**Kathmandu University**

**Department of Computer Science and Engineering**

**Dhulikhel, Kavre**

**A Project Report**

**on**

# "FRAEX: A Frame Simulating App | Using Raylib and C++"

**[Code No: COMP 202]**

**(For partial fulfillment of Year II / Semester I in Computer Engineering)**

**Submitted by**

**Sworup Jangam(037975-24)**

**Submitted to:**

**Mr. Sagar Acharya**

**Department of Computer Science and Engineering**

**Submission Date: 26th February, 2026**

# Bona fide Certificate

**This project work on**

**“FRAEX: A Frame Simulating App | Using Raylib and C++”**

**is the bona fide work of**

**“Sworup Jangam”**

**who carried out the project work under my supervision.**

---

**Mr. Sagar Acharya**
**Lecturer**
**Department of Computer Science and Engineering**

# Acknowledgement

I would like to express my sincere gratitude to my project supervisor for the valuable guidance, feedback, and encouragement they have provided during the development of the project. I think this has been very helpful in understanding the practical aspects of the project.

I would like to extend my gratitude to the faculty members of the Department of Engineering for providing the learning environment for me to complete this project successfully. The concepts learned during the class have been very useful for the development of this project.

Lastly, I would like to extend my gratitude for the learning opportunities provided during the development of the project, which have significantly enhanced my learning about Data Structures, problem-solving, and system development.

# Abstract

The processing of frames is an essential part of various multimedia systems, and its application is seen in video playback and animation rendering. The processing of video frames involves complex decoding algorithms and requires a high amount of computational resources. Therefore, in order to simplify the demonstration of frame processing concepts, this project aims to present a frame simulation system developed using a C++ programming language and a Raylib graphics library.

The proposed system aims to simulate a frame-by-frame visualization system without involving complex video decoding algorithms. The frame data is logically generated and displayed sequentially at regular intervals of time. The system is mainly aimed at demonstrating various data structure traversal techniques, especially sequential indexing of frame data. The frame simulation model is a simple system that aims to demonstrate a frame processing system and its application in various multimedia systems. The system is implemented using a simple array-based frame storage system and a timer-based frame update system. The proposed system is mainly aimed at demonstrating and educating users about various concepts and ideas in frame processing systems. The system is a simple implementation of a frame simulation system and can be extended in future to include various advanced multimedia processing system features.

The proposed system has successfully demonstrated a simple frame simulation system that can be extended in future to include various advanced multimedia processing system features.



Keywords**:** Frame Simulation, Raylib Graphics Library, *C++*

# TABLE OF CONTENTS


| | |
|---|---|
| Acknowledgement | ii |
| Abstract | iii |
| List of Figures | iv |
| Acronyms/Abbreviations | v |
| Chapter 1 Introduction | 6 |
| 1.1 Background | 6 |
| 1.2 Objectives | 7 |
| 1.3 Motivation and Significance | 7 |
| Chapter 2 Literature Review | 8 |
| Chapter 3 Design and Implementation | 9 |
| 3.1 Data Structure Design | 9 |
| 3.2 Implementation Details | 11 |
| 3.3 GUI Overview | 12 |
| 3.4 System Diagrams | 13 |
| Chapter 4 Discussion on the achievements | 14 |
| 4.1 Features | 15 |
| Chapter 5 Conclusion and Recommendation | 16 |
| 5.1 Limitations | 17 |
| 5.2 Future Enhancement | 17 |
| References | 18 |

# List of Figures

Fig 3.1: GUI of the app

Fig 3.2: Use Case Diagram

Fig 3.3: Workflow Diagram

# Acronyms/Abbreviations

| | |
|---|---|
| FPS | Frames Per Second |
| GUI | Graphical User Interface |
| RAM | Ramdom Access Memory |
| C++ | C Plus Plus Programming Language |
| Raylib | Raylib Graphics Library |

# Chapter 1 Introduction

With the advent of multimedia technology, video and animation processing has become a significant component of modern computer systems. Video processing often includes handling large volumes of data, especially during frame-by-frame processing. However, for implementing video frame extraction, complex multimedia decoding techniques have to be used. For educational purposes, simulation techniques have been used for this project.

This project aims to develop a frame simulation system using the C++ programming language and the Raylib library for graphics rendering. It simulates the processing of video frames without using video decoding techniques. The main aim of this project is to develop a simple system for simulating the processing of video frames for educational purposes.

## 1.1 Background

Multimedia systems are commonly employed in modern computing systems for video playback, animation, gaming, and visualization. The process of video processing involves handling continuous streams of frame data, where each frame represents a single image in a video. The process of frame extraction in a video involves decoding compressed multimedia systems. For educational and prototyping purposes, a simulation model is commonly employed in frame processing. The process of frame simulation is a simpler approach for understanding frame processing without employing complex video decoding algorithms.

The objective of this project is to demonstrate a simple idea of frame visualization using a C++ programming language and a graphics library called Raylib. The system is designed to demonstrate frame visualization in a sequential manner using basic

concepts of data structures. The background motivation of this project is to understand the process of animation and frame processing in a simple environment for mini-project implementation.

## 1.2 Objectives

The primary objectives of this project are:

- To design a frame-based simulation system.
- To demonstrate sequential frame visualization.
- To implement basic data traversal concepts.
- To develop a lightweight graphical animation prototype.
- To utilize C++ programming for system logic.
- To use Raylib library for rendering frames.

## 1.3 Motivation and Significance

The motivation of this project also came from the course COMP 202, where we learned about different data structures and algorithms. The core idea is to bring those concepts to real life and learn more about them.

The motivation behind this project is to create a frame simulation app as a starter to see what can be done for further ideas like frame extractor which can be used to extract frames from a video provided by us and edit the frames as per the requirement of the user.

# Chapter 2 Literature Review

Frame processing and animation rendering are the primary concepts of multimedia computing and computer graphics. Several research papers have been written on efficient video processing and animation rendering techniques. In the early days of multimedia computing, complex video processing was done by utilizing hardware-level processing and complex software rendering techniques. With the advent of programming environments and libraries for computer graphics, simple rendering techniques have been employed for educational and prototype-level applications.

In the latest animation rendering techniques, frame sequencing and time-based rendering are employed to simulate animation. Frame buffering techniques are employed to enhance smooth video playback by storing multiple frames in memory and rendering them sequentially. Data structures like arrays and queues are employed for efficient frame sequencing and rendering. Several simple graphical libraries have been developed to simplify animation rendering and visualization. This has enabled developers to concentrate more on the logical implementation of the application rather than handling the complexities of multimedia and hardware-level processing. In educational software development, several simulation-based models are employed to simplify complex ideas and concepts. Frame simulation techniques are employed to simplify complex ideas and concepts for better educational outcomes. Frame simulation techniques are employed to simplify complex ideas and concepts for better educational outcomes.

In this project, the above ideas are employed to implement a simple frame simulation technique using the C++ programming language and the Raylib library. The proposed system differs from the conventional video processing and rendering techniques by employing simple frame visualization logic for educational purposes.

# Chapter 3 Design and Implementation

The project is designed to simulate how frames can be generated and manupulated. It consists of two main components: C++ for the programming section and Raylib for the GUI of the app.

## 3.1 Data Structure Design

The project mainly utilizes three fundamental data structures:

**1. Linked List (Frame Storage)**

Frames are stored using a singly linked list structure.

The linked list node contains:

- Frame number
- Timestamp (simulated)
- Pointer to next frame

Linked list is used because it allows dynamic frame insertion and deletion without requiring continuous memory allocation.

Operations supported:

- Add Frame
- Delete Frame
- Display Frame List
- Sorted Frame Restoration during undo operation

The frame insertion operation adds new frames at the end of the list.

**2. Stack (Undo Delete Operation)**

A stack data structure is used to implement the undo deletion functionality.

The stack follows Last In First Out (LIFO) principle.

When a frame is deleted:

- The frame data is pushed into the stack.

During undo operation:

- The most recently deleted frame is popped and restored into the linked list.

Stack operations implemented:

- Push operation
- Pop operation
- Stack empty check

**3. Queue (Frame Processing Module)**

Queue data structure is used in the frame processing function.

The queue follows First In First Out (FIFO) principle.

Frame processing workflow:

1. Traverse linked list and enqueue all frames
2. Dequeue frames sequentially
3. Display processing message

Queue operations implemented:

- Enqueue
- Dequeue
- Front and rear pointer management

## 3.2 Implementation Details

The system was implemented using the C++ programming language.

**Frame Addition**

When the user clicks the "Add Frame" button, a new frame node is dynamically created and appended to the linked list. Frame numbering is automatically generated using a global counter variable.

**Frame Deletion**

Frame deletion is performed based on frame number. The system searches the linked list and removes the matching frame node.

Deleted frames are stored in the stack structure to support undo functionality.

**Undo Operation**

The undo feature restores the most recently deleted frame by popping data from the stack and reinserting it into the linked list in sorted order.

**Frame Processing**

The process module transfers frame data from linked list to queue structure and simulates frame processing sequentially.

**Tools and Technologies Used**

- Programming Language: C++

- Graphics Library: Raylib
- Data Structures Used:
    - Singly Linked List
    - Stack
    - Queue.

## 3.3 GUI Overview

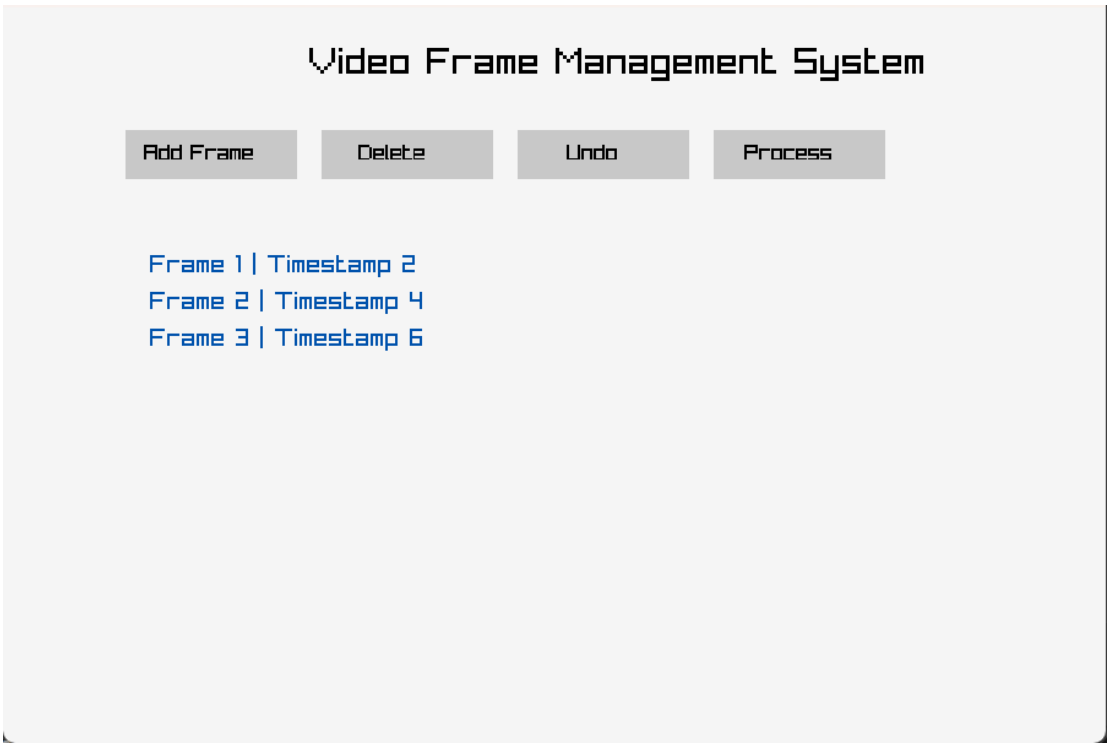


Fig 3.1: GUI of the app

## 3.4 System Diagrams

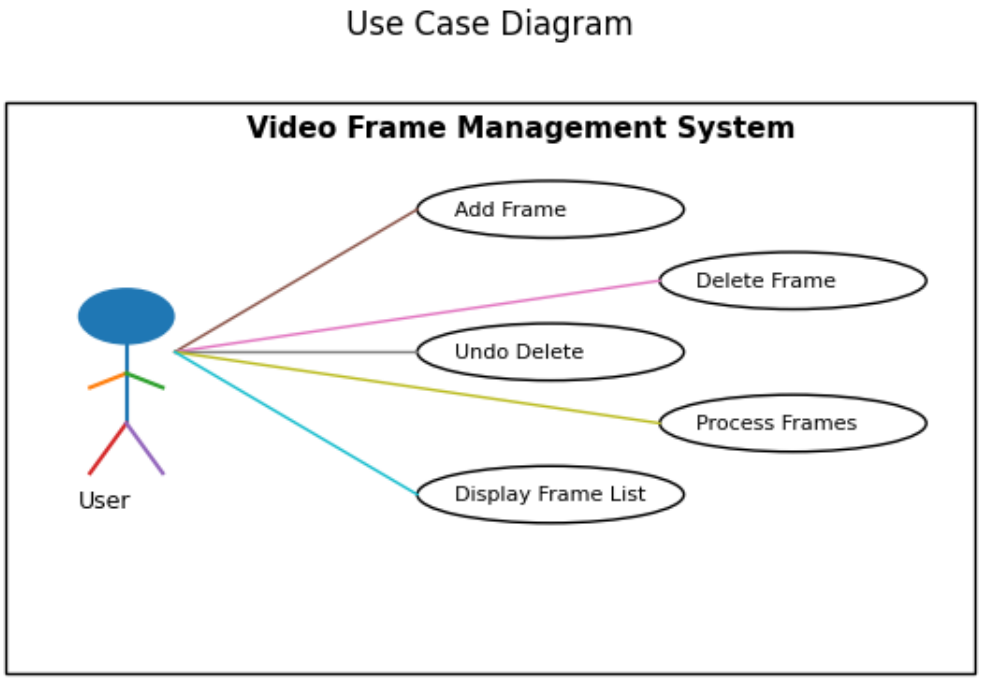


Fig 3.2: Use Case Diagram


Fig 3.3: Workflow Diagram

# Chapter 4 Discussion on the achievements

The Video Frame Management System successfully demonstrates the implementation of basic data structures in a graphical simulation environment using the C++ programming language with the Raylib library. It successfully fulfills its main objective of providing a simple model for a frame simulation environment without using complex multimedia decoding techniques. One of the greatest achievements of this project is the incorporation of multiple data structures such as a singly linked list, a stack, and a queue to manage the frames efficiently. The structure of a linked list allows dynamic insertion and deletion of frames. A stack is used to implement the undo feature by storing the recently deleted frames. A queue is used to simulate sequential processing of frames.

A successful implementation of a graphical interface is also provided through the use of the Raylib library. This allows users to interact with the Video Frame Management System through buttons such as "Add Frame," "Delete Frame," "Undo," and "Process Frames." It provides a real-time simulation environment for the visualization of frame management operations. This project also demonstrates a basic understanding of event-driven programming concepts, in which the input provided by a user is used to execute operations on the incorporated data structures. It provides a simulation model that can be used to understand how frames can be stored, retrieved, and processed sequentially.

This project also provides a better understanding of memory management concepts through dynamic allocation of nodes and pointer manipulation. A successful implementation of the undo feature through the incorporation of a stack is also a major feature of this system.

This project has successfully fulfilled its objective of providing a simple model for a frame simulation environment through the incorporation of multiple data structures in a simple yet effective manner.

## 4.1 Features

The Video Frame Management System provides several functional features for frame simulation and management.

**1. Frame Addition**

- Users can dynamically add frames to the system.
- Each frame is automatically assigned a frame number and simulated timestamp.
- Frames are stored using a singly linked list data structure.

**2. Frame Deletion**

- Frames can be deleted based on frame number.
- The system searches the linked list and removes the selected frame node.

**3. Undo Delete Operation**

- The undo feature restores the most recently deleted frame.
- Stack data structure is used to implement Last-In-First-Out (LIFO) restoration.

**4. Frame Processing Simulation**

- Frames are processed sequentially using a queue data structure.
- The system demonstrates First-In-First-Out (FIFO) frame handling.

**5. Graphical User Interface**

- The system provides a simple interactive graphical interface developed using Raylib.
- Users can interact using buttons such as:
    - Add Frame

- Delete Frame
- Undo
- Process Frames

**6. Dynamic Memory Management**

- Frame nodes are dynamically allocated and deallocated.
- Pointer-based structure management is used.

**7. Simulation-Based Frame Visualization**

- The system simulates frame management rather than performing real video decoding.
- Provides an educational model for understanding frame traversal.

# Chapter 5 Conclusion and Recommendation

The Video Frame Management System was successfully designed and implemented as a simulation-based prototype for demonstrating frame management operations. The project achieved its primary goal of providing a simple graphical model for understanding frame-based processing without involving complex multimedia decoding techniques.

The system effectively demonstrates the use of fundamental data structures such as linked list, stack, and queue in practical application. The linked list structure allows dynamic frame storage and manipulation, while the stack structure supports undo functionality. The queue structure helps simulate sequential frame processing.

The graphical interface enables user interaction through button-based controls, making the system easier to operate and understand. Overall, the project helped in improving knowledge of pointer manipulation, dynamic memory allocation, event-driven programming, and data structure implementation.

Although the system is a simulation model, it serves as a useful educational tool for understanding basic multimedia frame management concepts.

## 5.1 Limitations

- The system does not support real video file decoding or playback.
- Frame data is manually simulated rather than extracted from actual multimedia sources.
- The graphical interface is basic and does not include advanced UI styling.
- The project is limited to educational simulation purposes and does not function as a full multimedia processing system.
- The system uses the Raylib library which is suitable for lightweight graphics but not for high-end multimedia processing.

## 5.2 Future Enhancement

- Integration of real-time video frame extraction and decoding modules.
- Development of improved buffering mechanisms such as circular queue implementation.
- Addition of audio processing and synchronization with frame playback.
- Support for multiple video formats and external media file processing.
- Introduction of machine learning-based frame analysis and automation features.
- Enhancement of graphical user interface design with modern visualization components.

# References

Raylib Documentation. Available at: https://www.raylib.com/

Bjarne Stroustrup, *The C++ Programming Language*, Addison-Wesley.

Introduction to Algorithms, Thomas H. Cormen, et al., MIT Press.

Data Structures and Algorithm Analysis in C++, Mark Allen Weiss.