**KATHMANDU UNIVERSITY**

Department Of Mathematics

Dhulikhel,Kavre



A Project Report

On

***“TUBE PUZZLE GAME”***

**[COURSE:COMP 202]**

# (For partial fulfillment of 2nd Year/ 1st Semester on Computational

Mathematics)

**Submitted by:**

Himanshu Gurung (07)

Swikar Jaiswal (10)

**Submitted to:**

**Dr Prakash Paudyal**

Department of Computer Science and Engineering

Submission Date: May 15, 2023

# ACKNOWLEDGEMENT

We would like to extend our sincere gratitude to our project supervisor, Dr. Prakash Paudyal, for his professional counsel and his efforts to provide inspiration and encouragement.He has been a constant source of inspiration for us and we are highly indebted to him for believing in us. Next, with our sincere regards, we would like to thank the Department of Mathematics, Kathmandu University (KU), School of Science(SOS) for including the project COMP 202 in our syllabus, which provided a platform and a network to allow this project to succeed.

Last but not least, we would like to extend our gratitude to all of our classmates who have made significant contributions through their criticism and counsel.This project was a great opportunity for us to learn new concepts and solve the problems that can arise during our project session. This will truly make us better individuals and help us in the near future.

Thank You!!

# ABSTRACT

The Tube Puzzle Game is a C++ project that presents an engaging and challenging puzzle-solving experience. It involves arranging a grid of tubes in a specific order by swapping them within columns. The project utilizes the wxWidgets framework for the graphical user interface, allowing for an intuitive and visually appealing game interface. Additionally, standard libraries and DSA concepts have been implemented, such as stack, linked list data structures for managing tube stacks, undo/redo commands and algorithms for tubes arrangement and game completion check. The project's significance lies in its application of data structures and algorithms, providing a practical implementation of DSA concepts. It serves as a valuable learning tool for DSA students, allowing them to see how these concepts can be used in real-world scenarios. As for future aspects of project, more complex levels and efficient operations can be implemented with better algorithms and data structure implementation.

**Keywords**: Tube Puzzle Game, C++, puzzle-solving, grid, tubes, swapping, wxWidgets framework, GUI, stack, linked list, undo/redo, algorithms, game completion check

# TABLE OF CONTENTS

**ACKNOWLEDGEMENT 2**

**ABSTRACT 3**

**TABLE OF CONTENTS 4**

**1.0 INTRODUCTION 5**

1.1 Background 5

1.1.1 Brief history 5

1.3 Objective 5

1.4 Significance 6

**2.0 RELATED WORKS 6**

**3.0 SYSTEM ANALYSIS AND DESIGN 7**

3.1 Design and Implementation 7

3.1.1 UI ( User Interface) 7

**4.0 CODE DESIGN AND ANALYSIS 15**

4.1 Execution of the code 15

**5.0 TIME COMPLEXITY 19**

**6.0 CONCLUSION 20**

6.1 Things we learnt 20

6.2 Conclusion 20

**7.0 REFERENCES 21**

# 1.0 INTRODUCTION

## 1.1 Background

The motivation behind developing the Tube Puzzle Game was to create a captivating and addictive puzzle-solving game that can be enjoyed by users in their leisure time. The game challenges players to arrange a grid of tubes in a specific order by swapping them within columns, testing their problem-solving skills and logical thinking abilities. By providing a variety of levels with increasing difficulty, the game aims to keep users engaged and motivated to improve their performance.

## 1.1.1 Brief history

The concept of arranging objects in a specific order or pattern has been prevalent in various puzzle games throughout history. The Tube Puzzle Game draws inspiration from this concept, offering a unique twist and digital adaptation. While the specific origins of the tube arrangement puzzle are unclear, similar puzzles have been popularized in different forms over the years.

## 1.3 Objective

The objective of the Tube Puzzle Game project is to create an interactive and challenging puzzle-solving experience for players. The project aims to:

* Develop a digital game that utilizes the concept of arranging tubes in a specific order, providing players with a unique and engaging gameplay mechanic.
* Incorporate algorithms and data structures to efficiently manage game elements, such as tube arrangement, undo/redo functionality, and game completion checks.
* Showcase the practical application of concepts related to data structures and algorithms, allowing students to gain hands-on experience in implementing these concepts in a real-world project.
* Offer multiple levels of difficulty and progressively introduce more challenges to keep players engaged and motivated.

## 1.4 Significance

The Tube Puzzle Game project holds significance in multiple ways:

* Educational Value: It provides practical implementation of data structures and algorithms, enhancing understanding and problem-solving skills for DSA students.
* Cognitive Development: Players engage in critical thinking and strategic planning, enhancing cognitive abilities such as pattern recognition and decision-making.
* Entertainment and Engagement: The game offers entertaining and challenging puzzles, appealing to gamers and casual players alike.
* Practical Application of Software Development Principles: It demonstrates modular programming, code organization, and naming conventions, showcasing good coding practices.
* Collaboration and Learning: Team members collaborate, share knowledge, and improve their coding skills through discussions and code reviews.

# 2.0 RELATED WORKS

Related works in the field of puzzle games and logic-based games have shown a wide range of approaches and implementations. Some notable examples include:

* "Tetris" by Alexey Pajitnov: A classic puzzle game that involves arranging falling blocks to complete lines.
* "Sudoku": A popular logic-based puzzle game where players fill a grid with numbers based on given clues and specific rules.
* "Candy Crush Saga" by King: A match-3 puzzle game that challenges players to align and match similar candies to achieve objectives.

While around the exact type of game, there are several applications especially for smartphones such as:

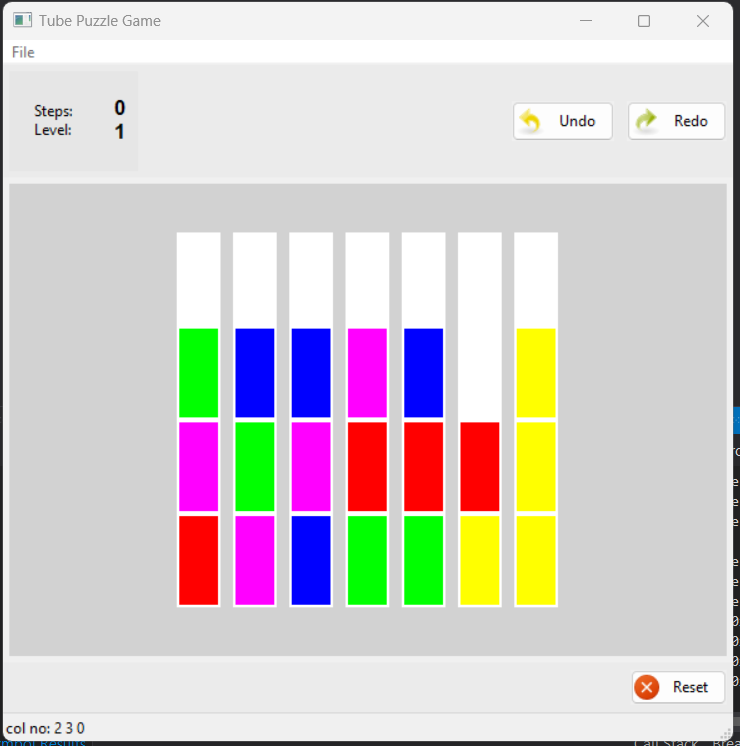
* Water Sort Puzzle
* Liquid Sort Puzzle
* Bottle Sort Puzzle

# 3.0 SYSTEM ANALYSIS AND DESIGN

3.1 Design and Implementation

This section covers the project UI designs, and the system features

3.1.1 UI (User Interface)



Steps Counter,  
Current Level

Tube

Column

Play Area

UndoButton

RedoButton

Title Bar

Reset Button

Fig: The Home Screen

Features:

* UndoButton: Reverts back the recent change
* RedoButton: Reverts back the change due to UndoButton
* ResetButton: Resets Play Area to Current Level’s initial state

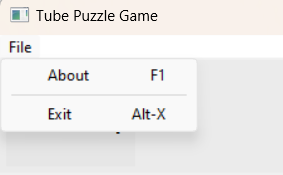


Fig: Menu Bar

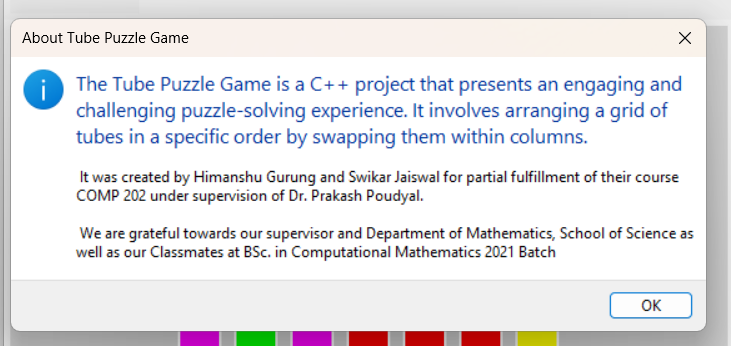


Fig: About Window

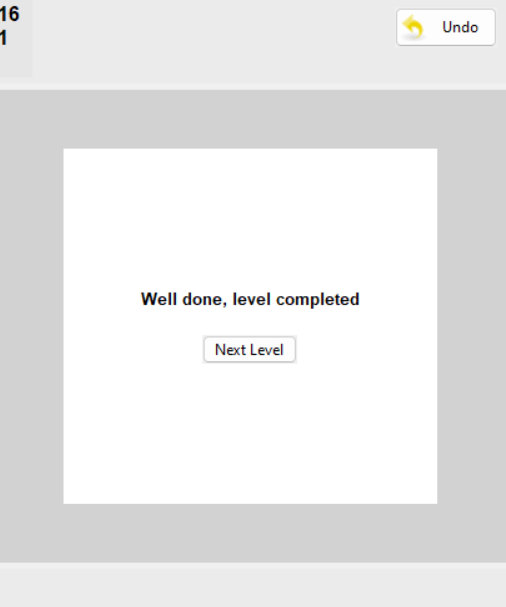


Fig: Level Completion Page

Features:

* NextLevel Button: opens Play Area for next Level

# 4.0 CODE DESIGN AND ANALYSIS

File structure:

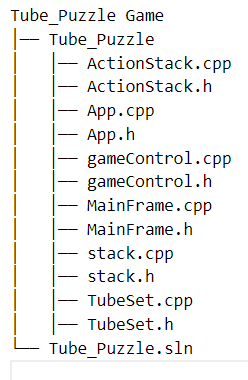
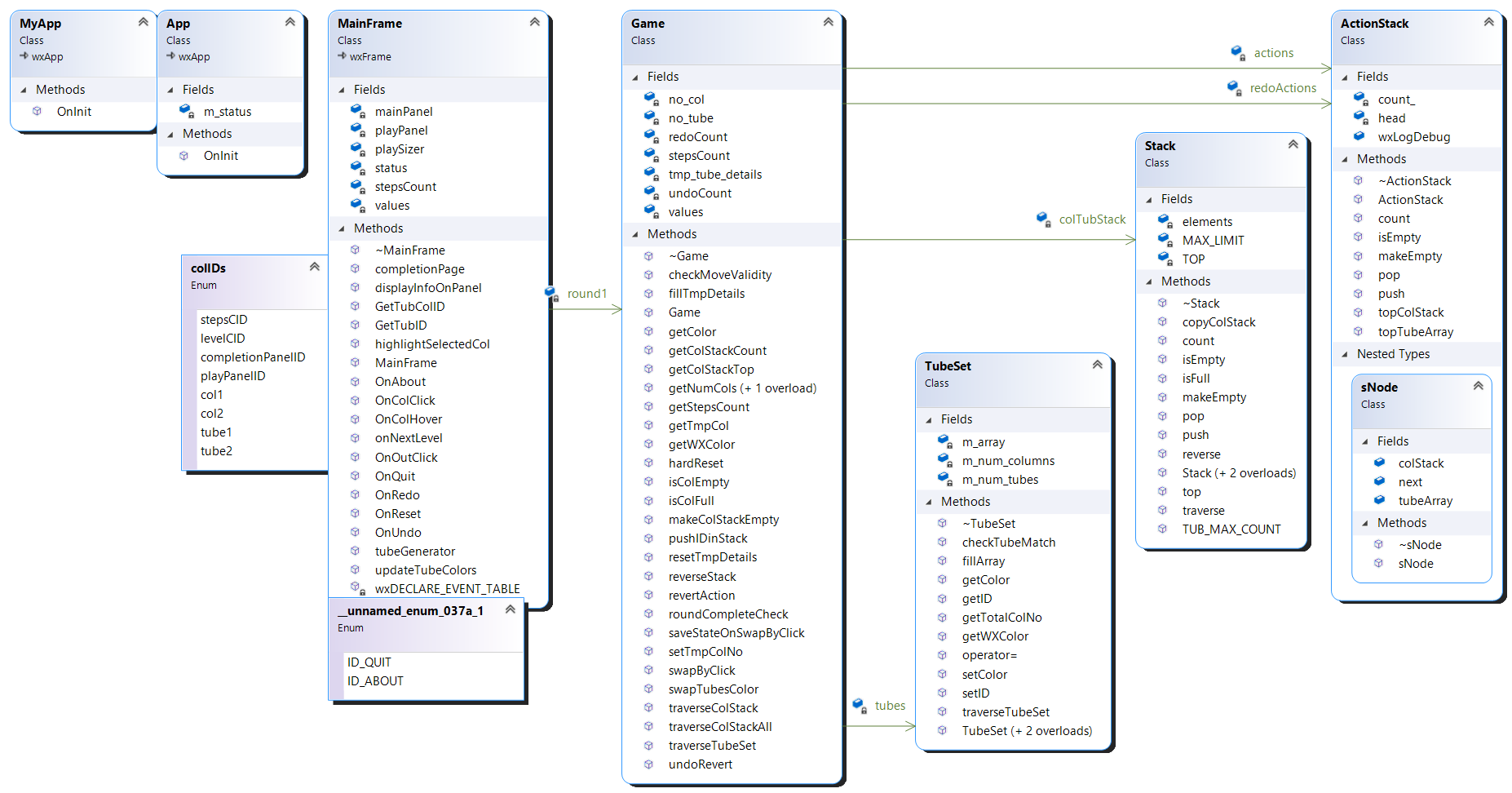


Fig: File Hierarchy

All the files required are stored in the Tube\_Puzzle directory within their respective subdirectories

|  |  |
| --- | --- |
| **Tube\_Puzzle.sln** | Visual Studio Solution File |
| **App** | App Class files |
| **MainFrame** | MainFrame Class files |
| **GameControl** | **GameControl** Class files |
| **ActionStack** | **ActionStack** Class files |
| **TubeSet** | **TubeSet** Class files |
| **Stack** | **Stack** Class files |

System Design:



Classes:

1. App class

It is the primary class that opens the app window.

|  |
| --- |
| App |
| #include "App.h"  #include "MainFrame.h"  #include <wx/wx.h>  #include<cstdlib>  class MyApp : public wxApp  {  public:      virtual bool OnInit()      {          MainFrame\* frame = new MainFrame(wxT("Tube Puzzle Game "));          frame->Show(true);          wxLogDebug("OnInit");          return true;      }  };  wxIMPLEMENT\_APP(MyApp); |

1. MainFrame class

This class sets the GUI layout of the Game and adds functionality to it.

|  |
| --- |
| MainFrame |
| #pragma once  #include <wx/wx.h>  #include "Stack.h"  #include "GameControl.h"  class MainFrame : public wxFrame {  public:      MainFrame(const wxString& title);      ~MainFrame();      void OnQuit(wxCommandEvent& event);      void OnAbout(wxCommandEvent& event);      void onColHover(wxMouseEvent& event);      void onColClick(wxMouseEvent& event);      void onOutClick(wxMouseEvent& event);      int getTubColID(int tmp);      int getTubID(int tmp);      void updateTubeColors();      void highlightSelectedCol(int col\_n,int col=0);      void displayInfoOnPanel(int tubeID);        void OnRedo(wxCommandEvent& event);      void OnUndo(wxCommandEvent& event);      void OnReset(wxCommandEvent& event);      void onNextLevel(wxCommandEvent& event);      void completionPage();      void tubeGenerator(int arr[]);  private:      Game\* round1;      int GameLevel;      int\* values;      wxPanel\* mainPanel;      wxPanel\* playPanel;      wxBoxSizer\* playSizer;      wxStaticText\* stepsCount;      wxDECLARE\_EVENT\_TABLE();  };  enum {      ID\_QUIT = wxID\_EXIT,      ID\_ABOUT = wxID\_ABOUT  }; |

1. GameControl

This class stores all the possible game states and performs checks, tube swaps, undo,redo,etc In short, acts as Backend for UI (MainFrame).

|  |
| --- |
| GameControl |
| #pragma once  #include "TubeSet.h"  #include "stack.h"  #include "ActionStack.h"  #include "Queue.h"  #include <wx/wx.h>  class Game  {  private:      TubeSet\* tubes;      int no\_col;      int no\_tube;      int stepsCount;      int undoCount;      int redoCount;      int\* values;      int tmp\_tube\_details[3];      Stack\*\* colTubStack;      ActionStack\* actions;      ActionStack\* redoActions;  public:      Game(int no\_col, int no\_tube, int\* values);      ~Game();      void resetTmpDetails();      void pushIDinStack(int col, int color);      void reverseStack(int col);      void fillTmpDetails(int col\_n, int color, int ID);      void setTmpColNo(int col);      int getTmpCol();      int getColStackTop(int col);    // get top element of stack      int getColStackCount(int col);  // get the number of elements in the stack      void makeColStackEmpty(int col\_n); //make stack empty      wxString traverseColStack(int col\_n); //traverse      wxString traverseColStackAll(); //traverse      wxString traverseTubeSet(); //traverse tube set      void swapTubesColor(int col1, int tub1, int col2, int tub2);      int getColor(int column, int tube);      wxColor getWXColor(int value);      int getNumCols() const;      int getNumTubes() const;      int getStepsCount() const;      bool isColEmpty(int col\_n);      bool isColFull(int col\_n);      bool swapByClick(int col\_n1, int col\_n2);      bool checkMoveValidity(int col\_n1, int col\_n2);      void saveStateOnSwapByClick();      void revertAction();      void undoRevert();      int roundCompleteCheck();      void hardReset();    }; |

1. ActionStack

This class provides Stack that can stores game states along with required Functionality.

|  |
| --- |
| ActionStack |
| #pragma once  #include "TubeSet.h"  #include "stack.h"  class ActionStack {  public:      ActionStack();      ~ActionStack();      bool isEmpty() const;      void push( TubeSet\* data, Stack\*\* other,int no\_col) ;      TubeSet\* pop();      TubeSet\* topTubeArray() const; // return TubeSet at Top      Stack\*\* topColStack() const; // return column stacks at Top      int count() const;      void makeEmpty();  private:      sNode\* head;      int count\_;      int COL\_MAX\_COUNT;  }; |
| class sNode {      public:          sNode( TubeSet\* data, Stack\*\* other,int no\_col, sNode\* next = nullptr) {              this->tubeArray = new TubeSet(data);              this->next = next;              Stack jpt;              colStack = jpt.copyColStack(colStack, other,no\_col);          }          ~sNode()          {              delete tubeArray;              delete colStack;          }          TubeSet\* tubeArray;          sNode\* next;          Stack\*\* colStack;      }; |

1. TubeSet Class

This class stores Tubes state and provides relevant functinalities.

|  |
| --- |
| TubeSet |
| //GameControl.h  #pragma once  #include <wx/wx.h>  class TubeSet {  public:      TubeSet();      TubeSet(int num\_columns, int num\_tubes);      TubeSet( TubeSet\* other);      ~TubeSet();      void fillArray(int\* values); // fill tubeset based on values      int getColor(int column, int tube) const; //return color at given indices      wxColor getWXColor(int value); //return wxColor object for given color value      int setColor(int column, int tube, int value);      int getID(int column, int tube) const; // returns ID stored at given indices      void setID(int column, int tube, int value);      wxString traverseTubeSet();      int checkTubeMatch(); // checks if tubeSet matches with Round Completion      int getTotalColNo(); // returns no of columns  private:      int\*\*\* m\_array;      int m\_num\_columns;      int m\_num\_tubes;  }; |

1. Stack Class

This class stores tubes in each column as Stack and provides relevant functionality.

|  |
| --- |
| Stack |
| #pragma once  #define \_CRT\_SECURE\_NO\_WARNINGS 1  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS 1  #include <iostream>  #include <wx/wx.h>  class Stack {  public:      Stack();      Stack(int limit); // Constructor      Stack(Stack\* data); // Constructor      ~Stack(); // Destructor      Stack\*\* copyColStack(Stack\*\* one, Stack\*\* two, int no\_col);      bool isEmpty() const; // Check if stack is empty      bool isFull() const; // Check if stack is full      void push(int data); // Push element to top of stack      int pop(); // Remove and return element from top of stack      int top() const; // Get value of top element without removing it      int count() const; // Get count of elements      void reverse(); // Reverse the elements in the stack      void makeEmpty(); // Make stack empty      wxString traverse() const; // Traverse the stack and return a string of all elements      int TUB\_MAX\_COUNT() const; // Get  private:      int MAX\_LIMIT;// Maximum limit of elements      int TOP; // Index of the top element in the stack      int\* elements; // Array of elements in the stack  }; |

## 4.2 Execution of the code

App

MainFrame

GameControl

ActionStack

TubeSet

Stack

Exchange Game States (Frontend < - >Backend)

Exchange Game States from Components

Initialization

Fig 14:Execution of the code

# 5.0 TIME COMPLEXITY

# Major Functions and their time complexities are presented below:

1. TubeGenerator: ɵ(n)
2. TubeArray Fill: O(col\*tub)
3. CompletionCheck: O(tub)
4. UpdateTubeColors: O(tub)

# 6.0 CONCLUSION AND RECOMMENDATION

6.1 Things we Learnt

Throughout the process of creating our tube puzzle game, we gained valuable knowledge and skills. Here are some of the things we learned:

1. We became proficient in C++ programming language, which helped us understand concepts like classes and inheritance. It was exciting to see how these programming techniques could be used to create a fun and interactive game.
2. We discovered how to develop graphical user interfaces (GUIs) using a framework called wxWidgets. This involved creating windows, buttons, and panels to make our game visually appealing and user-friendly.
3. Implementing data structures like stacks and arrays taught us how to efficiently manage the game's elements. We learned how to organize and manipulate data to keep track of the tube arrangement and other game features.
4. Designing game logic and algorithms was challenging but rewarding. We had to come up with algorithms for tasks like tube arrangement, undo/redo functionality, and checking if the game was completed. It was exciting to see how these algorithms made our game work smoothly.
5. Debugging and troubleshooting became a regular part of our development process. We encountered various issues and learned how to identify and fix bugs. It taught us perseverance and problem-solving skills.
6. Working as a team helped us improve our collaboration and project management abilities. We divided tasks, communicated effectively, and supported each other throughout the project. It was a great learning experience in teamwork.

Overall, this project provided us with practical knowledge in programming, problem-solving, and teamwork. It was an exciting journey that expanded our skills and understanding of computer science.

6.2 Conclusion

The development of our tube puzzle game was an enriching and fulfilling experience. We successfully created a challenging and entertaining game using our newfound knowledge of programming concepts, data structures, and algorithms. The project not only honed our technical skills but also taught us the importance of teamwork, perseverance, and problem-solving. We are proud of the final outcome and believe that our game showcases our dedication and passion for computer science. Through this project, we have not only created a fun game but also gained invaluable insights into the world of software development. We hope that our game brings joy to its players and inspires others to explore the exciting possibilities of coding and game development.

# 7.0 REFERENCES

GUI learning Playlist wxwidgets C++:

<https://www.youtube.com/watch?v=BjQhp0eHmJw&list=PLFk1_lkqT8MbVOcwEppCPfjGOGhLvcf9G>

Event handling: <https://www.youtube.com/watch?v=5v00F8hEV7E>

Stack Overflow - [https://stackoverflow.com](https://stackoverflow.com/)

wxWidgets Documentation - <https://docs.wxwidgets.org/>

Visual Studio Community - <https://visualstudio.microsoft.com/vs/community/>