Software Design Description

Doner

Table Of Contents

1 Introduction

This document presents the architecture and detailed design for the software for the Doner project. This project performs functions that allows users to create visual novels without programming knowledge and allow them to view changes instantly.

1.1 System Objectives

The objective of this application is to provide a game engine that serves solely for visual novels. It will provide a frame of visual novel and allows users to insert and edit texts, images and audios for it. This project will save what's been changed, and a export function is provided for the user to get their own game.

1.2 Hardware, Software, and Human Interfaces Section

1.2.1 Hardware Interfaces

This project requires the user to have a computer, a monitor, and both a mouse or a keyboard. Downloading this program requires internet connection, but using this program does not.

1.2.2 Software Interfaces

This project runs on the basis of Windows32 system and Windows vista, it requires OpenGL, c++14 compiler, and directX 11.

1.2.3 Human Interfaces

This project includes input from keyboard, and can be controlled with either a mouse or a touchpad.

2 Architectural Design Section

This application will have two part, the engine part and the game part. By creating a project, this application will create a program that could run as a visual novel game. The user will use the engine to modify a file that saves settings and data, and the game could read the data file to perform the game the user created.

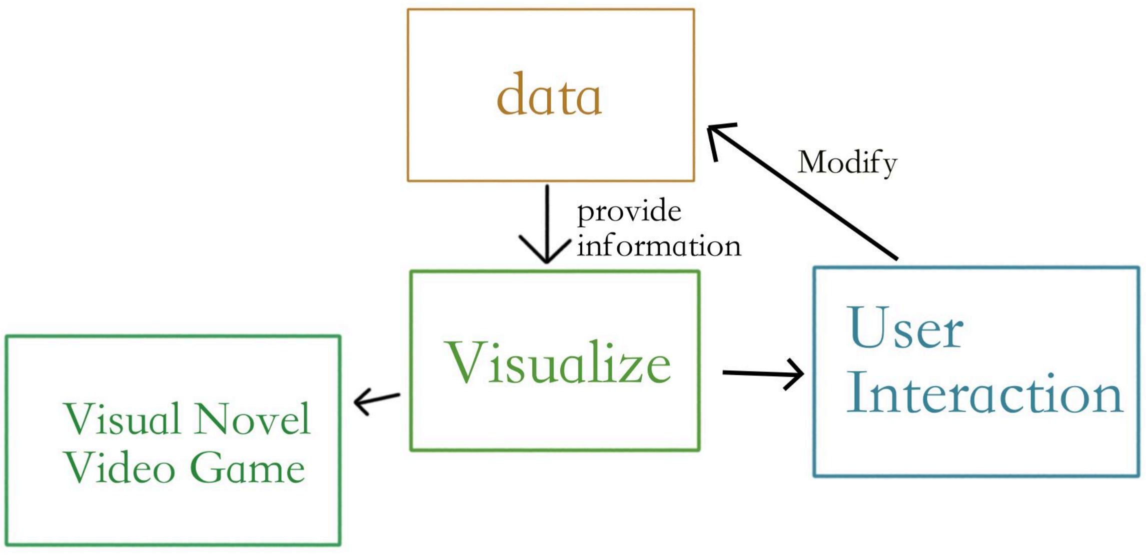
2.1 Major Software Components Section

This application should have the ability to read and write into files, as presenting their contents in the window. This application should also accept user's input on editing files.

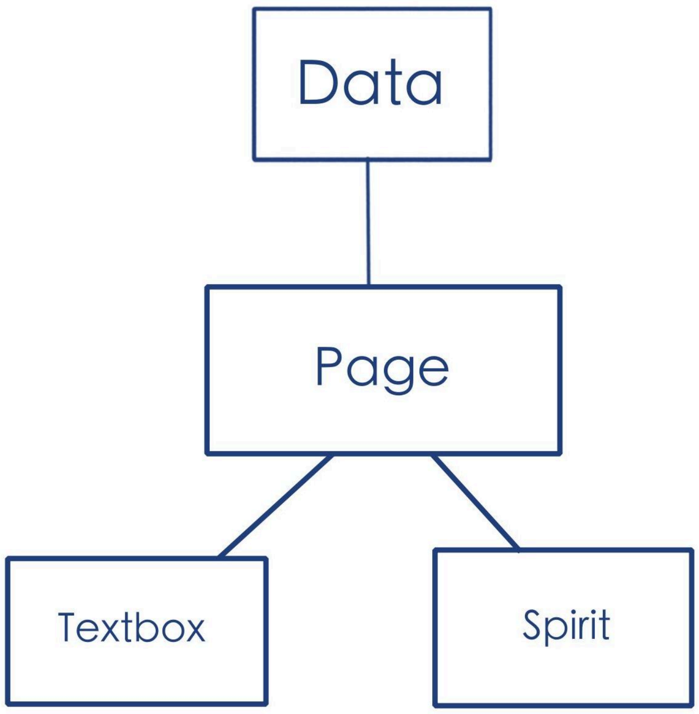
2.2 Major Software Interactions Section

In this application, the function about file, including reading, writing and storing data, should be in one segment. It should pass the data it reads to the modification segment, as these functions are presented to the user with the visualizing section which collaborate with ImGUI. On the other hand, the data that's being read could also be accepted by the section that presented the data in the form of game.

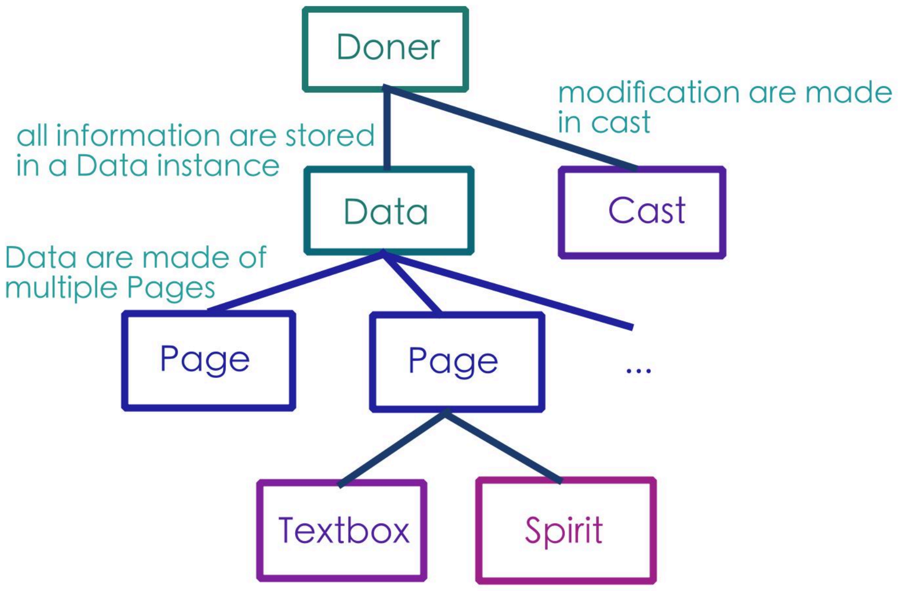
2.3 Architectural Design Diagrams Sections



Component Diagram



Data Structure Diagram


Doner
modification are made in cast
all information are stored in a Data instance
Data
Cast
Data are made of multiple Pages
Page
Page
...
Textbox
Spirit

说明

3 CSC and CSU Descriptions Section

The components of this CSCI are showed in the graph above. More explanations will be made below.

3.1 Detailed Class Descriptions section

3.1.1 Tools (Tools.h and Tools.cpp)

Tools is a helper class containing some methods that can be generally used in other classes.

Method:

static void setBackground(std::string background_name) // not in use

This method helps setting background picture with the given address of the background picture.

static std::string wordEncrypt(std::string word)

This method takes in a string, and output it in the given format:

```
/ -> //
# -> /#
{ -> /{
} -> /}
[ -> /[
] -> /]
```

3.1.2 texture (structure located inside Tools.h)

This structure stores what is needed for a texture

Field:

int w

contains the width of the texture

int h

contains the height of the texture

ID3D11ShaderResourceView* t

the texture as being loaded

3.1.3 Textbox (Textbox.h)

This class only have inline header file, and it stores the information of a single text box.

Field:

std::string name

the name of this text box

std::string content

the content in this text box

ImVec2 positionRatio

a vector of two float that represent coordinate of the right top corner of this textbox

ImFont* font

the font of this text box

std::string fontPath

the path of the file that contains the font this text box is using

ImColor color

the color of the text in this text box

float fontSize

the size of the text in this text box

Method:

Textbox()

the constructor for a new text box, currently not in use

Textbox(std::string data_str[10])

the constructor for a text box with content given, each of the element in data_str will be put into the field given above in order

3.1.4 Spirit (Spirit.h)

This class only have inline header file, and it stores the information of a single spirit.

Field:

private std::string spiritFileName

this is a private string storing the address of the file of this spirit

std::string spiritName

this is the nick name of this spirit given by the user, default the same as spiritFileName

float sizeRatio[2]

contains the ratio of the image size to the window size in the format [image_width / window_width, image_height / window_height]

float positionRatio[2]

contains the ratio of the position on the left top of the spirit to the window in the format [position_x / window_width, position_y / window_height]

Method:

Spirit(const std::string file_name)

a constructor that reads in the spirit's address, and set sizeRatio to [1, 1] and positionRatio to [0.25, 0.25].

Spirit(const std::string& name, const std::string& file_name, const float& sr_x, const float& sr_y, const float& pr_x, const float& pr_y)

a constructor that sets each variable to the field. Sr represent size_ratio, and pr represent positionRatio.

std::string getFileName()
std::string fileName()   // These two are for spiritFileName
std::string name()
std::string* getRealNickName()   // these two are for spiritName

getters for names, as "real" meaning getting the pointer

Imvec2 getSize(int x, int y)
Imvec2 getPosition(int x, int y)

getters for the true number, by multiplying x and y to the two elements in the ratio vectors.

std::string toString()

returns the encrypted string that contains all information in this spirit. Using Tools::wordEncrypt()

3.1.5 Page (Page.h and Page.cpp)

Page is a class that contains all information on a single page, thus it contains multiple spirits and textboxs. This class is also responsible for visualizing the page, painting it onto the window.

Field:

std::vector<Spirit> spirits
std::vector<Textbox> textboxs

the vectors storing the two classes items

std::string backgroundName

the string storing the address of the background picture

int pageID

an integer of this page's ID, for quick jump function (which is not yet implemented)

Method:

Page()

a constructor for an empty page, with no spirit, no textbox, no background, and pageID set to 0.

Page(int page_id)

a constructor for an empty page, with no spirit, no textbox, no background, and pageID set by given.

Page(int page_id, std::string page_data) a constructor setting everything according to given. The page_data will go through a process of decryption and send into each field.

static bool LoadTextureFromFile(const char* filename, ID3D11SharderResourceView** out_srv, int* out_width, int* out_height, ID3D11Device* g_pd3dDevice)

a helper function provided by ImGui using stb_image library to load picture as texture. Returns if the texture is loaded.

static void showBackGround(std::string file_name, ImVec2 windowSize, ID3D11Device* g_pd3dDevice, ID3D11ShaderResourceView* back_texture, int* back_w, int* back_h)

the method that prints the background picture in the background. This method will first load the background picture with LoadTextureFromFile, then write it into the drawList for the host window.

private static void showSpirit(std::string file_path, Spirit spirit, ImVec2 window_size, ID3D11Device* g_pd3dDevice, texture* texture)

A method that prints a single spirit. The spirit's image will be loaded with LoadTextureFromFile, and the result will be stored into the given texture. Then the given texture will be add to the draw list for the host window.

private static void showTextbox(Textbox textbox, Imvec2 window_size)

A method that prints a single text box. The texts will be add to the draw list for the host window.

void visualizePage(some param)   // not in use

void visualizePage2(some param)   // not in use

void visualizePage3(D3D11Device* g_pd3dDevice, ImVec2 windowSize, std::string file_path_str, std::vector<texture>* textureList)

The methods to add all elements in this page into the drawList for the window. It will call showBackground, showSpirit and showTextbox for each element.

std::string exportInString()

The method that put all information in this Page into a encrypted string, in the format [backgroundName{spirit1##spirit2##}{textbox1##textbox2##}], where backgroundName will go through Tools::wordEncrypt() and spirit and textbox will each go through their own encryption.

Spirit* getRealSpirits(int id)

Textbox* getRealTextbox(int id)

getters for the pointer to the corresponding element in the vector

void setFont(ImFont* font_given) // not in used

set the font for textboxs

3.1.7 Data (Data.h and Data.cpp)

The class used to store all data in a user's project. It contains the address of the project and multiple pages.

Field: (all private)

ImFont* font

font for the project

std::vector<Page> pages

the pages in this project

std::string fileName

the address for the data storing file

std::string fileData

the string that contains all information of this file

PWSTR pszFileName

the address of the data storing file in PWSTR

std::string demoPath

the address for the Demo file

PWSTR filePath

the PWSTR of the address of the folder the user's project is in

std::string filePathStr

the string of the address of the folder the user's project is in

Method:

data(std::string file_path)

the constructor reading in the address of Doner application, and have a default with a empty page.

void newFile(bool* start_visual)

The method is for starting modification without reading data. It will ask the user to choose a folder, then create a Doner project inside that folder. Then it will set start_visual to true to visual the page it is on.

void openFile(bool* start_visual)

The method is for starting modification with the folder or file the user selected. It will read the data in that folder or file, then it will set start_visual to true to visual the page it is on.

void openDemo(bool* start_visual)

The method is for starting modification while showing a given demo. The data in demo will be read, then it will set start_visual to true to visual the page it is on.

visualizeData(something) // not in use

visualizeData2(something) // not in use

void visualizeData3(ID3D11Device* g_pd3dDevice, ImVec2 windowSize, std::vector<texture>* textureList, int page_id = 0)

call visualizePage3 for the given page

void setFont(ImFont* font_given) // not in use

std::string fileData()

getter that returns fileData string

void save()

calls encryptIntoFile and write it into the project data file.

int pageSize()

returns the total number of pages

void addPage(int page_id)

add a empty page after the current showing page

void CopyPage(int page_id)

copy everything except for the content in textboxs on the current showing page, create and add the copy to the page right after the current page.

void DeletePage(int page)

delete the current page from Pages vector.

private std::string encryptIntoFile()

This method collects all encrypted pages and put them into one string with encryption in the format [page1][page2], then return it.

private void decryptFile(std::string data_str)

This method decrypt given data into pages, and pass them down to each class to decrypt.

private HRESULT basicFileOpen(bool findFile)

The helper function provided by Microsoft to open common window and let user choose the project they want to open.

3.1.8 Cast (Cast.h and Cast.cpp)

This class serves for modification on the data with visual ImGui windows.

Method:

static void showCastWindow(bool* p_open, int pageID, Page *pageInfo, ImVec2 window_size);

showing a ImGui window that present each cast (spirit, textbox and background) in the page with modifications

static void showWelcomePage(data* gameData, bool* show_welcome_window, bool* start_visual, bool* page_setting)

showing a ImGui window that each links to Data::NewFile, Data::OpenFile, Data::OpenDemo.

static void showFileWindow(bool* p_open, int pageID, Page* pageInfo)

not in used

static void showPageWindow(bool* p_open, int* pageID, data* game_data)

showing a ImGui window that allows the user to change the page they are on, and links to Data::addPage, Data::CopyPage, Data::DeletePage, with the two functions adding jumping to the newly added page, and deletePage to the previous page (if exist).

3.1.9 Doner (Doner.h, Doner.cpp and Doner.rc)

Doner is the main file for the application to run. Its main use is to communicate with the system to create a window which supports all other functions to run on it.

Field:

std::string PathLoc

The address of the application

std::string Path

The address of the application's "project storage" file

std::string DefaultBackground

The address of the background picture that will be show when there is no user's project opened

data gameData

The data of the user's project, empty in default

std::vector<texture> textureList(20)

The list of texture (the structure created earlier) the application is loading every frame. It is set to 20 because it is a fair amount that is either too little for the user or too much and take more memory than needed. The first element in this vector will always be for the background picture.

bool StartVisual

The boolean to set if data should be presented.

static ID3D11Device* g_pd3dDevice

static ID3D11DeviceContext* g_pd3dDeviceContect

static IDXGISwapChain* g_pSwapChain

static UINT g_ResizeWidth, g_ResizeHeight

static ID3D11RenderTargetView* g_mainRenderTargetView

The above are some global variable for the project to run in window with directX11

Method:

ATOM MyRegisterClass(HINSTANCE hInstance)

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

bool CreateDeviceD3D(HWND hWnd)

void CleanupDeviceD3D()

void CreateRenderTarget()

void CleanupRenderTarget()

helper functions provided by ImGui and Microsoft for the program to run on windows

LRESULT WINAPI WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)

Windows method to get call backs for menu bars int APIENTRY wWinMain(_In_ HINSTANCE hInstance, _In_opt_ HINSTANCE hPrevInstance, _In_ LPWSTR lpCmdLine, _In_ int nCmdShow)

Main method that loads the whole program. It contains all setters and a loop for the program to refresh lively.

3.2 Detailed Interface Descriptions Section

In this CSCI, Doner will initiate and get all setting and pointers for the project and window. It will create Data CSU and pass in window parameter to make sure the pages could show properly. Then it will pass the data to Cast, where cast would change it and pass it back. The process happening in the loop will be Doner passing Data to Cast, Cast changing Data and pass it back, Doner passing window parameters to Data, Data pass back visualization information to Doner, Doner shows it and start from the beginning of the loop again.

3.3 Detailed Data Structure Descriptions Section

Doner will have a Data file that contains all information of the user’s current project. Data contains the location of the project, the font for this project, and multiple Pages. Page contains the information about the page including background picture, multiple Spirits and Textboxs. Spirit contains all information needed for the image, such as it size, location, and filePath. Textbox contains the name and content for text.

3.4 Detailed Design Diagrams Section