COMP315

Project Documentation

Shadows of Redemption

Members	
1. Kival Mahadew	221001688
2. Ali Murtaza Caunhye	221007132

Contents

Introduction 3
Question Structure 4
1. True/False questions, Multiple-Choice Questions 4
2. Questions appear individually 4
3. Question order is different for every quiz attempt 4
4. Unique question order - no duplicates 5
User Interaction 6
1. User input 6
2. User feedback 7
3. Score accumulator 8
Levels and Progression 9
1. Various levels 9
2. At least three levels and eight questions each 10
3. Display progress of the quiz 11
Programming Techniques 12
4. Function 12
5. Class 13
6. Struct 15
7. Pointer 16
8. Reference 17
9. Data Structures 18
10. Class Template 19
11. Function Template 21
12. Operator Overloading 22
Graphics Interface 24
Additional Game 26
Additional Item/s 27
Appendix 28
Contributions 29

Introduction

A murder mystery quiz game, where the player's memory is tested based on the story of the game. The story follows Detective Conan, who is tasked with solving the murder of Elizabeth Killingsworth. The player is shown the dialogue between characters in the game and thereafter asked questions based on the dialogue. There is a confidence score which depicts the police department's confidence in Detective Conan. If the confidence score drops too low, Detective Conan gets fired and the game ends.

Question Structure

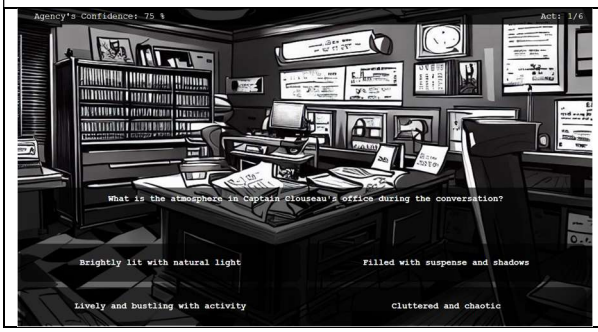
1. True/False questions, Multiple-Choice Questions

Screenshot	Explanation
	A statement is shown, and the player has to choose whether this statement is true or false based on the dialogue from this act.

2. Questions appear individually

Screenshot	Explanation
	There is only one question appearing at a time. For each question. The player has 4 possible answers to choose from and once they have chosen an answer it will then move on to the next question.

3. Question order is different for every quiz attempt

Screenshot	Explanation
	Here, for 2 separate playthroughs of the game the first question is not the same and it will randomly choose a question from a list of questions to present to the player.

4. Unique question order - no duplicates

Screenshot	Explanation
 	No question is asked twice. Once a question is asked, it will never be asked again.

User Interaction

1. User input

Screenshot	Explanation
	For each question that is not true/false question, the player has 4 options to choose from and may click on what they believe to be the correct option.

Code Screenshot

```
w->bind("option_1", {[this](const string &r) -> string
    {draw_answer_stage(0);return ""; }});

utils::set_hidden(w.get(), "option_2", false);
w->bind("option_2", {[this](const string &r) -> string
    {draw_answer_stage(1);return ""; }});

if (dialog->options[2] != "")
{
    utils::set_hidden(w.get(), "option_3", false);
    w->bind("option_3", {[this](const string &r) -> string
        {draw_answer_stage(2);return ""; }});
}
else
{
    utils::set_hidden(w.get(), "option_3", true);
}

if (dialog->options[3] != "")
{
    utils::set_hidden(w.get(), "option_4", false);
    w->bind("option_4", {[this](const string &r) -> string
        {draw_answer_stage(3);return ""; }});
}
else
{
    utils::set_hidden(w.get(), "option_4", true);
}
```

2. User feedback

Screenshot	Explanation

	Whenever the player gets a choice correct or incorrect, it displays them with a message saying so.

Code Screenshot
`` if (dialog->correct_option == answer) {   utils::set_text(w.get(), "char-text", "Correct!");   ++(*game); } else {   utils::set_text(w.get(), "char-text", "Incorrect!");   --(*game); } ```

3. Score accumulator

Screenshot	Explanation
	This is the agency's confidence in Detective Conan. It will adjust every time a question is

	answered. If the score goes to zero, the player loses the game.

Code Screenshot

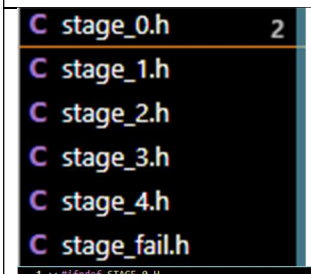
```
void Game::correct_answer()
{

    if (score == 100)
        return;

    // we will only increase the score if the difficulty is standard
    switch (difficulty)
    {
    case 1:
        score += 5;
        break;
    default:
        break;
    }
}
void Game::incorrect_answer()
{

    // we will only decrease the score if the difficulty is standard or hard
    switch (difficulty)
    {
    case 1:
        score -= 5;
        break;
    case 2:
        score -= 5;
        break;
    default:
        break;
    }

    if (score == 0)
    {
        stage = STAGE_FAIL;
        current = &stage_fail_root;
        current_stage = 12;
        return;
    }
}
```

Levels and Progression

1. Various levels

Screenshot	Explanation
	There are 6 stages to the game, each one progressing further into the story/investigation, as well as unlocking a unique quiz.

Code Screenshot




```
#ifndef STAGE_0_H
#define STAGE_0_H

#include <string>
#include "dialog.h"

// Narrator introduction
inline extern Dialog dialog_0_19 = create_stage_end("End of stage 1", 2, 14);
inline extern Dialog dialog_0_18 = create_npc_dialog("I hope you mean it, Conan. The department's future rests in your hands. Don't let us down.", 2, 0, &dialog_0_19);
inline extern Dialog dialog_0_17 = create_npc_dialog("You won't be disappointed, Captain. I'll give it my all. This is my shot at redemption, and I won't let it slip away.", 0, 0, &dialog_0_18);
inline extern Dialog dialog_0_16 = create_npc_dialog("That's the spirit, Conan. Prove to me that you still have that fire. Solve this case, find the truth, and make sure justice is served. We're
inline extern Dialog dialog_0_15 = create_npc_dialog("Nightingale? The new guy? I won't let him steal my chance at redemption.", 0, 0, &dialog_0_16);
inline extern Dialog dialog_0_14 = create_npc_dialog("Damn right it would be. We can't afford any slip-ups. Trevor Nightingale is waiting in the wings, ready to take your place if you falter.", 2
inline extern Dialog dialog_0_13 = create_npc_dialog("I understand the gravity of the situation, Captain. Losing this opportunity would be disastrous for both of us.", 0, 0, &dialog_0_14);
inline extern Dialog dialog_0_12 = create_npc_dialog("You've got it. She's surrounded herself with trusted individuals, hidden away in that mansion of hers. We're putting our last hope in you, Co
inline extern Dialog dialog_0_11 = create_npc_dialog("Elizabeth Killingsworth, the reclusive magnate? The stakes are high, I see.", 0, 0, &dialog_0_12);
inline extern Dialog dialog_0_10 = create_npc_dialog("We're in a financial crisis, Conan. The department's on the edge. We need a breakthrough, a high-profile case that will restore our reputatio
inline extern Dialog dialog_0_9 = create_npc_dialog("I know, Captain. I've hit a rough patch. But what's this all about?", 0, 0, &dialog_0_10);
inline extern Dialog dialog_0_8 = create_npc_dialog("Exactly. That case put you in the spotlight, but it's been years since then. And lately, you've been struggling to find leads. You need someth
inline extern Dialog dialog_0_7 = create_npc_dialog("Yeah, it feels like a lifetime ago. A queen murdered for a pink diamond, and the culprit running a black market museum.", 0, 0, &dialog_0_8);
inline extern Dialog dialog_0_6 = create_npc_dialog("You remember the Pink Panther case, don't you? The one that shot you to fame?", 2, 0, &dialog_0_7);
inline extern Dialog dialog_0_5 = create_npc_dialog("What's on your mind, Captain?", 0, 0, &dialog_0_6);
inline extern Dialog dialog_0_4 = create_npc_dialog("Take a seat, Conan. We need to have a serious talk.", 2, 0, &dialog_0_5);
inline extern Dialog dialog_0_3 = create_npc_dialog("Detective Conan entered Captain Clouseau's dimly lit office, the faint glow of a desk lamp casting shadows on the worn-out wooden furniture.",
inline extern Dialog dialog_0_2 = create_npc_dialog("The game will now begin", 1, 14, &dialog_0_3);
inline extern Dialog dialog_0_1 = create_npc_dialog("We hope you have read the `How to play` section in the main menu before starting this. Otherwise gameplay elements will seem a bit foreign.",
inline extern Dialog stage_1_root = create_npc_dialog("Welcome to Mystery Noir: Shadows of Redemption", 1, 14, &dialog_0_1);

#endif // STAGE_0_H
```

2. At least three levels and eight questions each

Screenshot	Explanation

	Here are the first 3 stages of the game: Stage 1: This is the introduction, where Detective Conan is talking to Captain Clouseau and he is being briefed on the case. Stage 2: This is the crime scene where Detective Conan is searching for Clues. Stage 3: This is where Detective Conan begins conducting interviews on the first few suspects. Each stage is followed by a quiz on the topics of the stage.

Code Screenshot

```
void Game::randomize_quiz()
{
    int quiz_len = 8;

    std::random_device rd;
    std::mt19937 rng(rd());

    int size = 17;

    std::shuffle(stage_1_questions, stage_1_questions + size, rng);
    // fill the question arrays
    for (int i = 0; i < quiz_len; i++)
    {
        Question *q = stage_1_questions[i];
        string choices[4] = {q->choices[0], q->choices[1], q->choices[2], q->choices[3]};
        quiz_one[i] = create_choice_mcq(q->question, nullptr, choices, q->answer_index);
    }
    // now point each question to the next
    for (int i = 0; i < quiz_len - 1; i++)
    {
        quiz_one[i].next = &quiz_one[i + 1];
    }
    // set the last question to point to the stage end
    quiz_one[quiz_len - 1].next = new Dialog(create_stage_end("You have completed the first stage quiz!", 1, 0));

    size = 18;
    std::shuffle(stage_2_questions, stage_2_questions + size, rng);
    // fill the question arrays
    for (int i = 0; i < quiz_len; i++)
    {
        Question *q = stage_2_questions[i];
        string choices[4] = {q->choices[0], q->choices[1], q->choices[2], q->choices[3]};
        quiz_two[i] = create_choice_mcq(q->question, nullptr, choices, q->answer_index);
        quiz_two[i].bg = 2;
    }
    // now point each question to the next
    for (int i = 0; i < quiz_len - 1; i++)
    {
        quiz_two[i].next = &quiz_two[i + 1];
```

3. Display progress of the quiz

Screenshot	Explanation
	Above the question, the display indicates what question the player is currently on and how many questions are left till the end of the act.

Code Screenshot

```
if (dialog->type == DialogType::QUESTION)
{
    utils::set_text(w.get(), "char-name", "Q " + to_string(game->current_question) + "/8");

    cout << "Drawing Question" << endl;
    // onclick we will go to the next dialog
    utils::set_hidden(w.get(), "options", false);
```

Programming Techniques

4. Function

Screenshot:

```cpp
std::string utils::load_img_to_base64(const std::string &path)
{
    std::ifstream image_file(path, std::ios::binary);
    if (!image_file)
    {
        std::cerr << "Failed to open the image file: " << path << std::endl;
        return "";
    }

    std::ostringstream oss;
    std::copy(std::istreambuf_iterator<char>(image_file), {}, std::ostreambuf_iterator<char>(oss));

    const std::string &file_data = oss.str();
    std::string base64_encoded = utils::construct_img_url(utils::base64_encode(file_data));

    return base64_encoded;
}
```

Motivation:

This is a function for loading an image from the file system given a relative path to the file. This is a function that is utilised largely when initialising the program. Modularising this functionality makes our code much cleaner and easier to test and debug.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	This is a single example of a function from the larger program that showcases our use. This function serves a single purpose of loading an image from a file, and is a good example of functional programming.

5. Class

Screenshot:

```
class Game
{
public:
    // constructor
    Game();
    ~Game();

    void set_difficulty(int difficulty);

    // the current dialog node
    Dialog *current;

    // holds a list of all the characters in the game
    Character *characters;

    // holds the current stage
    Stage stage = STAGE_1;

    // overload ++ and -- to call correct_answer() and incorrect_answer() on a player'
    void operator++();
    void operator--();

    // getters for the game's score
    int get_score()
    {
        return score;
    }

    // function to be called to progress to the next stage
    void next_stage();

    // pulls the question pool, randomizes the content, and builds a dialog tree
    void randomize_quiz();

    // is the current stage a quiz?
    bool is_quiz();
    bool is_memory();

    int
    current_act();

private:
    // the player's score
    int score = 75;

    // the game's difficulty
    int difficulty = 0;
```

Motivation:

A game class is necessary to keep track of variables such as the score, current dialogue and difficulty level. It allows us to keep all core game functionality in a single place.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	Having a single large class to represent our game lets us organise our game logic in a

		single place as well as control how the different pieces of the game interact with each other. The proper use of private and public variables lets us keep stray variables from existing in other parts of the program. It also lets us create an instance of the game every time it is started, solving any runtime issues caused by restarts

6. Struct

Screenshot:

```
struct Dialog
{
    // A piece of text that the character says
    string text;

    // the index of the background to display
    int bg = 0;

    // the id of the character to display
    int speaking = 0;

    // the id of the character to display
    int char_to_draw = 0;

    // The type of dialog
    DialogType type = DIALOG;

    // the next dialog to be displayed if this is a dialog
    Dialog *next = nullptr;

    // The options to be made available to the player if this is a choice
    string options[4];
    int correct_option = 0;
};
```

Motivation:

Since the dialogue is called by multiple files and multiple areas of code, we need to store its variables as global so that it is easily accessible. As the struct is only used as part of a larger list, it has no purpose other than to store information, it also provides no member functions, so a class definition would be unsuitable.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	This is a single example of a struct in the program. The dialog nodes serve no other purpose other than to hold data. A class definition would be unsuitable.

7. Pointer

Screenshot:

```
// pointers to dialog roots
Dialog *stages[14] = {
    &stage_1_root,
    quiz_one,
    &stage_2_root,
    quiz_two,
    &stage_3_root,
    quiz_three,
    &stage_4_root,
    quiz_four,
    new Dialog(create_stage_end("You have completed the matching game", 4, 0)),
    &stage_5_root,
    new Dialog(create_end_dialog("You have completed the game", 5, 0)),
    new Dialog(create_end_dialog("You have completed the game", 5, 0)),
    &stage_fail_root,
    new Dialog(create_end_dialog("You have failed the game", 5, 0))};
```

Motivation:

An array named stages is declared, which holds pointers to different stages of a game. Each element in the array represents a stage or a dialog in the game. Pointers enable storing different types of objects in the same array, providing flexibility in the structure of the program.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	The stages pointer holds a list of root nodes to the various dialog stages of the game. This is declared statically and will exist throughout the lifetime of the program. As a further example of the dialog struct, has pointers to the next node in the linked list, which also displays advanced use of pointers. See Data Structures below.

8. Reference

Screenshot:

```
std::string utils::load_img_to_base64(const std::string &path)
{
    std::ifstream image_file(path, std::ios::binary);
    if (!image_file)
    {
        std::cerr << "Failed to open the image file: " << path << std::endl;
        return "";
    }

    std::ostringstream oss;
    std::copy(std::istreambuf_iterator<char>(image_file), {}, std::ostreambuf_iterator<char>(oss));

    const std::string &file_data = oss.str();
    std::string base64_encoded = utils::construct_img_url(utils::base64_encode(file_data));

    return base64_encoded;
}
```

Motivation:

This function takes a reference to the path of an image, it then uses the path to build an input stream of the data of the file. It then copies the data into a stringstream. We take a reference of a string of the data of the string stream. We encode the string and return a built base64 string.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	This displays advanced use of reference. initially with passing the path, which is called many times when initialising the program, as well as taking a reference to the string content of the stream rather than copying it into the file_data variable. This keeps our memory usage from doubling during this step.

9. Data Structures

Screenshot:

```
struct Dialog
{
    // A piece of text that the character says
    string text;

    // the index of the background to display
    int bg = 0;

    // the id of the character to display
    int speaking = 0;

    // the id of the character to display
    int char_to_draw = 0;

    // The type of dialog
    DialogType type = DIALOG;

    // the next dialog to be displayed
    Dialog *next = nullptr;

    // The options to be made available to the player if this is a choice
    string options[4];
    int correct_option = 0;
};
```

Motivation:

The Dialog struct in the screenshot demonstrates the use of a Linked List data structure. A pointer to the next dialog in the list is stored in the *next pointer. We utilise the linked list to create a flowing dialogue system that progresses by advancing the node, rather than keeping track of an index. This allows us to create a recursive callback system that draws each screen, and will end when a node type is of 'End'.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	We feel the use of a linked list data structure in this part of the project showcases a common use of data structures in a program that involves linear dialog.

10. Class Template

Screenshot:

```
template <typename T>
class Cell
{
private:
    T value;
    bool isFlipped; // whether to display it as flipped or not
    bool isMatched; // whether it has been matched or not
    int x, y;       // position on the board

public:
    friend class MemoryGame;
    Cell(){}; // default constructor
    Cell(int x, int y, T value);
    ~Cell(){};
};

class MemoryGame
{
private:
    // Cell<int> will hold the index of the image in the images array,
    // which will double as a unique id
    // ranges from 1 to 8 eg. 2 of each value

    // 4x4 grid of cells
    Cell<int> cells[4][4];

    // pointers to the two flipped cells
    Cell<int> *first_cell = nullptr;
    Cell<int> *second_cell = nullptr;

    // first element is background, the following are the images of the cell
    std::string images[9];

    webview::webview *ui_context = nullptr;

    // this function checks if the two flipped cells match
    bool checkForMatch();

    // this function draws the initial state of the game
    void draw();
```

Motivation:

By using a class template, the Cell class can be instantiated with different data types (T). This allows the memory game to work with various types of values, not just integers. For example, it could be used to create a memory game with cells containing strings, images, or any other type of data that can be compared for matching.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		

Completely	X	This lets us create various type s of cells for our memory game. Though in implementation we just use it for a type of int, if we wanted to expand on this project it provides a good baseline to allow the memory game to store data other than just an int.

11. Function Template

Screenshot:

```
void set_text(webview::webview *w,
              const string &id, const string &text);

/*
 *   A template struct to check if a class has a to_string() function
 */
template <typename T>
struct has_to_string
{
    template <typename U>
    static auto test(U *p) -> decltype(to_string(*p), std::true_type());

    template <typename>
    static std::false_type test(...);

    static constexpr bool value = decltype(test<T>(nullptr))::value;
};

/*
 *   A template function for set
 */
template <typename T>
typename std::enable_if<has_to_string<T>::value>::type
set_text(webview::webview *w, const std::string &id, const T &text)
{
    set_text(w, id, to_string(text));
}
```

Motivation:

A helper function for setting the text 'set_text' is redefined as a template function that will call the original function with any type parameter that has a 'to_string' method. This allows us to have a function that can accept various data types that can be converted to string, without explicitly calling the 'to_string' method. This leads to cleaner code as well as not having to remember to call to_string for types like int and double.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	This was the only acceptable use of template functions in the program. As we are not building a game rather than a larger library that provides generics, our possible uses of various types are limited. In our case, this single use benefitted us greatly by not having to call to_string() everytime we wanted to set the text of a ui element.

12. Operator Overloading

Screenshot:

```
// overload ++ and -- to call correct_answer() and incorrect_answer()
void Game::operator++()
{
    correct_answer();
}

void Game::operator--()
{
    incorrect_answer();
}

void Game::correct_answer()
{

    if (score == 100)
        return;

    // we will only increase the score if the difficulty is standard
    switch (difficulty)
    {
    case 1:
        score += 5;
        break;
    default:
        break;
    }
}
void Game::incorrect_answer()
{

    // we will only decrease the score if the difficulty is standard or hard
    switch (difficulty)
    {
    case 1:
        score -= 5;
        break;
```

Motivation:

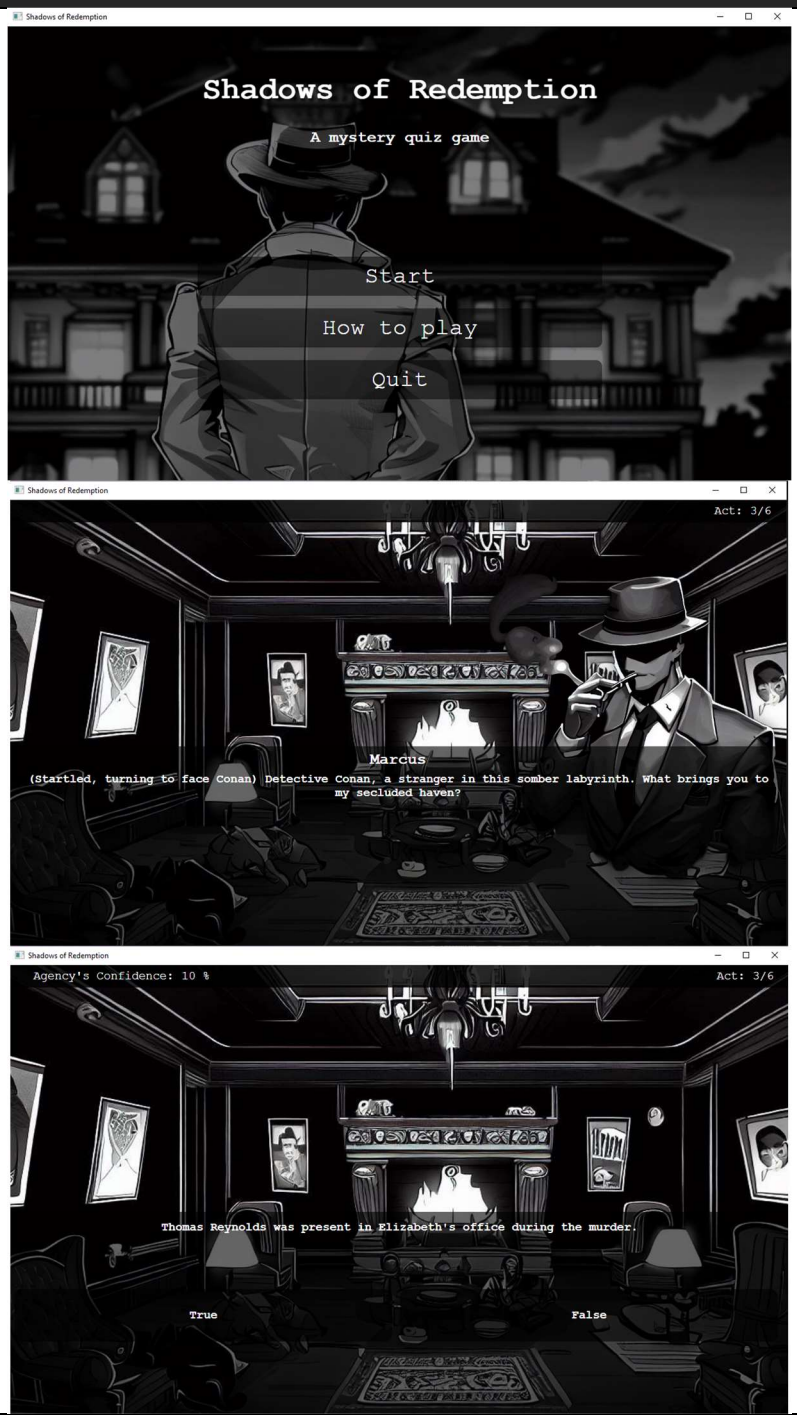
Here operator overloading is used on a 'Game' object to increment and decrement the player's score. In this case the overloaded operators call an existing private function in order to expose its functionality to the main program.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	This was the only acceptable use of operator overloading for this program. The score was the only piece of data that could realistically be incremented AND decremented on the game object.

Graphics Interface

We have used an open source project that provides a C++ API for the platform's native web browser, for Windows, Edge is used. https://github.com/webview/webview. It works by injecting static HTML into a browser window, and binding various DOM events to C++ callbacks. This provided a simple and lightweight, yet beautiful user interface.

Screenshot



Code Screenshot

```
void GUI::main_menu()
{
    // load the main menu
    w->set_html(this->html_templates[0]);

    // slight delay to allow the webview to load the html
    Sleep(100);

    utils::set_image(w.get(), "background", this->backgrounds[1]);

    // set bindings
    w->bind("start", {[this](const std::string &r) -> std::string
                    { this->game_settings(); return ""; }});
    w->bind("how_to_play", {[this](const std::string &r) -> std::string
                        { this->how_to_play(); return ""; }});
    w->bind("quit", [this](const std::string &r) -> std::string
            { w->terminate(); return ""; });
}
```

```
// set callbacks
if (dialog->type == DialogType::QUESTION)
{
    cout << "Drawing Question" << endl;
    // onclick we will go to the next dialog
    utils::set_hidden(w.get(), "options", false);

    // set the options
    utils::set_text(w.get(), "option_1", dialog->options[0]);
    utils::set_text(w.get(), "option_2", dialog->options[1]);
    utils::set_text(w.get(), "option_3", dialog->options[2]);
    utils::set_text(w.get(), "option_4", dialog->options[3]);

    // bind the options
    utils::set_hidden(w.get(), "option_1", false);
    w->bind("option_1", {[this](const string &r) -> string
                        {draw_answer_stage(0);return ""; }});

    utils::set_hidden(w.get(), "option_2", false);
    w->bind("option_2", {[this](const string &r) -> string
                        {draw_answer_stage(1);return ""; }});

    if (dialog->options[2] != "")
    {
        utils::set_hidden(w.get(), "option_3", false);
        w->bind("option_3", {[this](const string &r) -> string
                            {draw_answer_stage(2);return ""; }});
    }
    else
    {
        utils::set_hidden(w.get(), "option_3", true);
    }
```

Additional Game

Screenshot


Explanation of the Gameplay and screen
This is a traditional memory matching game where players will need to match pairs of correct images. They can view two cards at a time, if they match they will remain to be flipped over, if they do not they will be flipped back over. This will continue until all pairs are matched. **It stays true to the theme of our game with the idea of putting clues together and remembering vital information.**

Additional Item/s

Screenshot	What does your quiz include?
	**Our core quiz revolves around answering questions based on dialogue that was previously presented to the player. As an additional item we have included a full narrative following the story of Detective Conan solving the murder of Elizabeth, as well as various interviews with suspects. Conan will then use clues from these conversations in his investigation.**

Code Screenshot

```
inline extern Dialog dialog_1_15 = create_stage_end("This is end of stage 2.", 2, 14);
inline extern Dialog dialog_1_14 = create_npc_dialog("In this perilous stage, Detective Conan had uncovered vital clues. The blood-soaked cheque, Marcus's pen, the missing
inline extern Dialog dialog_1_13 = create_npc_dialog("(firmly): The window, a potential portal for the perpetrator. An avenue for intrusion or a means of evasion.", 0, 2, &
inline extern Dialog dialog_1_12 = create_npc_dialog("His piercing gaze shifted towards the open window, a gateway for both escape and entry.", 1, 2, &dialog_1_13);
inline extern Dialog dialog_1_11 = create_npc_dialog("(with a determined tone): The killer struck precisely, exploiting Elizabeth's moment of solitude. A calculated move, a
inline extern Dialog dialog_1_10 = create_npc_dialog("He pondered the timeline, knowing that the murder occurred between 12 and 1 pm, Elizabeth's regular meditation hour. T
inline extern Dialog dialog_1_9 = create_npc_dialog("(with a touch of intrigue): A fiery strand, a clue to the enigma. The mystery of the killer's appearance deepens.", 0,
inline extern Dialog dialog_1_8 = create_npc_dialog("As he meticulously examined the crime scene, a single strand of vibrant redhead hair caught his attention. It lay near
inline extern Dialog dialog_1_7 = create_npc_dialog("(whispering): Vincent, my old acquaintance. You played your part well. Elizabeth never stood a chance.", 0, 2, &dialog_
inline extern Dialog dialog_1_6 = create_npc_dialog("His eyes then darted to the gun cabinet on the wall, still intact but left open. A key remained in the lock, suggesting
inline extern Dialog dialog_1_5 = create_npc_dialog("(muttering): A silenced 10mm. A professional's weapon of choice. Deadly and discreet", 0, 2, &dialog_1_6);
inline extern Dialog dialog_1_4 = create_npc_dialog("His attention shifted to the gunshot wound on Elizabeth's head, a swift and lethal execution. The absence of any defens
inline extern Dialog dialog_1_3 = create_npc_dialog("(thoughtfully): A generous payment, tainted by blood. Marcus's involvement demands closer scrutiny. But where is he now
inline extern Dialog dialog_1_2 = create_npc_dialog("He carefully examined the scene, his keen eyes scanning every detail. His gaze fell upon a blood-soaked table, where a
inline extern Dialog dialog_1_1 = create_npc_dialog("(in a low, gravelly voice): Another soul lost to the darkness. No bruises, no signs of struggle. The killer moved swift
inline extern Dialog stage_2_root = create_npc_dialog("Detective Conan stood in the dimly lit office, a cloud of tension hanging in the air. The lifeless body of Elizabeth
```

```
struct Dialog
{
    // A piece of text that the character says
    string text;

    // the index of the background to display
    int bg = 0;

    // the id of the character to display
    int speaking = 0;

    // the id of the character to display
    int char_to_draw = 0;

    // The type of dialog
    DialogType type = DIALOG;

    // the next dialog to be displayed if this is a dialog
    Dialog *next = nullptr;

    // The options to be made available to the player if this is a choice
    string options[4];
    int correct_option = 0;
};
```

Appendix

Webview - https://github.com/webview/webview

This was the graphics library used for the game.

Images - https://www.bing.com/images/create

The bing image creator, powered by Dall-E was used to create all the images used in this project.

Contributions

Kival Roshan Mahadew	● Primary Focus ○ Creation of core dialog system and quiz logic ○ Integrating dialog and quiz system with user interface (HTML, callbacks etc.) ○ Loading files and templates at program start(utility functions) ○ Handling game progression and failure ○ Core game logic, including scoring, stages. ● Provided general contributions to project as a whole
Ali Murtaza Caunhye	● Primary focus ○ Memory Game(Minigame) implementation logic and GUI ○ Storyboard design and implementation ○ Definition and instantiation of 'Character' classes ○ Quiz Creation and implementation ● Provided general contributions to project as a whole

Note.

Pair programming was heavily utilised throughout the whole development process. This allows both members to provide deep insights into the design and implementation of various elements of the project.