical user interface operations and CAD repaint workflows introduced significant challenges related to rendering consistency and geometry updates.

This issue was addressed through centralized parameter synchronization and rendering refresh operations integrated within the visualization pipeline.

4.10.2 Challenge 2: Viewport Scaling and Visualization Consistency

Different bridge configurations introduced challenges related to aspect-ratio preservation, viewport fitting, centering operations, and visualization consistency.

Viewport-aware fit-to-screen scaling workflows and dynamic centering operations were implemented to ensure consistent rendering behavior across multiple bridge layouts.

4.10.3 Challenge 3: Component Placement and Geometric Alignment

Bridge components including crash barriers, railings, medians, footpaths, and deck systems required synchronized placement within the sectional rendering environment.

Centralized geometric calculations and parameter-driven alignment workflows were implemented to maintain consistency across different bridge configurations.

4.10.4 Challenge 4: Interactive Visualization and CAD Navigation

Implementing smooth zoom handling, viewport synchronization, and interactive CAD navigation workflows required coordination between rendering operations, scroll areas,

and viewport positioning systems.

This issue was addressed through viewport-aware navigation logic and synchronized zoom-control workflows integrated directly within the CAD widgets.

4.11 Results

The semester-long internship successfully established a fully integrated and parameter-driven 2D CAD cross-section visualization framework within OsdagBridge.

The developed system supported synchronized sectional rendering, dynamic dimension visualization, viewport-aware scaling, interactive CAD navigation, hover-based visualization feedback, and real-time synchronization between bridge geometry and graphical user interface workflows.

The implementation also established reusable drafting utilities, modular geometry abstraction workflows, responsive rendering operations, and synchronized bridge component visualization systems integrated within the OsdagBridge environment.

Overall, the developed framework established a complete interactive drafting-oriented CAD visualization system integrated with bridge geometry generation, synchronized rendering workflows, IRC-compliant component visualization, and advanced user interaction mechanisms within OsdagBridge.

Chapter 5

Conclusions

5.1 Overview of Work Accomplished

The semester-long internship successfully established an advanced bridge CAD visualization and modeling framework integrated within OsdagBridge. The work extended the previously developed bridge modeling workflows into a significantly more synchronized, interactive, and engineering-oriented system capable of supporting advanced bridge visualization and drafting operations.

The internship focused on the development of advanced parametric 3D CAD workflows, synchronized 2D CAD cross-section visualization systems, modular rendering architectures, and improved interaction between graphical user interface operations and CAD generation environments.

Major contributions included the implementation of advanced bridge detailing systems, synchronized visualization workflows, interactive CAD environments, dynamic rendering operations, and drafting-oriented bridge visualization capabilities integrated within the OsdagBridge ecosystem.

The internship also established deeper integration between bridge geometry generation, visualization systems, parameter synchronization workflows, and application-level interaction mechanisms within OsdagBridge.

5.2 Advanced Parametric 3D CAD Development

A major portion of the semester internship focused on the development and extension of the advanced parametric 3D CAD bridge generation framework.

The implemented workflows established a synchronized and configurable bridge modeling environment capable of generating multiple bridge components dynamically from user-defined engineering parameters.

Major developments completed during the internship included:

- Advanced stiffener systems including configurable end stiffeners, longitudinal stiffeners, and outstand stiffener workflows.
- Parametric shear stud generation integrated with synchronized placement workflows within plate girder assemblies.
- IRC-compliant crash barrier systems including single and double W-beam configurations.
- Multiple median configurations integrated within bridge geometry generation workflows.
- Parametric railing systems with synchronized placement and skew-aware geometry alignment.
- Enhanced cross-bracing and end diaphragm systems with configurable section selection workflows.
- Interactive 3D CAD visualization features including hover labels, component highlighting, multi-component selection, zoom controls, and synchronized rendering operations.
- CAD export workflows integrated within the visualization environment.
- Improved synchronization between bridge geometry generation and graphical user interface operations.

The developed framework significantly improved configurability, engineering realism, visualization consistency, and maintainability within the bridge CAD generation environment.

5.3 Development of the 2D CAD Cross-Section Visualization System

One of the major contributions of the semester internship was the development and integration of a complete interactive 2D CAD cross-section visualization framework within OsdagBridge.

The implemented system established a synchronized drafting-oriented visualization environment integrated with bridge geometry generation, rendering utilities, viewport-aware scaling systems, and graphical user interface workflows.

The developed framework included:

- Parameter-driven sectional CAD rendering workflows.
- Dynamic synchronization between graphical user interface inputs and CAD visualization.
- Automatic viewport scaling and fit-to-screen rendering operations.
- Interactive CAD navigation including zoom controls and viewport synchronization.
- Dynamic dimension rendering and drafting annotation systems.
- Hover-based interaction and visualization feedback mechanisms.
- Integration of crash barriers, medians, railings, footpaths, and deck systems within the sectional CAD environment.
- Reusable drafting utilities and modular rendering workflows.
- Improved rendering consistency and drafting-oriented visualization behavior.

The developed 2D CAD framework established a responsive and synchronized visualization environment capable of dynamically generating bridge sectional representations from engineering parameters within OsdagBridge.

5.4 Software Engineering and Architectural Improvements

The semester internship also focused extensively on software engineering improvements and architectural organization within the bridge CAD framework.

Significant effort was devoted toward improving modularity, synchronization workflows, maintainability, and integration between visualization systems and bridge generation operations.

Major software engineering contributions included:

- Development of modular rendering workflows for bridge visualization systems.
- Centralized parameter synchronization between CAD environments and graphical user interface operations.
- Improved organization of bridge geometry generation workflows.
- Development of reusable drafting and rendering utility systems.
- Synchronized interaction between 2D and 3D visualization environments.
- Viewport-aware rendering and scaling architectures.
- Enhanced maintainability and extensibility of the CAD framework.
- Improved integration between bridge generation systems and application-level workflows.

These architectural improvements established a significantly stronger engineering foundation for future development and extension of bridge visualization systems within OsdagBridge.

5.5 Technical Skills Developed

The internship provided extensive exposure to advanced CAD visualization systems, engineering software development workflows, and large-scale parametric modeling environments.

Major technical skills developed during the internship include:

- Parametric CAD modeling using pythonOCC and OpenCascade.
- Development of synchronized 2D and 3D CAD visualization systems.
- Advanced geometric modeling and coordinate transformation workflows.
- Interactive rendering and CAD visualization techniques.
- GUI development and integration using PySide6.
- Modular software architecture and synchronization workflows.
- Engineering-oriented drafting and annotation systems.
- Viewport-aware rendering and scaling operations.
- Debugging, rendering optimization, and workflow synchronization.
- Large-scale engineering software integration practices.

The internship also improved problem-solving abilities, engineering software design practices, technical documentation skills, and collaborative software development experience within a large engineering software environment.

5.6 Future Scope

The developed framework establishes a strong architectural and visualization foundation for future bridge CAD and engineering workflow development within OsdagBridge.

Possible future extensions include:

- Advanced bridge segmentation and variable-depth girder modeling workflows.
- Automated engineering drawing and drafting generation systems.
- Integration of structural analysis workflows with bridge visualization environments.
- Advanced material rendering and realistic visualization systems.
- Expansion of IRC-compliant bridge component libraries.

- Performance optimization for large-scale bridge assemblies.
- Automated engineering checking and bridge design validation systems.
- Extended bridge detailing and fabrication-oriented visualization workflows.
- Cloud-based collaborative bridge visualization systems.

The modular and synchronized architecture developed during the internship provides sufficient extensibility to support future integration of advanced bridge engineering workflows within OsdagBridge.

5.7 Conclusion

The semester-long internship successfully established a significantly advanced bridge CAD visualization and modeling framework integrated within OsdagBridge.

The work contributed toward the development of synchronized parametric bridge generation workflows, advanced 3D CAD systems, interactive 2D sectional visualization environments, drafting-oriented rendering utilities, and responsive graphical user interface integration mechanisms.

The internship also established improved synchronization between engineering parameters, bridge geometry generation, CAD visualization systems, and application-level interaction workflows.

The developed framework enhanced engineering realism, configurability, visualization consistency, maintainability, and drafting-oriented usability within the OsdagBridge environment while establishing a scalable architectural foundation for future bridge engineering development.

Overall, the internship provided valuable experience in engineering software development, parametric CAD modeling, visualization system design, rendering architecture development, and large-scale software integration practices while contributing to the continued development of OsdagBridge at FOSSEE, IIT Bombay.

Chapter A

Appendix

A.1 Source Code & Contributions

The complete source code and commit history for the work done during this internship is available on GitHub:

- **Repository:** `https://github.com/Faizan-9077/OsdagBridge`
- **Branch / Fork:** `dev`

A.2 Work Reports

Internship Work Report

Name	Faizan Khan
Project	Osdag
Internship	FOSSEE Semester Long Internship

DATE	DAY	TASK	HOURS WORKED
01-Feb-26	Sunday	Studied OsdagBridge codebase and set up semester internship development environment	4
02-Feb-26	Monday	Reviewed winter internship bridge CAD framework and identified areas for extension	5
03-Feb-26	Tuesday	Explored pythonOCC and OpenCascade visualization workflows for advanced rendering	5
04-Feb-26	Wednesday	Studied AIS-based rendering architecture and interactive context workflows in OCC	5
05-Feb-26	Thursday	Planned modular CAD generation pipeline and builder-based architecture design	4
06-Feb-26	Friday	Reviewed IS and IRC standards relevant to crash barriers, railings, and bridge detailing	4
07-Feb-26	Saturday	Set up development branches and reviewed existing plate girder CAD generation code	4
08-Feb-26	Sunday	Added 3D CAD viewer window using CustomViewer3d and QMainWindow-based architecture	6
09-Feb-26	Monday	Introduced class-based PlateGirder CAD generator and cross bracing builder	6
10-Feb-26	Tuesday	Integrated cross bracing builder into plate girder CAD and resolved placement issues	5
11-Feb-26	Wednesday	Implemented checkbox-based component isolation in 3D CAD viewer and hover labels	6
12-Feb-26	Thursday	Worked on rolled beam alignment and fixed end diaphragm placement behind girder	6
13-Feb-26	Friday	Worked on report making and documentation of winter internship work	4
14-Feb-26	Saturday	Continued report making and documentation	5
15-Feb-26	Sunday	Report writing and implemented multi-select components feature in 3D CAD viewer	6
16-Feb-26	Monday	Added different section options for members in end diaphragm and cross bracings	7

17-Feb-26	Tuesday	Added option of selecting end bearing and toggling intermediate stiffeners inclusion	7
18-Feb-26	Wednesday	Added outstand options for intermediate and end stiffeners in plate girder	6
19-Feb-26	Thursday	Added longitudinal stiffeners with configurable outstand option	7
20-Feb-26	Friday	Added shear studs in the bridge model with configurable dimensions and pitch spacing	7
21-Feb-26	Saturday	Refactored code and fixed issues in shear stud placement and geometry	6
22-Feb-26	Sunday	Started implementing girder segmentation with GirderSegmentDTO representation	6
23-Feb-26	Monday	Adjusted cross bracings for different girder segments with depth-aware logic	6
24-Feb-26	Tuesday	Added skew angle in each bridge component and aligned geometry accordingly	5
25-Feb-26	Wednesday	Adjusted stiffeners for different segments and synchronized placement	6
26-Feb-26	Thursday	Fixed issues in component placement and resolved geometry conflicts	6
27-Feb-26	Friday	Understanding the code base of 2D CAD cross-section widget and rendering pipeline	5
28-Feb-26	Saturday	Made different crash barrier types per IRC 5 spec and rendered on both CAD views	7
01-Mar-26	Sunday	Added different types of median (raised kerb, RCC barrier, metallic) to CAD	7
02-Mar-26	Monday	Added steel railing with concrete base and IRC 5 compliant geometry	7
03-Mar-26	Tuesday	Created pull request and resolved merge conflicts in codebase	5
04-Mar-26	Wednesday	Understood code structure for dimension analysis in 2D CAD widget	5
05-Mar-26	Thursday	Worked on synchronizing both 2D CAD views (cross-section and top view)	6
06-Mar-26	Friday	Synchronized both CAD views: homepage to additional inputs dialog and vice-versa	7
07-Mar-26	Saturday	Implemented always-visible hover labels from application startup	6

08-Mar-26	Sunday	Configured dimensions to display when carriageway width or span length is changed	6
09-Mar-26	Monday	Fixed footpath width calculation and removed all hardcoded dimension values	6
10-Mar-26	Tuesday	Fixed deck width calculation and corrected footpath width labeling	6
11-Mar-26	Wednesday	Adjusted colors for CAD components: girder, stiffeners, and cross bracing	5
12-Mar-26	Thursday	Fixed default CAD view size and centering for standard startup configuration	5
13-Mar-26	Friday	Worked on fixing the CAD size inside the additional inputs dialog preview	6
14-Mar-26	Saturday	Implemented viewport-aware zoom control positioning fixed to scroll area viewport	6
15-Mar-26	Sunday	Added fit-to-screen functionality and improved scaling logic for cross-section CAD	6
16-Mar-26	Monday	Synchronized initial CAD state and ensured safe application startup sequence	5
17-Mar-26	Tuesday	Implemented conditional dimension visibility and default labeling system	6
18-Mar-26	Wednesday	Fixed race condition when closing AdditionalInputs dialog with local dialog reference	5
19-Mar-26	Thursday	Initialized additional inputs CAD preview from homepage CAD state parameters	6
20-Mar-26	Friday	Fixed railing width unit handling and improved CAD scaling behavior	6
21-Mar-26	Saturday	Triggered CAD update on Save button click and updated crash barrier geometry	6
22-Mar-26	Sunday	Enhanced crash barrier geometry, CAD rendering, and UI responsiveness	6
23-Mar-26	Monday	Improved CAD rendering and UI synchronization across all views	5
24-Mar-26	Tuesday	Improved CAD preview scaling, centering, and dimension layout in cross-section and top view	6
25-Mar-26	Wednesday	Fixed deck width and footpath layout by including railing width in calculations	6
26-Mar-26	Thursday	Disabled railing and median tabs based on input dock selections dynamically	5

27-Mar-26	Friday	Fixed crash barrier CAD rendering and synchronized with IRC 5 specification	6
28-Mar-26	Saturday	Added different median CAD rendering and synced with IRC 5 geometry spec	6
29-Mar-26	Sunday	Fixed database integration issue and resolved sqlite path configuration	4
30-Mar-26	Monday	Added crash barrier types: Single W-beam and Double W-beam per IRC 5	7
31-Mar-26	Tuesday	Added IRC5-based crash barrier geometry updates and median geometry refinements	6
01-Apr-26	Wednesday	Added double W-beam median geometry and updated raised kerb median rendering	6
02-Apr-26	Thursday	Corrected placement of post channel in metallic type crash barrier geometry	5
03-Apr-26	Friday	Added different colors for flange and web of the girder for visual distinction	5
04-Apr-26	Saturday	Made W-beam hollow and improved the overall visual appearance in 3D viewer	6
05-Apr-26	Sunday	Returned W-beam as a separate shape and added distinct color rendering	5
06-Apr-26	Monday	Added girder supports with transparency and integrated into CAD viewer display	7
07-Apr-26	Tuesday	Added steel railing per IRC 5 conditions and updated concrete base geometry	6
08-Apr-26	Wednesday	Fixed metallic median post distance and resolved position offset issues	5
09-Apr-26	Thursday	Added skew angle support in each bridge component for skew-aware geometry	6
10-Apr-26	Friday	Added some offset to fix the metallic type median post placement	5
11-Apr-26	Saturday	Added concrete base and circular post for steel type railing assembly	6
12-Apr-26	Sunday	Modified circular post to rectangle and fixed metallic median post position	5
13-Apr-26	Monday	Fixed double angle section type cross bracings and refactored bracing code	6
14-Apr-26	Tuesday	Aligned railing, crash barrier, and median geometry with bridge skew angle	6

15-Apr-26	Wednesday	Added end stiffeners to plate girder assembly and validated geometry	6
16-Apr-26	Thursday	Fixed minor issues in plate girder geometry and cross bracing alignment	5
17-Apr-26	Friday	Aligned cross bracing members with plate girder geometry using depth-aware logic	6
18-Apr-26	Saturday	Added custom type drawing for crash barrier, median, and railing with dashed borders	7
19-Apr-26	Sunday	Added curved wearing course, cleaned deck-footpath rendering, and fixed cross-section bugs	7
20-Apr-26	Monday	Created DTO for 3D CAD bridge design parameters and integrated with generation pipeline	7
21-Apr-26	Tuesday	Added shear stud geometry, placement logic, and 3D display for plate girder	7
22-Apr-26	Wednesday	Added per-girder segment configuration and adapted bracing geometry for segments	7
23-Apr-26	Thursday	Added parametric girder segmentation support with GirderSegmentDTO integration	6
24-Apr-26	Friday	Integrated girder segmentation and shear stud parameters with DTO and CAD updates	7
25-Apr-26	Saturday	Fixed footpath rendering and improved cross-section visual quality	6
26-Apr-26	Sunday	Cleaned unused functions, improved rendering, and fixed cross-section issues	6
27-Apr-26	Monday	Updated wearing course inputs and synchronized with CAD rendering pipeline	6
28-Apr-26	Tuesday	Removed dead code and centralized scaling logic across CAD widgets	6
29-Apr-26	Wednesday	Reviewed and tested full bridge assembly with all detailing systems active	5
30-Apr-26	Thursday	Performed integration testing of 2D and 3D CAD views with synchronized parameters	5
01-May-26	Friday	Began writing semester internship report and documenting implemented systems	5
02-May-26	Saturday	Documented modular CAD generation pipeline and DTO-based synchronization architecture	5
03-May-26	Sunday	Wrote chapter on advanced stiffener and shear stud systems for internship report	5

04-May-26	Monday	Documented cross-bracing and end diaphragm generation workflows in report	5
05-May-26	Tuesday	Wrote chapter on interactive 3D CAD visualization and AIS rendering architecture	5
06-May-26	Wednesday	Documented 2D CAD cross-section visualization framework and viewport scaling	5
07-May-26	Thursday	Wrote documentation for zoom controls, hover system, and UI synchronization workflows	5
08-May-26	Friday	Prepared figures and diagrams for internship report chapters	5
09-May-26	Saturday	Reviewed and refined all report chapters with mentor feedback incorporated	5
10-May-26	Sunday	Final testing of complete OsdagBridge CAD framework before report submission	5
11-May-26	Monday	Finalized conclusion and results chapter of semester internship report	5
12-May-26	Tuesday	Polished report formatting, figures, and table of contents	4
13-May-26	Wednesday	Reviewed complete internship report and prepared for final submission	4
14-May-26	Thursday	Added Save 3D CAD functionality for exporting bridge assembly to external CAD systems	7
15-May-26	Friday	Final review, cleanup, and submission of semester internship work report	4
		TOTAL HOURS WORKED	**588**

Bibliography

[1] Siddhartha Ghosh, Danish Ansari, Ajmal Babu Mahasrankintakam, Dharma Teja Nuli, Reshma Konjari, M. Swathi, and Subhrajit Dutta. Osdag: A Software for Structural Steel Design Using IS 800:2007. In Sondipon Adhikari, Anjan Dutta, and Satyabrata Choudhury, editors, *Advances in Structural Technologies*, volume 81 of *Lecture Notes in Civil Engineering*, pages 219–231, Singapore, 2021. Springer Singapore.

[2] FOSSEE Project. FOSSEE News - January 2018, vol 1 issue 3. Accessed: 2024-12-05.

[3] FOSSEE Project. Osdag website. Accessed: 2024-12-05.