University of Dhaka

Department of Computer Science and Engineering

CSE – 1201

Lab Project: Adventure of Chunchin

Submitted to:

Mr. Hasnain Heickal
Lecturer

Mr. Md. Mahmudur Rahman Rana
Lecturer

Mr. Abu Ahmed Ferdaus
Lecturer

Dr. Md. Haider Ali
Professor

Dr. Chowdhury Farhan Ahmed
Professor

Dr. Sarker Tanveer Ahmed Rumee
Professor

Submitted by:

Nayeemuzzaman, Roll-02
Rafid Ahmed, Roll-23

Date of Submission: 19th November, 2019

Table of contents:

Introduction:

This is a project is intended to make the students familiar with some application of basic C++ programming using SDL(Simple Directmedia Layer) Graphics interface. Implementation of the students' theoretical knowledge of C++ language in some real life scenario was the objective of this project.

Project overview:

The project is a story based 2D game (above camera view) where you have to go talk to some people and do some tasks(collecting items) for them to move on to the next stages unlocking new maps.

Project objective:

The main objective of the project is to simulate a story-based game where you can talk with people and do tasks. To give the user best experience as possible, the game was made aesthetic as much as possible, by using appropriate graphics, music, and many more extra features. It's a game where you just go along and enjoy the story.

Game outline:

There are 5 stages in this game.You have to basically finish all the stages and then exit the place to end the game.

Each of first 4 stages contain a person whom you talk to and then start helping them. After you helped them by collecting specific items that the person need, they will give you the key to next map where you will meet the next person to do the task.

In the final stage,you have completed everybody's task. You have to find exit to that place to finish the game.

There are two difficulties. Where you can choose if you want some additional features blocked or not.

Game overview:

(i) Stage 1:

In the beginning of the game, you are spawned into an place named “Land of opportunity” where you have to talk with Mario. He will ask you to collect some boxes for him (depending on difficulties) . After you finish the task. He will give you the key to the place called “Liberty City”. At stage 1, all the maps are unlocked. Completing Mario’s task, you will proceed to stage 2.

(ii) Stage 2:

In liberty city, you will meet an unknown person. He will say that he is tired and want some coffees. Your job is to collect coffees for him. After completing that. He will give you the key to a place named “Asgard”. And you’ll proceed to stage 3.

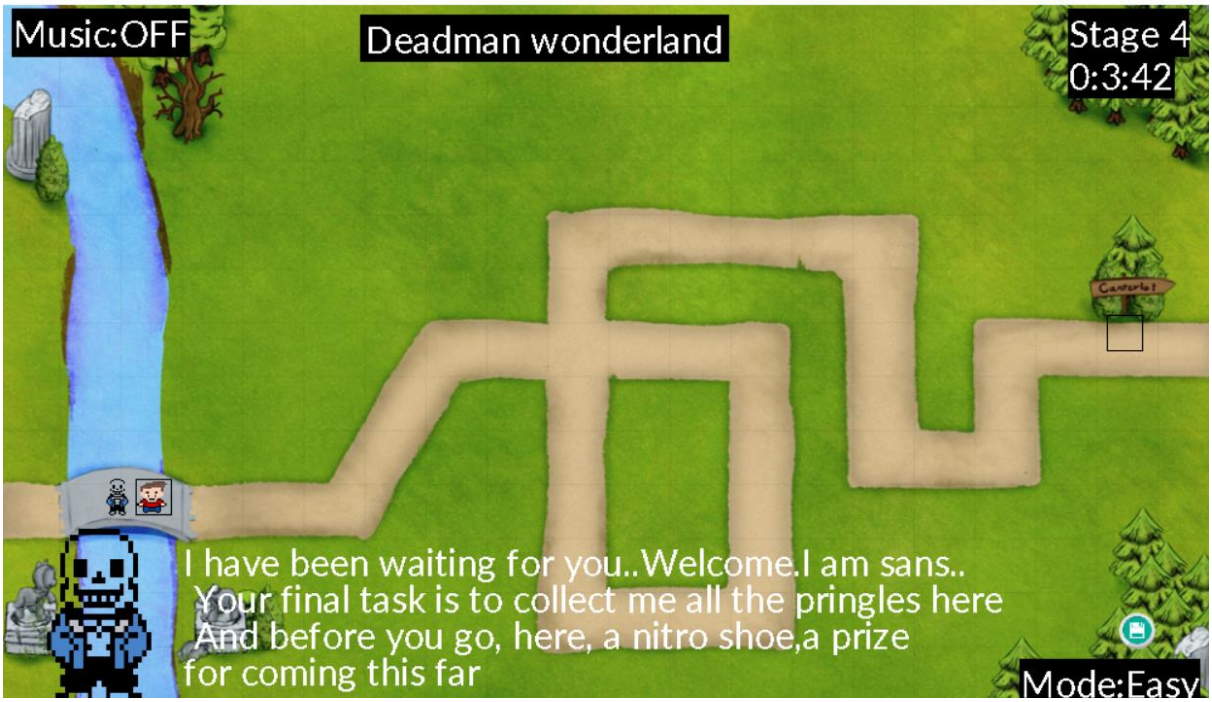
(iii) Stage 3:

In Asgard, you’ll find a someone named Papyrus who is obsessed with chocolates. Your job is to collect him chocolates and proceeds to stage 4 unlocking a place named “Deadman Wonderland”.

(iv) Stage 4:

You'll find a person named Sans at Deadman Wonderland where he'll congratulate you for coming this far. And then will ask you do the final task , which is collecting all the pringles there is in all of the places.

The major difference between this stage and the previous ones is that , after talking with Sans, you'll achieve nitro which you can use to speed up your movement.



(v) Stage 5:

After completing all of the missions, you'll have to exit the place. You can do that by going to "Liberty City" and exiting the game through the right gate

Resources used:

None of the images and music used here owned by us. Many of the materials used here belong to:

- Undertale
- Doki Doki Literature Club
- Super Smash bros.
- Random images found on google.

The entire source code is written based on the knowledge earned in http://lazyfoo.net/tutorials/SDL/index.php

Game features:

Main features:

(i) Main menu:
The main menu is like a dashboard that gives the user access to any part of the program.

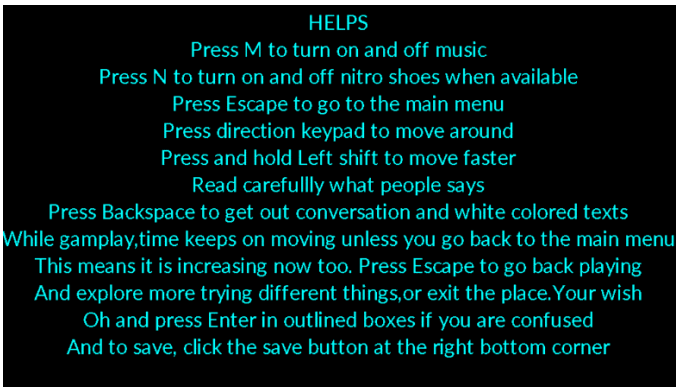
(ii) Time display:
At the upper right corner, time you spent during gameplay is displayed.

(iii) Leader board:
The game also has a leader board option that can be accessed through the main menu. The dashboard stores the top 5 scores based on minimal gamplay time.

(iv) Save game and load game:
During gameplay, you can save your state by clicking on the save icon at the bottom right corner of the screen. And from main menu, you can access the saved state anytime.

(v) Exit game:
You can exit the game from main menu by clicking exit bar.

(vi) Instruction page:
During gameplay, if you press H, instruction page will be displayed. This can help you to get an idea of how to play the game.



(vii) Sound effects and Music:

When you collect an item, there is a sound effect letting you know that you collected it. And also there is a background music option based on some stages, which you can turn on and off pressing M.

(viii) Fast movement and Nitro shoes:

During gameplay if you press and hold left shift, you will move faster. And at stage 4, after talking with Sans, you can move even faster by putting on nitro shoes, which you can do by pressing N.

(ix) Difficulty:

There are two difficulties in this game. In hard mode, you have to collect more items and you cannot save while collecting them which you can do if you play on easy mode.

Bonus features:

(i) Opening intro:

When you open a game, intro will be displayed which you can skip by pressing enter.

(ii) Use of both keyboard and mouse:

While using menu or other dashboards, you can use both keyboard and mouse.

(iii) User friendliness:

We've worked extra hard to make this game user friendly as much as possible. From various types of feedbacks to ending credits, there are a lot of small works done to make this game as user friendly as possible

Source Code properties:

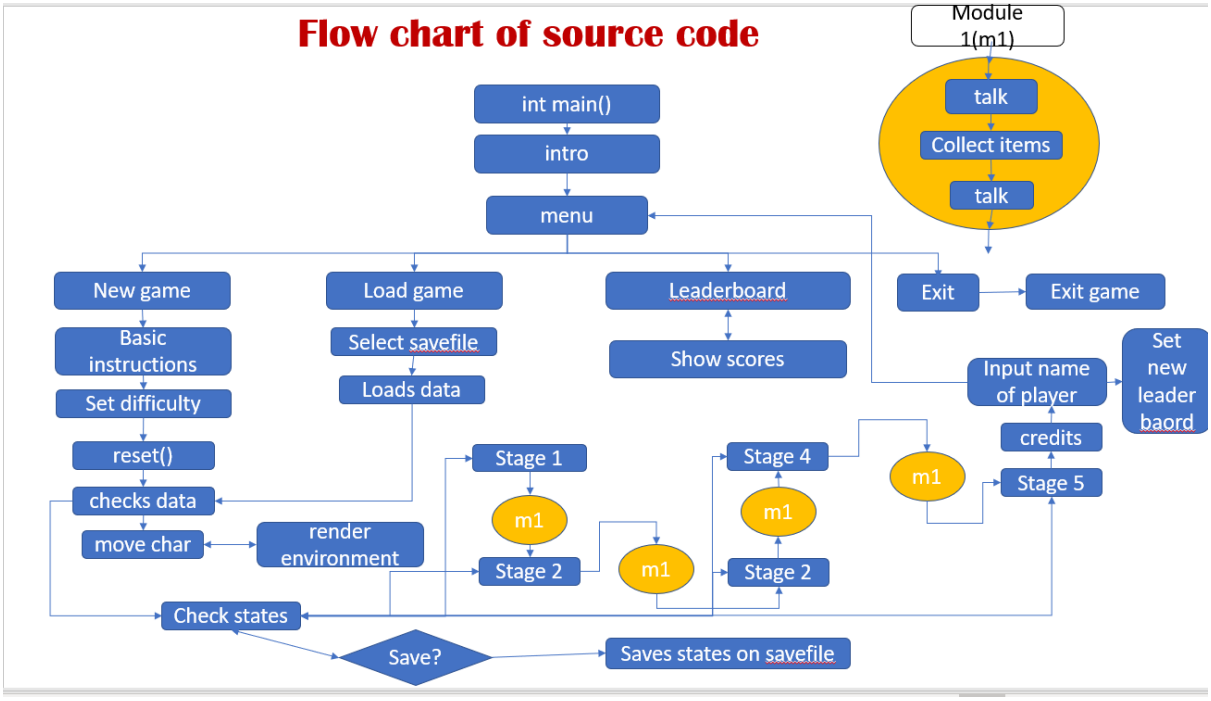
(i) The source was written by following an efficient modular programming system, dividing the whole code in different separate parts and making new user defined header and cpp files.

(ii) The coding part of this project contains-

a. Basic input output
b. File input output
c. Conditional logic
d. For loop and While loop
e. Single and multi dimensional arrays
f. String handling
g. Event polling
h. Keypresses fucntions
i. Pointers
j. Structures
k. Sorting
l. Reading image and music files
m. SDL built-in fucntions.

And so on.

The game structure:



As shown in the above figure, after running the main cpp file, intro is displayed and we are directed towards menu.

There are 4 choices in the menu.

In new game, after basic instruction and difficulty setting, the data is reset. Here the data are of character positions and environmental states. After that it starts checking the data and then rendering the environment and character position according to the data. And you pass each stage completing m1 . In stage 5, you exit the game stating your name. If you are good enough, you set your name on leader board.

Load game works same way, just instead of resetting the data, it reads a data from a saved file.

Leader board shows top 5 person who finished the game in minimal time.

If you click exit game option, you exit the game.

The files and functions used in the source code:

The list of header and cpp files:

01. initialization.hpp
    It contains all header files and all of the user-defined function declared

02. gamemedia.cpp
    It contains all works relating to loading media. And also button press functions.

03. gameaction.cpp
    It contains all of the rendering and character movements of the game.

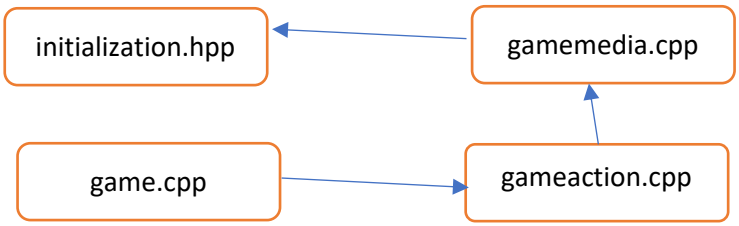
04. game.cpp
    It is the main cpp file of the game.

Functions used:

All of the user-defined functions are defined in initialization.hpp
We also used various SDL built-in functions like SDL_Render(), SDL_Delay() etc.

Layout of the header and cpp files:

The files here works linearly. The game.cpp calls gameaction.cpp, gameaction.cpp calls gamemedia.cpp, gamemedia.cpp calls initialization.hpp where all of the library files are declared.



Overview and conclusion:

The project was a very good way to encourage young students to develop their skills and make something useful with their knowledge. This project is a perfect example of all the exceptional things that can be done with some basic C++ programming.

This project will also encourage young minds who want to be a game developer in the future. The project promotes originality and creativity and the skill of working in a team.

The project also summarizes the basic tools of C++ language as they had to be implemented while making the game.

The source code:

For an easy understanding of the code, I've put the initialization.hpp first then game.cpp(main). Then gamemedia.cpp and gameaction.cpp.
For surface level understanding of the source code, a look at the initialization.hpp and game.cpp would be enough.

The "initialization.hpp" file:

```
 //Using SDL, SDL_image, standard IO, and strings
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_mixer.h>
#include <stdio.h>
#include <string>
#include <cmath>
#include <sstream>


//Texture wrapper class
class LTexture
{
  public:
    //Initializes variables
    LTexture();

    //Deallocates memory
    ~LTexture();

    //Loads image at specified path
    bool loadFromFile( std::string path );

    //Creates image from font string
    bool loadFromRenderedText( std::string textureText, SDL_Color textColor );

    //Deallocates texture
    void free();

    //Set color modulation
    void setColor( Uint8 red, Uint8 green, Uint8 blue );

    //Set blending
    void setBlendMode( SDL_BlendMode blending );

    //Set alpha modulation
    void setAlpha( Uint8 alpha );

    //Renders texture at given point
    void render( int x, int y, SDL_Rect* clip = NULL, double angle = 0.0, SDL_Point*
center = NULL, SDL_RendererFlip flip = SDL_FLIP_NONE );
    //Renders texture at given point (used for some limitations)
    void renderb( int x, int y, SDL_Rect* clip = NULL, double angle = 0.0, SDL_Point*
center = NULL, SDL_RendererFlip flip = SDL_FLIP_NONE );
```

```
   //Gets image dimensions
   int getWidth();
   int getHeight();

  private:
   //The actual hardware texture
   SDL_Texture* mTexture;

   //Image dimensions
   int mWidth;
   int mHeight;
};
//Starts up SDL and creates window
bool init();

//Loads media
bool loadMedia();

//Frees media and shuts down SDL
void close();


class LButton
{
  public:
   //Initializes internal variables
   LButton();

   //Sets top left position
   void setPosition( int x, int y );

   //Handles mouse event
   bool handleEvent( SDL_Event* e ,int width, int height,int bardetectioncode,int
*menu_bar);


  private:
   //Top left position
   SDL_Point mPosition;

};
//

//Read the saved data from a specific file
void read_savefile(int ith);

//Write the saved data on a specific file
void write_savefile(int ith);

//initialze SDL and loads the media
bool init_and_loading();

//initializes menu buttons' positions
void menu_initialization();

//render the menu
void render_menu(int menu_bar);

//puts invisible rectangular boxes from a file on specific places
void put_walls();

//checks collision between two rectangular shapes (the character and the wall)
bool checkCollision( int x,int y, int w, int h, SDL_Rect b );
```

```
//and many other static recangular shapes(the walls)
bool collisionarrchec(SDL_Rect wall[],int len,int x, int y, int w, int h);

//moves the movable rectangular pic(the character) using keypresses
void movechar();

//start the collections of objects.. by displaing texts and finding if he collected
enough collectables
void missionstart(char x[],char y[],char z[], int endingcount,int xlen , int ylen, int
zlen);

//render the environment and character condionally and sequentially
void render_environment();

//render  outlined box of fixed size
void render_outline(int x, int y, int w,int h);

//render black filled box of fixed size
void render_darkbox(int x, int y, int w,int h);

//render stage numbers at the corner while gameplay
void render_stagefont();

//render the name of the place where we are in
void render_nameoftheplace();

//render the collectable single object such as pringles,boxes ,coffees and modify them
void rendersingleobject(int xtm,int ytm, int number_of_obj);

//render all collectable objects
void render_collectable_objects();

//speak a specific text dynamically putting the letters one after another
void speak(char text[],int tot_line,int startsignal,int xlen,int ylen,bool middle);

//same as speak() , used in in texts(before difficulty pops up) after starting a new
game just for somme limitations of the prev one
void speak_helps(char text[],int tot_line,int startsignal,int xlen,int ylen, bool
middle);

//rendering the time at the corner
void render_time();

//renders and modify everything after the game ends,eg,credits, name, highscores set
void endgame();

//sort the leaderboard according to the new score
void leaderboardreset();

//takes the name of the player at the end
void enterscorername(SDL_Event* e);

//resets progresses,eg,stage,time,character positions,environment
void reset();

//renders and takes the values of difficulty
void difficultyset(bool *quit,bool *esaped);

//render the save files while saving and loading
void render_saves(int *ith,bool *quit);

//checks if it should save or not during save
```

```
bool ashholei(bool *quit);

//renders the help texts
void helpline(bool *quit);
```

The “game.cpp” cpp file:

```
//using gameaction (including gamemedia.cpp and gameinitialization.cpp)
#include "gameaction.cpp"

/*go  to gameinitialization.hpp and read the comments
just above userdefined functions to know their uses*/
void reset()
{
  x_char = 90; y_char = 90; current_bacground = 0;
  stage = 1;


  stagestart = false; extratime = 0;
  playmusic = true;
  nitro = false;
  current_count = 0; totalcollectables = 1;
  for (int i = 0; i < 4; i++)
    memset(collectable_object_present[i], 1, sizeof(collectable_object_present[i]));

  //write_savefile();
  put_walls();
  startTime = SDL_GetTicks();
}
int main(int argc, char* args[])
{

  if (init_and_loading())
  {
    //Main loop flag
    bool quit = false, intro = true;

    //Event handler
    SDL_Event e;
    menu_initialization();
    //While application is running
    int tempmenubar = -1;
    memset(collectable_object_present[0], 1, sizeof(collectable_object_present[0]));
    int menu_bar = -1;
    int tmpp = 1; bool backward = false;
    while (!quit && intro)
    {
      //Handle events on queue
      while (SDL_PollEvent(&e) != 0)
      {
        //User requests quit
        if (e.type == SDL_QUIT)
        {
          quit = true;
        }
        if (e.type == SDL_KEYDOWN && e.key.keysym.sym == SDLK_RETURN)
        {
          intro = false;
        }
      }
      SDL_SetRenderDrawColor(gRenderer, 0xFF, 0xFF, 0xFF, 0x00);
      SDL_RenderClear(gRenderer);
      intropic.render(0, 0);
      intropic.setAlpha(tmpp);
      if (!backward)tmpp++;
      else tmpp--;
      if (tmpp == 255)backward = true;
      //SDL_Delay(5);
      if (tmpp == 0)intro = false;
      SDL_RenderPresent(gRenderer);
    }
    while (!quit)
    {
      //Handle events on queue
      while (SDL_PollEvent(&e) != 0)
      {
        //User requests quit
        if (e.type == SDL_QUIT)
        {
          quit = true;
```

```
        }
        if (checkscorername)
          enterscorername(&e);

        if (check_score)
          if (leaderboard_to_main.handleEvent(&e, 280, 50, -1, &menu_bar) || e.type == SDL_KEYDOWN &&
e.key.keysym.sym == SDLK_ESCAPE)
          {
            Mix_PlayChannel(-1, click, 0);
            SDL_Delay(500);
            check_score = false;
            check_menu = true;
          }
        if (check_menu)
        {
          if (newgamebutton.handleEvent(&e, 310, 65, 0, &menu_bar) || (e.type == SDL_KEYDOWN &&
e.key.keysym.sym == SDLK_RETURN && menu_bar == 0))
          {
            Mix_PlayChannel(-1, click, 0);
            SDL_Delay(500);
            bool esaped = false;
            difficultyset(&quit, &esaped);
            if (!esaped)
            {
              check_menu = false;
              reset();
            }
          }
          else if (loadgamebutton.handleEvent(&e, 330, 65, 1, &menu_bar) || (e.type == SDL_KEYDOWN &&
e.key.keysym.sym == SDLK_RETURN && menu_bar == 1))
          {
            Mix_PlayChannel(-1, click, 0);
            SDL_Delay(500);
            int ithsave = -1;
            render_saves(&ithsave, &quit);
            if (ithsave != -1)
            {
              read_savefile(ithsave);
              check_menu = false;
            }
            put_walls();
            startTime = SDL_GetTicks() - extratime;
            if (Mix_PlayingMusic() == 0 && stage != 4)
              intermissionmusic = false;
            //stage=5;
          }
          else if (scoreboard.handleEvent(&e, 390, 65, 2, &menu_bar) || (e.type == SDL_KEYDOWN &&
e.key.keysym.sym == SDLK_RETURN && menu_bar == 2))
          {
            Mix_PlayChannel(-1, click, 0);
            SDL_Delay(500);
            check_score = true;
            check_menu = false;
          }
          else if (exitbutton.handleEvent(&e, 150, 65, 3, &menu_bar) || (e.type == SDL_KEYDOWN &&
e.key.keysym.sym == SDLK_RETURN && menu_bar == 3))
          {
            Mix_PlayChannel(-1, click, 0);
            SDL_Delay(500);
            quit = true;

          }
          if (e.type == SDL_KEYDOWN && e.key.keysym.sym == SDLK_UP && menu_bar > 0)
            menu_bar--;
          if (e.type == SDL_KEYDOWN && e.key.keysym.sym == SDLK_DOWN && menu_bar < 3)
            menu_bar++;
          if (e.type == SDL_KEYDOWN && e.key.keysym.sym == SDLK_UP && menu_bar < 0)
            menu_bar = 3;
          if (menu_bar >= 0 && tempmenubar != menu_bar)
          {
            Mix_PlayChannel(-1, selected, 0);
            tempmenubar = menu_bar;

          }
        }
```

```
       if (e.type == SDL_KEYDOWN && e.key.keysym.sym == SDLK_h && !endgamecheck && !check_menu &&
!check_score)
       {
         helpline(&quit);
       }
       if (!checpress && !check_score && !checkscorername && !check_menu && !endgamecheck)
       {
         if (savebutton.handleEvent(&e, 30, 30, -1, &menu_bar))
         {

           Mix_PlayChannel(-1, click, 0);
           SDL_Delay(500);
           int ithsave = -1;
           if (!stagestart || easyplay)
           {
             render_saves(&ithsave, &quit);
             extratime = SDL_GetTicks() - startTime;
             if (ithsave != -1)
             {
               write_savefile(ithsave);
               gamesaved = true;
             }
           }
           savepause = true;

         }
         if (e.type == SDL_KEYDOWN && e.key.keysym.sym == SDLK_ESCAPE)
         {
           Mix_PauseMusic();
           bool asholei = ashholei(&quit);
           if (asholei)
           {
             check_menu = true;
             Mix_HaltMusic();
             extratime = SDL_GetTicks() - startTime;
           }
           else
           {
             Mix_PlayChannel(-1, click, 0);
             SDL_Delay(500);
           }
           Mix_ResumeMusic();
         }
       }
       if (e.type == SDL_KEYDOWN && e.key.keysym.sym == SDLK_n && ((stage == 4 && stagestart) || stage
== 5))
         if (nitro)nitro = false;
         else nitro = true;
       if (e.type == SDL_KEYDOWN && e.key.keysym.sym == SDLK_m)
         if (playmusic)playmusic = false;
         else playmusic = true;
       if (stage < 4)
       {
         playmeg = false;
         intermissionmusic = false;
       }
       if (playmusic)
       {
         if (beat)
         {
           Mix_PlayChannel(-1, collectingbeat, 0);
           beat = false;
         }

         //Play the music

         if (intermissionmusic && !playmeg)
           Mix_HaltMusic();
         if (Mix_PlayingMusic() == 0 && !check_menu && !intermissionmusic && !check_score)
           Mix_PlayMusic(ddlc, -1);
         if (Mix_PlayingMusic() == 0 && playmeg && !check_menu && !check_score)
           Mix_PlayMusic(gMusic, -1);

       }
       else
```

```
        Mix_HaltMusic();
      if (endgamecheck)
      {
        Mix_HaltMusic();
        playmeg = false;
        intermissionmusic = false;
      }

    }
    if (endgamecheck)
      endgame();
    else if (check_score)
    {
      render_scoreboard();
    }
    else if (check_menu)
      render_menu(menu_bar);
    else
    {

      //moving character
      movechar();
      render_environment();

    }


   }

  }
  //Free resources and close SDL
  close();
  return 0;
}
```

The “gamemedia.cpp” cpp file:

```
//Using gameinitializtion.hpp
#include "gameinitialization.hpp"

/*                          media part                      */


//Screen dimension constants

const int SCREEN_WIDTH = 999;
const int SCREEN_HEIGHT = 571;

//The window we'll be rendering to
SDL_Window* gWindow = NULL;

//The window renderer
SDL_Renderer* gRenderer = NULL;

TTF_Font *gFont = NULL;

const int WALKING_ANIMATION_FRAMES = 4,BACKGROUND_NO=4;
//character clips
SDL_Rect gSpriteClipsdown[ WALKING_ANIMATION_FRAMES ];
SDL_Rect gSpriteClipsup[ WALKING_ANIMATION_FRAMES ];
SDL_Rect gSpriteClipsleft[ WALKING_ANIMATION_FRAMES ];
SDL_Rect gSpriteClipsright[ WALKING_ANIMATION_FRAMES ];
SDL_Rect savebuttoncliprect;
SDL_Rect sansquad;
SDL_Rect marioquad;
SDL_Rect someonequad;
SDL_Rect papyrusquad;
SDL_Rect choco;
SDL_Rect pringles;
SDL_Rect coffee;
SDL_Rect box;


LTexture intropic;
LTexture savebuttonclip;
LTexture menu;
LTexture menuraw;
LTexture sans;
LTexture mario;
LTexture someone;
LTexture papyrus;
LTexture chocotex;
LTexture pringlestex;
LTexture coffeetex;
LTexture boxtex;

LTexture speech[20];
LTexture stagefont;
LTexture timeshow;
LTexture nitrotext;
LTexture musictext;
LTexture dificultytext[10];

//the whole picture of character frames
LTexture gSpriteSheetTexture;
LTexture gTimeTexture;

//background images
```

```
LTexture gBackgroundTexture[BACKGROUND_NO];

Mix_Music *gMusic = NULL;
Mix_Music *ddlc =NULL;
Mix_Chunk *collectingbeat = NULL;
Mix_Chunk *selected = NULL;
Mix_Chunk *click = NULL;

LTexture::LTexture()
{
  //Initialize
  mTexture = NULL;
  mWidth = 0;
  mHeight = 0;
}
LTexture::~LTexture()
{
  //Deallocate
  free();
}

bool LTexture::loadFromFile( std::string path )
{
  //Get rid of preexisting texture
  free();

  //The final texture
  SDL_Texture* newTexture = NULL;

  //Load image at specified path
  SDL_Surface* loadedSurface = IMG_Load( path.c_str() );
  if( loadedSurface == NULL )
  {
    printf( "Unable to load image %s! SDL_image Error: %s\n", path.c_str(),
IMG_GetError() );
  }
  else
  {
    //Color key image
    //SDL_SetColorKey( loadedSurface, SDL_TRUE, SDL_MapRGB( loadedSurface->format,
0xFF, 0xFF, 0xFF ) );

    //Create texture from surface pixels
         newTexture = SDL_CreateTextureFromSurface( gRenderer, loadedSurface );
    if( newTexture == NULL )
    {
      printf( "Unable to create texture from %s! SDL Error: %s\n", path.c_str(),
SDL_GetError() );
    }
    else
    {
      //Get image dimensions
      mWidth = SCREEN_WIDTH;
      mHeight = SCREEN_HEIGHT;
    }

    //Get rid of old loaded surface
    SDL_FreeSurface( loadedSurface );
  }

  //Return success
  mTexture = newTexture;
  return mTexture != NULL;
```

```
}
bool LTexture::loadFromRenderedText( std::string textureText, SDL_Color textColor )
{
  //Get rid of preexisting texture
  free();

  //Render text surface
  SDL_Surface* textSurface = TTF_RenderText_Solid( gFont, textureText.c_str(),
textColor );
  if( textSurface == NULL )
  {
    printf( "Unable to render text surface! SDL_ttf Error: %s\n", TTF_GetError() );
  }
  else
  {
    //Create texture from surface pixels
         mTexture = SDL_CreateTextureFromSurface( gRenderer, textSurface );
    if( mTexture == NULL )
    {
      printf( "Unable to create texture from rendered text! SDL Error: %s\n",
SDL_GetError() );
    }
    else
    {
      //Get image dimensions
      mWidth = textSurface->w;
      mHeight = textSurface->h;
    }

    //Get rid of old surface
    SDL_FreeSurface( textSurface );
  }

  //Return success
  return mTexture != NULL;
}

void LTexture::free()
{
  //Free texture if it exists
  if( mTexture != NULL )
  {
    SDL_DestroyTexture( mTexture );
    mTexture = NULL;
    mWidth = 0;
    mHeight = 0;
  }
}

void LTexture::setColor( Uint8 red, Uint8 green, Uint8 blue )
{
  //Modulate texture rgb
  SDL_SetTextureColorMod( mTexture, red, green, blue );
}

void LTexture::setBlendMode( SDL_BlendMode blending )
{
  //Set blending function
  SDL_SetTextureBlendMode( mTexture, blending );
}

void LTexture::setAlpha( Uint8 alpha )
{
```

```
  //Modulate texture alpha
  SDL_SetTextureAlphaMod( mTexture, alpha );
}

void LTexture::render( int x, int y, SDL_Rect* clip, double angle, SDL_Point* center,
SDL_RendererFlip flip )
{
  //Set rendering space and render to screen
  SDL_Rect renderQuad = { x, y, mWidth, mHeight };

  //Set clip rendering dimensions
  if( clip != NULL )
  {
    renderQuad.w = 30;
    renderQuad.h = 30;
  }

  //Render to screen
  SDL_RenderCopyEx( gRenderer, mTexture, clip, &renderQuad, angle, center, flip );
}
void LTexture::renderb( int x, int y, SDL_Rect* clip,double angle, SDL_Point* center,
SDL_RendererFlip flip )
{
  //Set rendering space and render to screen
  SDL_Rect renderQuad = { x, y, mWidth, mHeight };

  //Set clip rendering dimensions
  if( clip != NULL )
  {
    renderQuad.w = 150;
    renderQuad.h = 150;
  }

  //Render to screen
  SDL_RenderCopyEx( gRenderer, mTexture, clip, &renderQuad, angle, center, flip );
}
int LTexture::getWidth()
{
  return mWidth;
}

int LTexture::getHeight()
{
  return mHeight;
}
bool init()
{
  //Initialization flag
  bool success = true;

  //Initialize SDL
  if( SDL_Init( SDL_INIT_VIDEO | SDL_INIT_AUDIO ) < 0 )
  {
    printf( "SDL could not initialize! SDL Error: %s\n", SDL_GetError() );
    success = false;
  }
  else
  {
    //Set texture filtering to linear
    if( !SDL_SetHint( SDL_HINT_RENDER_SCALE_QUALITY, "1" ) )
    {
      printf( "Warning: Linear texture filtering not enabled!" );
    }
```

```
   //Create window
   gWindow = SDL_CreateWindow( "Adventure of chunchin", SDL_WINDOWPOS_UNDEFINED,
SDL_WINDOWPOS_UNDEFINED, SCREEN_WIDTH, SCREEN_HEIGHT, SDL_WINDOW_SHOWN );
   if( gWindow == NULL )
   {
     printf( "Window could not be created! SDL Error: %s\n", SDL_GetError() );
     success = false;
   }
   else
   {
     //Create vsynced renderer for window
     gRenderer = SDL_CreateRenderer( gWindow, -1, SDL_RENDERER_ACCELERATED |
SDL_RENDERER_PRESENTVSYNC );
     if( gRenderer == NULL )
     {
       printf( "Renderer could not be created! SDL Error: %s\n", SDL_GetError() );
       success = false;
     }
     else
     {
       //Initialize renderer color
       SDL_SetRenderDrawColor( gRenderer, 0xFF, 0xFF, 0xFF, 0xFF );

       //Initialize PNG loading
       int imgFlags = IMG_INIT_PNG;
       if( !( IMG_Init( imgFlags ) & imgFlags ) )
       {
         printf( "SDL_image could not initialize! SDL_image Error: %s\n", IMG_GetError()
);
         success = false;
       }
        //Initialize SDL_mixer
       if( Mix_OpenAudio( 44100, MIX_DEFAULT_FORMAT, 2, 2048 ) < 0 )
       {
         printf( "SDL_mixer could not initialize! SDL_mixer Error: %s\n", Mix_GetError()
);
         success = false;
       }
        //Initialize SDL_ttf
       if( TTF_Init() == -1 )
       {
         printf( "SDL_ttf could not initialize! SDL_ttf Error: %s\n", TTF_GetError() );
         success = false;
       }
     }
   }
  }

  return success;
}
bool loadMedia()
{
  //Loading success flag
  bool success = true;


  //Load sprite sheet texture
  if( !gSpriteSheetTexture.loadFromFile( "media/charframes.png" ) )
  {
    printf( "Failed to load walking animation texture!\n" );
    success = false;
  }
```

```
else
{
  //Set sprite clips
  for(int i=0;i<4;i++)
  {
    gSpriteClipsdown[ i ].x = i*233;
    gSpriteClipsdown[ i ].y = 0;
    gSpriteClipsdown[ i ].w = 233;
    gSpriteClipsdown[ i ].h = 283;
  }
  for(int i=0;i<4;i++)
  {
    if(i<2)gSpriteClipsup[ i ].x = i*233;
    else if (i==2)gSpriteClipsup[ i ].x = 0;
    else gSpriteClipsup[ i ].x = 2*233;
    gSpriteClipsup[ i ].y = 283;
    gSpriteClipsup[ i ].w = 233;
    gSpriteClipsup[ i ].h = 283;
  }
  for(int i=0;i<4;i++)
  {
    gSpriteClipsleft[ i ].x = i*233;
    gSpriteClipsleft[ i ].y = 283*2;
    gSpriteClipsleft[ i ].w = 233;
    gSpriteClipsleft[ i ].h = 283;
  }
  for(int i=0;i<4;i++)
  {
    gSpriteClipsright[ i ].x = i*233;
    gSpriteClipsright[ i ].y = 283*3;
    gSpriteClipsright[ i ].w = 233;
    gSpriteClipsright[ i ].h = 283;
  }
}
//load first background picture
if( !gBackgroundTexture[0].loadFromFile( "media/background1.png" ) )
{
  printf( "Failed to load background texture image!\n" );
  success = false;
}
//loads second background picture
if( !gBackgroundTexture[1].loadFromFile( "media/background2.png" ) )
{
  printf( "Failed to load background texture image!\n" );
  success = false;
}
if( !gBackgroundTexture[2].loadFromFile( "media/background3.png" ) )
{
  printf( "Failed to load background texture image!\n" );
  success = false;
}
if( !gBackgroundTexture[3].loadFromFile( "media/background4.png" ) )
{
  printf( "Failed to load background texture image!\n" );
  success = false;
}
if( !menu.loadFromFile( "media/menu.png" ) )
{
  printf( "Failed to load walking animation texture!\n" );
  success = false;
}
if( !intropic.loadFromFile( "media/intro.png" ) )
{
```

```
  printf( "Failed to load walking animation texture!\n" );
  success = false;
}
if( !menuraw.loadFromFile( "media/menuraw.png" ) )
{
  printf( "Failed to load walking animation texture!\n" );
  success = false;
}
if( !savebuttonclip.loadFromFile( "media/savebutton.png" ) )
   {
       printf( "Failed to load walking animation texture!\n" );
       success = false;
   }
   savebuttoncliprect.x=0;
   savebuttoncliprect.y=0;
   savebuttoncliprect.w=375;
   savebuttoncliprect.h=300;
if( !sans.loadFromFile( "media/sans.png" ) )
   {
       printf( "Failed to load walking animation texture!\n" );
       success = false;
   }
   sansquad.x=100;
   sansquad.y=0;
   sansquad.w=600;
   sansquad.h=450;
   if( !mario.loadFromFile( "media/mario.png" ) )
   {
       printf( "Failed to load walking animation texture!\n" );
       success = false;
   }
   marioquad.x=0;
   marioquad.y=0;
   marioquad.w=220;
   marioquad.h=337;
   if( !someone.loadFromFile( "media/someone.png" ) )
   {
       printf( "Failed to load walking animation texture!\n" );
       success = false;
   }
   someonequad.x=300;
   someonequad.y=0;
   someonequad.w=600;
   someonequad.h=900;
   if( !papyrus.loadFromFile( "media/papyrus.png" ) )
   {
       printf( "Failed to load walking animation texture!\n" );
       success = false;
   }
   papyrusquad.x=0;
   papyrusquad.y=0;
   papyrusquad.w=452;
   papyrusquad.h=678;
if( !boxtex.loadFromFile( "media/box.png" ) )
{
  printf( "Failed to load walking animation texture!\n" );
  success = false;
}
box.x=0;
box.y=0;
box.h=500;
box.w=500;
if( !chocotex.loadFromFile( "media/choco.png" ) )
```

```
  {
    printf( "Failed to load walking animation texture!\n" );
    success = false;
  }
  choco.x=0;
  choco.y=0;
  choco.h=554;
  choco.w=554;
  if( !pringlestex.loadFromFile( "media/pringles.png" ) )
  {
    printf( "Failed to load walking animation texture!\n" );
    success = false;
  }
  pringles.x=0;
  pringles.y=0;
  pringles.h=554;
  pringles.w=554;
  if( !coffeetex.loadFromFile( "media/coffee.png" ) )
  {
    printf( "Failed to load walking animation texture!\n" );
    success = false;
  }
  coffee.x=0;
  coffee.y=0;
  coffee.h=300;
  coffee.w=300;
  gFont = TTF_OpenFont( "media/lazy.ttf", 30 );
  if( gFont == NULL )
  {
    printf( "Failed to load lazy font! SDL_ttf Error: %s\n", TTF_GetError() );
    success = false;
  }
  gMusic = Mix_LoadMUS( "media/Undertaleaudio.mp3" );
  if( gMusic == NULL )
  {
    printf( "Failed to load beat music! SDL_mixer Error: %s\n", Mix_GetError() );
    success = false;
  }
  ddlc = Mix_LoadMUS( "media/ddlc.wav" );
  if( ddlc == NULL )
  {
    printf( "Failed to load beat music! SDL_mixer Error: %s\n", Mix_GetError() );
    success = false;
  }
  collectingbeat = Mix_LoadWAV( "media/beat.wav" );
  if( collectingbeat == NULL )
  {
    printf( "Failed to load high sound effect! SDL_mixer Error: %s\n", Mix_GetError()
);
    success = false;
  }
  selected = Mix_LoadWAV( "media/select.wav" );
  if( selected == NULL )
  {
    printf( "Failed to load high sound effect! SDL_mixer Error: %s\n", Mix_GetError()
);
    success = false;
  }
  click = Mix_LoadWAV( "media/click.wav" );
  if( click == NULL )
  {
    printf( "Failed to load high sound effect! SDL_mixer Error: %s\n", Mix_GetError()
);
```

```
    success = false;
  }
  return success;


}
void close()
{
  gBackgroundTexture[0].free();
  gBackgroundTexture[1].free();
     menu.free();
     savebuttonclip.free();
  gSpriteSheetTexture.free();

  Mix_FreeMusic( gMusic );
     gMusic = NULL;
  //Destroy window
  SDL_DestroyRenderer( gRenderer );
  SDL_DestroyWindow( gWindow );
  gWindow = NULL;
  gRenderer = NULL;

  //Quit SDL subsystems
  Mix_Quit();
  TTF_Quit();
  IMG_Quit();
  SDL_Quit();
}

/*                          Button part                          */









//int menu_bar=-1;

//Buttons objects
LButton exitbutton;
LButton loadgamebutton;
LButton newgamebutton;
LButton scoreboard;
LButton savebutton;
LButton leaderboard_to_main;
LButton easybutton;
LButton hardbutton;

LButton::LButton()
{
  mPosition.x = 0;
  mPosition.y = 0;

}

void LButton::setPosition( int xpos, int ypos )
{
  mPosition.x = xpos;
  mPosition.y = ypos;
}
```

```
bool LButton::handleEvent( SDL_Event* e ,int width , int height,int
bardetectioncode,int *menu_bar)
{
  //If mouse event happened
  if( e->type == SDL_MOUSEMOTION || e->type == SDL_MOUSEBUTTONDOWN || e->type ==
SDL_MOUSEBUTTONUP )
  {
    //Get mouse position
    int xpos, ypos;
    SDL_GetMouseState( &xpos, &ypos );

    //Check if mouse is in button
    bool inside = true;

    //Mouse is left of the button
    if( xpos < mPosition.x )
    {
      inside = false;
    }
    //Mouse is right of the button
    else if( xpos > mPosition.x + width )
    {
      inside = false;
    }
    //Mouse above the button
    else if( ypos < mPosition.y )
    {
      inside = false;
    }
    //Mouse below the button
    else if( ypos > mPosition.y + height )
    {
      inside = false;
    }

    //Mouse is outside button
    if( !inside )
    {
             return false;
    }
    //Mouse is inside button
    if(e->type == SDL_MOUSEBUTTONDOWN)
    {
      return true;
    }
    else if(e->type == SDL_MOUSEMOTION)
    {
      *menu_bar=bardetectioncode;
      return false;
    }
    else
      return false;
  }
  else
    return false;
}
```

The “gameaction.cpp” cpp file:

```
//Using gamemedia.cpp(including gameinitialization.cpp)
#include "gamemedia.cpp"


//Character positions and current char_frame number
int char_frame,x_char,y_char;

int checpress,stage,totalcollectables,current_count;

bool stagestart=false;
bool check_menu=true,check_score=false,endgamecheck=false;
bool pause=false;
bool pause_for_blocking_stage2=false;
bool pause_for_blocking_stage3=false;
bool pause_for_blocking_stage4=false;
bool savepause=false;
bool playmeg=false;
bool beat=false;
bool nitro=false;
bool fastrunning=false;
bool intermissionmusic=false;
bool playmusic=true;
bool easyplay=true;
bool gamesaved=false;
int startTime,extratime;
std::stringstream timeText,positiontext;

//number of walls on specific backgrround
//and currentbackground indicator
int len[10],current_bacground;

//rectangular boxes which is used to make collisions
SDL_Rect wall[10][100];

bool collectable_object_present[5][20];
//Current rendered texture
//and the box through which we change places
SDL_Rect* currentClip,backgroundchange,collectable_object;

void read_savefile(int ith)
{
    if(ith==1)
    {
     FILE *fp;
     fp = fopen("saveddoc/savefile.txt","r");
     fscanf(fp,"%d%d%d",&x_char,&y_char,&current_bacground);
      fscanf(fp,"%d%d%d",&stage,&stagestart,&extratime);
      fscanf(fp,"%d%d",&current_count,&totalcollectables);
      fscanf(fp,"%d",&easyplay);
      for(int james=0;james<4;james++)
          for(int khaled=0;khaled<20;khaled++)
              fscanf(fp,"%d",&collectable_object_present[james][khaled]);
      fclose(fp);
    }
    else if(ith==2)
    {
     FILE *fp;
     fp = fopen("saveddoc/savefile2.txt","r");
     fscanf(fp,"%d%d%d",&x_char,&y_char,&current_bacground);
      fscanf(fp,"%d%d%d",&stage,&stagestart,&extratime);
      fscanf(fp,"%d%d",&current_count,&totalcollectables);
      fscanf(fp,"%d",&easyplay);
```

```
      for(int james=0;james<4;james++)
          for(int khaled=0;khaled<20;khaled++)
              fscanf(fp,"%d",&collectable_object_present[james][khaled]);
      fclose(fp);
     }
     else if(ith==3)
     {
     FILE *fp;
     fp = fopen("saveddoc/savefile3.txt","r");
      fscanf(fp,"%d%d%d",&x_char,&y_char,&current_bacground);
      fscanf(fp,"%d%d%d",&stage,&stagestart,&extratime);
      fscanf(fp,"%d%d",&current_count,&totalcollectables);
      fscanf(fp,"%d",&easyplay);
      for(int james=0;james<4;james++)
          for(int khaled=0;khaled<20;khaled++)
              fscanf(fp,"%d",&collectable_object_present[james][khaled]);
      fclose(fp);
  }
}
void write_savefile(int ith)
{
     if(ith==1)
     {
     FILE *fp;
     fp = fopen("saveddoc/savefile.txt","w");
     fprintf(fp, "%d %d %d\n",x_char,y_char,current_bacground );
      fprintf(fp, "%d %d %d\n",stage,stagestart,extratime);
      fprintf(fp, "%d %d\n",current_count,totalcollectables );
      fprintf(fp, "%d\n",easyplay );
      for(int james=0;james<4;james++)
      {
          for(int khaled=0;khaled<20;khaled++)
              fprintf(fp,"%d ",collectable_object_present[james][khaled]);
          fprintf(fp, "\n");
      }
      fclose(fp);
     }
     else if(ith==2)
     { FILE *fp;
     fp = fopen("saveddoc/savefile2.txt","w");
     fprintf(fp, "%d %d %d\n",x_char,y_char,current_bacground );
      fprintf(fp, "%d %d %d\n",stage,stagestart,extratime);
      fprintf(fp, "%d %d\n",current_count,totalcollectables );
      fprintf(fp, "%d\n",easyplay );
      for(int james=0;james<4;james++)
      {
          for(int khaled=0;khaled<20;khaled++)
              fprintf(fp,"%d ",collectable_object_present[james][khaled]);
          fprintf(fp, "\n");
      }
      fclose(fp);
     }
     else if(ith==3)
     {
     FILE *fp;
     fp = fopen("saveddoc/savefile3.txt","w");
      fprintf(fp, "%d %d %d\n",x_char,y_char,current_bacground );
      fprintf(fp, "%d %d %d\n",stage,stagestart,extratime);
      fprintf(fp, "%d %d\n",current_count,totalcollectables );
      fprintf(fp, "%d\n",easyplay );
      for(int james=0;james<4;james++)
      {
          for(int khaled=0;khaled<20;khaled++)
```

```
                fprintf(fp,"%d ",collectable_object_present[james][khaled]);
            fprintf(fp, "\n");
         }
         fclose(fp);
    }
}
bool init_and_loading()
{
    //initialization
    if( !init() )
    {
        printf( "Failed to initialize!\n" );
        return false;
    }
    else
    {
        //Load media
        if( !loadMedia() )
        {
            printf( "Failed to load media!\n" );
            return false;
        }
        else
            return true;
    }
}
void menu_initialization()
{

    newgamebutton.setPosition(100,150);
    loadgamebutton.setPosition(100,235);
    exitbutton.setPosition(100,410);
    savebutton.setPosition(920,500);
    scoreboard.setPosition(100,320);
    leaderboard_to_main.setPosition(695,495);
    easybutton.setPosition(430,150);
    hardbutton.setPosition(430,235);

}
void render_menu(int menu_bar)
{
    SDL_SetRenderDrawColor( gRenderer, 0xFF, 0xFF, 0xFF, 0xFF );
    SDL_RenderClear( gRenderer );

    menu.render(0,0);
    //speech.render(100,100);
    SDL_Rect newgame = { 100, 150, 310, 65 };
    SDL_Rect loadgame = { 100, 235, 330, 65 };
    SDL_Rect scoreboard ={ 100, 320, 390, 65 };
    SDL_Rect exit = { 100, 410, 150, 65 };

    SDL_SetRenderDrawColor( gRenderer, 0x00, 0x00, 0x00, 0x05 );

    if(menu_bar==0){SDL_Delay(50);SDL_RenderDrawRect( gRenderer, &newgame );}
    else if(menu_bar==1){SDL_Delay(50);SDL_RenderDrawRect( gRenderer, &loadgame );}
    else if(menu_bar==3){SDL_Delay(50);SDL_RenderDrawRect( gRenderer, &exit );}
    else if(menu_bar==2) {SDL_Delay(50);SDL_RenderDrawRect( gRenderer, &scoreboard );}
    //Update screen
    SDL_RenderPresent( gRenderer );
    SDL_RenderClear(gRenderer);
}

void put_walls()
```

```
{
    //number of wall in first background
    len[0]=15;
    //number of wall in second background
    len[1]=18;
    //number of wall in third background
    len[2]=22;
    //number of wall in fourth background
    len[3]=18;
    //reading wall positions from file
    FILE *wall_input = fopen("saveddoc/savedwall.txt","r");
    for(int i=0;i<BACKGROUND_NO;i++)
        for(int j=0;j<len[i];j++)
        {
            fscanf(wall_input,"%d",&wall[i][j].x);
            fscanf(wall_input,"%d",&wall[i][j].y);
            fscanf(wall_input,"%d",&wall[i][j].w);
            fscanf(wall_input,"%d",&wall[i][j].h);
        }

    if(stage!=1)
    {
        wall[0][14].x=0;
        wall[0][14].y=0;
        wall[0][14].w=0;
        wall[0][14].h=0;
    }
    if(stage>3)
    {
        wall[2][21].x=0;
        wall[2][21].y=0;
        wall[2][21].w=0;
        wall[2][21].h=0;
    }
    if(stage>4)
    {
        wall[1][17].x=0;
        wall[1][17].y=0;
        wall[1][17].w=0;
        wall[1][17].h=0;
    }

}

bool checkCollision( int x,int y, int w, int h, SDL_Rect b )
{
    //The sides of the rectangles
    int leftA, leftB;
    int rightA, rightB;
    int topA, topB;
    int bottomA, bottomB;

    //Calculate the sides of rect A
    leftA = x;
    rightA = x + w;
    topA = y;
    bottomA =y + h;

    //Calculate the sides of rect B
    leftB = b.x;
    rightB = b.x + b.w;
    topB = b.y;
    bottomB = b.y + b.h;
```

```
    //If any of the sides from A are outside of B
    if( bottomA <= topB )
    {

        return false;
    }

    if( topA >= bottomB )
    {

        return false;

    }

    if( rightA <= leftB )
    {

        return false;
    }

    if( leftA >= rightB )
    {
        return false;
    }

    //If none of the sides from A are outside B
    return true;
}
bool collisionarrchec(SDL_Rect wall[],int len,int x, int y, int w, int h)
{
    //checkcollisions of multiple walls all together
    for(int i=0;i<len;i++)
        if(checkCollision(x,y,w,h,wall[i]))
            return false;
    return true;


}

void movechar()
{
    if(!pause && !pause_for_blocking_stage2 && !pause_for_blocking_stage3 &&
!pause_for_blocking_stage4 && !savepause)
    {
    //Set texture based on current keystate

     const Uint8* currentKeyStates = SDL_GetKeyboardState( NULL );

    checpress=0;
     //check key presses
     if( currentKeyStates[ SDL_SCANCODE_UP ] )
     {
         //printf("%d %d\n",x_char,y_char );//(used for knowing the position of
certain places)

         //fix the current character char_frame accordingly
         currentClip = &gSpriteClipsup[ char_frame ];
         //determines if the character is moving and changes the position accordingly
         y_char=y_char-5;checpress=1;
         //checks incoming collision and if there is then backs away
```

```
if(!collisionarrchec(wall[current_bacground],len[current_bacground],x_char,y_char,30,3
0))
              y_char=y_char+5;
      }
      if( currentKeyStates[ SDL_SCANCODE_DOWN ] )
      {
          currentClip = &gSpriteClipsdown[ char_frame ];
          checpress=1;y_char=y_char+5;

if(!collisionarrchec(wall[current_bacground],len[current_bacground],x_char,y_char,30,3
0))
              y_char=y_char-5;
      }
      if( currentKeyStates[ SDL_SCANCODE_LEFT ] )
      {
          currentClip = &gSpriteClipsleft[ char_frame ];
          checpress=1;x_char=x_char-5;

if(!collisionarrchec(wall[current_bacground],len[current_bacground],x_char,y_char,30,3
0))
              x_char=x_char+5;

      }
      if( currentKeyStates[ SDL_SCANCODE_RIGHT ] )
      {
          currentClip = &gSpriteClipsright[ char_frame ];
          checpress=1;x_char=x_char+5;

if(!collisionarrchec(wall[current_bacground],len[current_bacground],x_char,y_char,30,3
0))
              x_char=x_char-5;
      }
         if(currentKeyStates[SDL_SCANCODE_LSHIFT])
         {
             checpress=1;
             fastrunning = true;
         }
         if( currentKeyStates[ SDL_SCANCODE_RETURN ] )
      {
          checpress=1;
          //if the character is at certain positions(where the black box is rendered)
          //the key press change background and position of characters

          if(current_bacground==1 && x_char>=905 && x_char<=915 && y_char>=250 &&
y_char<=260)
          {

                current_bacground=0;
              x_char=825;y_char=155;
          }
          else if(current_bacground==0 && x_char>=810 && x_char <=820 && y_char>=150
&& y_char<= 160)
          {
              if(stage<4)
                {
                   pause_for_blocking_stage4=true;
                }
                else
                {
                    current_bacground=1;
                  x_char=920;y_char=255;
```

```
                }
            }
         else if(current_bacground==0 && x_char>=815 && x_char <=825 && y_char>=385
&& y_char<= 395)
         {
             if(stage<3)
                    pause_for_blocking_stage3=true;
               else
               {
                   current_bacground=2;
                x_char=945;y_char=265;
            }
           }
         else if(current_bacground==2 && x_char>=930 && x_char <=940 && y_char>=260
&& y_char<= 270)
         {
             current_bacground=0;
             x_char=830;y_char=390;
         }
         else if(current_bacground==0 && x_char>=150 && x_char <=160 && y_char>=145
&& y_char<= 155)
         {
             if(stage<2)
                    pause_for_blocking_stage2=true;
               else
               {
                   current_bacground=3;
                x_char=120;y_char=420;
            }
           }
         else if(current_bacground==3 && x_char>=105 && x_char <=115 && y_char>=415
&& y_char<= 425)
         {
                current_bacground=0;
             x_char=165;y_char=150;
         }
        else if(current_bacground==1 && x_char>=110 && x_char <=120 && y_char>=390 &&
y_char<= 400 && stage==4)
       {
             pause=true; //printf("1\n");
       }
            else if(current_bacground==0 && x_char>=605 && x_char <=615 && y_char>=250
&& y_char<= 260 && stage==1)
            {
                pause=true; //printf("1\n");
            }
            else if(current_bacground==3 && x_char>=585 && x_char <=595 && y_char>=415
&& y_char<= 425 && stage==2)
            {
                pause=true; //printf("1\n");
            }
         else if(current_bacground==2 && x_char>=730 && x_char <=740 && y_char>=120
&& y_char<= 130 && stage==3)
            {
                pause=true; //printf("1\n");
            }
            else if(current_bacground==3 && x_char>=795 && x_char <=805 && y_char>=495
&& y_char<= 505 && stage==5)
                endgamecheck=true;
        }
     if(!checpress)
     {
         //if no key press the character stands still facing forward
```

```
            currentClip = &gSpriteClipsdown[ 0 ];
        }
        //changes the char_frames if key are pressed in determined direction
        else char_frame++;
    }
}
void missionstart(char x[],char y[],char z[], int endingcount,int xlen , int ylen, int
zlen)
{
    if(!stagestart )
    {
        speak(x,xlen,1,150,560,false);
        totalcollectables=endingcount;
        current_count=0;
    }
    else if(stagestart && current_count<totalcollectables )
        speak(y,ylen,0,150,560,false);
    else if(stagestart && current_count>=totalcollectables)
        speak(z,zlen,-1,150,560,false);
}
void render_environment()
{
    if(stage==4 && current_bacground==1)playmeg=true;
    if(stage==4 && current_bacground==0)intermissionmusic=true;
    //renders the color for black box where you can change places
    SDL_SetRenderDrawColor( gRenderer, 0x00, 0x00, 0x00, 0xFF  );
    //clears grenderer
    SDL_RenderClear( gRenderer );
    //renders the current background
    gBackgroundTexture[current_bacground].render( 0, 0 );


    SDL_Color musictextcolor={255,255,255};

    if(current_bacground==0)
    {
        if(stage==1)
        {
            mario.render(640,250,&marioquad);
            render_outline(610,255,30,30);
            if(pause)
            {
                mario.renderb(5,431,&marioquad);
                char textini[200];
                if(easyplay)
                {
                    char tmpeorary[200]="Hello. I am Mario. You must have heard of me.
\n But that doesnt matter now.I need help. \n I lost some boxes on the way.\nWould you
help me collect atleast 3 boxes";
                    strcpy(textini,tmpeorary);
                }
                else
                {

                    char tmpeorary[200]="Hello. I am Mario. You must have heard of me.
\n But that doesnt matter now.I need help. \n I lost some boxes on the way.\nWould you
help me collect atleast 5 boxes";
                    strcpy(textini,tmpeorary);
                }
                char textmid[50]="Please bring me more of them";
                char textfinal[100]="Thanks. Here the key to opening the Liberty
City";
                int numberbasedoneasy;
```

```
                if(easyplay)numberbasedoneasy=3;
                else numberbasedoneasy=5;
                missionstart(textini,textmid,textfinal,numberbasedoneasy,4,3,3);

            }
        }

        if(pause_for_blocking_stage2)
        {
            char x[100]="You cannot enter this place(Liberty City) yet. \n Complete
stage 1 to continue.";
            speak(x,2,0,150,560,true);
        }
        else if(pause_for_blocking_stage3)
        {
            char x[100]="You cannot enter this place(Asgard) yet. \n Complete stage 2
to continue.";
            speak(x,2,0,150,560,true);
        }
        else if(pause_for_blocking_stage4)
        {
            char x[100]="You cannot enter this place(Deadman Wonderland) yet. \n
Complete stage 3 to continue.";
            speak(x,2,0,150,560,true);
        }
        render_outline(815,155,30,30);
        render_outline(820,390,30,30);
        render_outline(155,150,30,30);

    }
    else if(current_bacground==1)
    {

        if(stage==4)
        {
            sans.render( 80, 390, &sansquad);
            render_outline(110,390,30,30);

            if(pause)
            {
                sans.renderb(5,431,&sansquad);
                char textini[200]="I have been waiting for you..Welcome.I am sans.. \n
Your final task is to collect me all the pringles here\n And before you go, here, a
nitro shoe,a prize \nfor coming this far";
                char textmid[50]="You are not finished mate";
                char textfinal[100]="Congratulations.Here!The key to the exit in
Liberty City";
                missionstart(textini,textmid,textfinal,16,4,3,3);
            }
        }

        render_outline(910,255,30,30);


    }
    else if(current_bacground==2)
    {
        render_outline(935,265,30,30);
        if(stage==3)
        {
            papyrus.render(735,95,&papyrusquad);
            render_outline(735,125,30,30);
            if(pause)
```

```
            {
                papyrus.renderb(5,431,&papyrusquad);
                char textini[100];
                if(easyplay)
                {
                    char tmpeorary[100]="Hey. I am Papyrus.I want chocolates. \n So
give me 3 chocolates bar";
                    strcpy(textini,tmpeorary);
                }
                else
                {

                    char tmpeorary[100]="Hey. I am Papyrus.I want chocolates. \n So
give me 5 chocolates bar";
                    strcpy(textini,tmpeorary);
                }
                char textmid[50]="More More I need more chocolates";
                char textfinal[100]="You have done your job well.The Deadman
Wonderland is \nnow open";
                int numberbasedoneasy;
                if(easyplay)numberbasedoneasy=3;
                else numberbasedoneasy=5;
                missionstart(textini,textmid,textfinal,numberbasedoneasy,3,3,3);
            }

        }
    }
    else if(current_bacground==3)
    {
        if(stage==2)
        {
            someone.render(590,395,&someonequad);
            render_outline(590,420,30,30);
            if(pause)
            {
                someone.renderb(40,431,&someonequad);
                char textini[100]="Hello. I am ugghs..It doesnt matter. I am tired \n
Can you get me some atleast 4 cup of coffees?";
                if(easyplay)
                {
                    char tmpeorary[100]="Hello. I am ugghs..It doesnt matter. I am
tired \n Can you get me some atleast 4 cup of coffees?";
                    strcpy(textini,tmpeorary);
                }
                else
                {

                    char tmpeorary[100]="Hello. I am ugghs..It doesnt matter. I am
tired \n Can you get me some atleast 5 cup of coffees?";
                    strcpy(textini,tmpeorary);
                }
                char textmid[50]="More ! I Need More Coffees";
                char textfinal[100]="Thank you very much. Here the key to opening the
Asgard";
                int numberbasedoneasy;
                if(easyplay)numberbasedoneasy=4;
                else numberbasedoneasy=5;
                missionstart(textini,textmid,textfinal,numberbasedoneasy,3,3,3);
            }
        }
        else if(stage==5)
        {
            render_outline(800,500,30,30);
```

```
        }
        render_outline(110,420,30,30);


    }

    //renders the character char_frames
    gSpriteSheetTexture.render( x_char, y_char, currentClip);
    savebuttonclip.render(920,500,&savebuttoncliprect);


    if(savepause)
    {
        if(!easyplay && stagestart)
        {
            char saved[30]="Game cannot be Saved Now";
            speak(saved,1,0,150,250,true);
        }
        else if(!gamesaved)
        {
            char saved[30]="Game was not Saved";
            speak(saved,1,0,150,250,true);
        }
        else
        {
            char saved[30]="Game Saved";
            speak(saved,1,0,150,250,true);
        }
    }
    if(stagestart)render_collectable_objects();
    render_stagefont();
    render_nameoftheplace();
    render_time();

    if(easyplay)
        dificultytext[0].loadFromRenderedText("Mode:Easy",musictextcolor);
    else
        dificultytext[0].loadFromRenderedText("Mode:Hard",musictextcolor);
    render_darkbox(838,538,dificultytext[0].getWidth()+2,40);
    dificultytext[0].render(840,540);

    if(playmusic)musictext.loadFromRenderedText("Music:ON",musictextcolor);
    else musictext.loadFromRenderedText("Music:OFF",musictextcolor);
    render_darkbox(8,3,musictext.getWidth()+2,40);
    musictext.render(10,5);
    if((stage==4 && stagestart) || stage==5)
    {
        SDL_Color nitrotextcolor={255,255,255};
        if(nitro)nitrotext.loadFromRenderedText("Nitro:ON",nitrotextcolor);
        else nitrotext.loadFromRenderedText("Nitro:OFF",nitrotextcolor);
        render_darkbox(8,38,nitrotext.getWidth()+2,40);
        nitrotext.render(10,40);
    }
    //Update screen
    SDL_RenderPresent( gRenderer );
    //changes the char_frame number of the character
    if( char_frame >= WALKING_ANIMATION_FRAMES )
    {
        char_frame = 0;
    }
    if(pause || pause_for_blocking_stage2 || pause_for_blocking_stage3 ||
pause_for_blocking_stage4)
        SDL_Delay(30);
```

```
    else
    {
        if(nitro)SDL_Delay(10);
        else
            if(fastrunning) SDL_Delay(30);
            else SDL_Delay(70);
        fastrunning=false;
    }
    //printf("%d %d\n",x,y );
}
void render_outline(int x, int y, int w,int h)
{
    SDL_SetRenderDrawColor(gRenderer,0x00,0x00,0x00,0xFF);
    backgroundchange.x=x;
    backgroundchange.y=y;
    backgroundchange.w=w;
    backgroundchange.h=h;
    SDL_RenderDrawRect( gRenderer, &backgroundchange );

}
void render_darkbox(int x, int y, int w,int h)
{
    SDL_SetRenderDrawColor(gRenderer,0x00,0x00,0x00,0xFF);
    backgroundchange.x=x;
    backgroundchange.y=y;
    backgroundchange.w=w;
    backgroundchange.h=h;
    SDL_RenderFillRect( gRenderer, &backgroundchange );

}
void rendersingleobject(int xtm,int ytm, int number_of_obj)
{

        if(collectable_object_present[stage-1][number_of_obj])
        {
            if(stage==3)chocotex.render(xtm,ytm,&choco);
            else if(stage==4)pringlestex.render(xtm,ytm,&pringles);
            else if(stage==2)coffeetex.render(xtm,ytm,&coffee);
            else if(stage==1)boxtex.render(xtm,ytm,&box);

        }
        if(x_char>=xtm-20 && x_char<=xtm+20 && y_char>=ytm-20 && y_char<=ytm+20)
        {
            if(collectable_object_present[stage-1][number_of_obj])
            {
                current_count++;
                Mix_PlayChannel( -1, collectingbeat, 0 );
            }
            collectable_object_present[stage-1][number_of_obj]=false;
        }

}
void render_stagefont()
{
    SDL_Color textColor = { 255, 255, 255 };
    if(stage==1)
        stagefont.loadFromRenderedText( "Stage 1", textColor );
    else if(stage==2)
        stagefont.loadFromRenderedText( "Stage 2", textColor );
    else if(stage==3)
        stagefont.loadFromRenderedText( "Stage 3", textColor );
    else if(stage==4)
        stagefont.loadFromRenderedText("Stage 4",textColor);
```

```
    else if(stage==5)
        stagefont.loadFromRenderedText("Stage 5",textColor);
    render_darkbox(878,3,100,40);
    stagefont.render(880,5);
}
void render_nameoftheplace()
{
    int x=0;
    SDL_Color textColor = { 255, 255, 255 };
    if(current_bacground==0)
    {
        stagefont.loadFromRenderedText( "Land of oppurtunity", textColor );
    }
    else if(current_bacground==1)
    {
        stagefont.loadFromRenderedText( "Deadman wonderland", textColor );
    }
    else if(current_bacground==2)
    {
        stagefont.loadFromRenderedText( "Asgard", textColor );x=150;
    }
    else if(current_bacground==3)
    {
        stagefont.loadFromRenderedText("Liberty City",textColor);x=100;
    }
    render_darkbox(295+x,8,stagefont.getWidth()+10,stagefont.getHeight()+3);
    stagefont.render(300+x,10);
}

void render_collectable_objects()
{
    int cc=0;
    if(stage==4)
    {
        if(current_bacground==0)
        {
            rendersingleobject(200,200,cc++);
            rendersingleobject(265,445,cc++);
            rendersingleobject(755,320,cc++);
            rendersingleobject(320,200,cc++);
            rendersingleobject(610,140,cc++);
            rendersingleobject(750,90,cc++);

        }
        else if(current_bacground==1)
        {
            cc=6;
            rendersingleobject(440,145,cc++);
            rendersingleobject(450,500,cc++);
            rendersingleobject(580,455,cc++);
            rendersingleobject(730,160,cc++);
        }
        else if(current_bacground==3)
        {
            cc=10;
            rendersingleobject(230,470,cc++);
            rendersingleobject(420,390,cc++);
            rendersingleobject(910,410,cc++);
        }
        else if(current_bacground==2)
        {
            cc=13;
            rendersingleobject(570,40,cc++);
```

```
            rendersingleobject(600,145,cc++);
            rendersingleobject(830,250,cc++);
        }

    }
    else if(stage==3)
    {
        if(current_bacground==0)
        {
            rendersingleobject(200,200,cc++);
            rendersingleobject(265,445,cc++);
            rendersingleobject(755,320,cc++);
            rendersingleobject(320,200,cc++);
            rendersingleobject(610,140,cc++);
            rendersingleobject(750,90,cc++);

        }
        else if(current_bacground==3)
        {
            cc=6;
            rendersingleobject(610,140,cc++);
            rendersingleobject(750,90,cc++);
            rendersingleobject(910,410,cc++);
        }
        if(current_bacground==2)
        {
            cc=9;
            rendersingleobject(550,535,cc++);
            rendersingleobject(600,145,cc++);
            rendersingleobject(850,515,cc++);
        }


    }
    else if(stage==2)
    {
        if(current_bacground==0)
        {
            rendersingleobject(200,200,cc++);
            rendersingleobject(265,445,cc++);
            rendersingleobject(755,320,cc++);
            rendersingleobject(320,200,cc++);
            rendersingleobject(610,140,cc++);
            rendersingleobject(750,90,cc++);

        }

        else if(current_bacground==3)
        {
            cc=6;
            rendersingleobject(610,140,cc++);
            rendersingleobject(750,90,cc++);
            rendersingleobject(910,410,cc++);
        }
    }
    else if(stage==1)
    {
        if(current_bacground==0)
        {
            rendersingleobject(200,200,cc++);
            rendersingleobject(265,445,cc++);
            rendersingleobject(755,320,cc++);
            rendersingleobject(320,200,cc++);
```

```
            rendersingleobject(610,140,cc++);
            rendersingleobject(750,90,cc++);
            rendersingleobject(870,200,cc++);
        }

    }
}
char dynamic_speech[15][1000];
int index_of_whole_speech=0,ind_of_current_speech=0,line=0;
void speak(char text[],int tot_line,int startsignal,int xlen,int ylen, bool middle)
{
  if(index_of_whole_speech<strlen(text))
  {
   dynamic_speech[line][ind_of_current_speech++]=text[index_of_whole_speech++];

   if(dynamic_speech[line][ind_of_current_speech-1]=='\n')
   {
    dynamic_speech[line][ind_of_current_speech-1]=0;
    line++;ind_of_current_speech=0;
    dynamic_speech[line][ind_of_current_speech++]=text[index_of_whole_speech++];
   }
  }
  const Uint8* currentKeyStates = SDL_GetKeyboardState( NULL );

    if( currentKeyStates[ SDL_SCANCODE_BACKSPACE ] )
  {
   pause=false;
        pause_for_blocking_stage2=false;
   pause_for_blocking_stage3=false;
        pause_for_blocking_stage4=false;
        savepause=false;
        gamesaved=false;
        if(startsignal==1)stagestart=true;
        else if(startsignal==-1)
        {
            stagestart=false;
            if(stage==1)
            {
                wall[0][14].x=0;
                wall[0][14].y=0;
                wall[0][14].w=0;
                wall[0][14].h=0;
                stage=2;
            }
            else if(stage==2)
                stage=3;
            else if(stage==3)
            {
                wall[2][21].x=0;
                wall[2][21].y=0;
                wall[2][21].w=0;
                wall[2][21].h=0;
                stage=4;
            }
            else if(stage==4)
            {
                wall[1][17].x=0;
                wall[1][17].y=0;
                wall[1][17].w=0;
                wall[1][17].h=0;
                stage=5;

            }
```

```
        }
        for(int i=0;i<=line;i++)memset(dynamic_speech[i],0,sizeof(dynamic_speech[i]));
   index_of_whole_speech=0;ind_of_current_speech=0;line=0;
   return;
  }

  SDL_Color textColor = { 0xFF, 0xFF, 0xFF };
  for(int i=0;i<=line;i++)
  {
   speech[i].loadFromRenderedText( dynamic_speech[i], textColor );
   if(!middle)speech[i].render(xlen,ylen+(i*30)-tot_line*30);
     else speech[i].render((SCREEN_WIDTH - speech[i].getWidth()) / 2,ylen+(i*30)-
tot_line*30); //(SCREEN_WIDTH - speech[i].getWidth()) / 2
    }
}
void speak_helps(char text[],int tot_line,int startsignal,int xlen,int ylen, bool
middle)
{
    if(index_of_whole_speech<strlen(text))
    {
        dynamic_speech[line][ind_of_current_speech++]=text[index_of_whole_speech++];

        if(dynamic_speech[line][ind_of_current_speech-1]=='\n')
        {
            dynamic_speech[line][ind_of_current_speech-1]=0;
            line++;ind_of_current_speech=0;

dynamic_speech[line][ind_of_current_speech++]=text[index_of_whole_speech++];
        }
    }
    if(middle)
    {
        for(int i=0;i<=line;i++)memset(dynamic_speech[i],0,sizeof(dynamic_speech[i]));
        index_of_whole_speech=0;ind_of_current_speech=0;line=0;
        return;

    }
    SDL_Color textColor = { 0x00, 0x00, 0x00 };
    for(int i=0;i<=line;i++)
    {
        speech[i].loadFromRenderedText( dynamic_speech[i], textColor );
        if(!middle)speech[i].render(xlen,ylen+(i*40)-tot_line*30);
        else speech[i].render((SCREEN_WIDTH - speech[i].getWidth()) / 2,ylen+(i*30)-
tot_line*30); //(SCREEN_WIDTH - speech[i].getWidth()) / 2
    }
}
void render_time()
{
    int s;
    SDL_Color textColor = { 255, 255, 255};
    timeText.str( "" );
    s=(SDL_GetTicks() - startTime)/1000;
    timeText<<(s/3600)%24<<":" <<(s/60)%60<<":"<<s%60;

    //Render text
    gTimeTexture.loadFromRenderedText( timeText.str().c_str(), textColor );
    render_darkbox(878,39,gTimeTexture.getWidth()+4,38);
    gTimeTexture.render( 880, 40 );

}

void render_scoreboard()
{
```

```
    SDL_Color textColor = { 0, 0, 0, 255 };
    speech[11].loadFromRenderedText( "Name of the player", textColor);
    speech[12].loadFromRenderedText("Gameplay time",textColor);

    SDL_SetRenderDrawColor( gRenderer, 0xFF, 0xFF, 0xFF, 0xFF );
    SDL_RenderClear( gRenderer );
    menuraw.render(0,0);
    SDL_SetRenderDrawColor( gRenderer, 0x00, 0x00, 0x00, 0xFF );
    SDL_RenderDrawLine( gRenderer, SCREEN_WIDTH/2, 0, SCREEN_WIDTH/2, SCREEN_HEIGHT );
    SDL_RenderDrawLine( gRenderer, 0, 100, SCREEN_WIDTH,100);
  speech[11].render( 100, 50);
    speech[12].render(600,50);
    FILE *fp=fopen("saveddoc/leaderboard.txt","r");
    char x[32];
    for(int i=0;i<5;i++)
    {
        fgets(x,32,fp);
        x[strlen(x)-1]=0;
        positiontext.str( "" );
        positiontext<<i+1<<"." <<x;
        speech[i].loadFromRenderedText( positiontext.str().c_str(), textColor);
        speech[i].render(100,150+60*i);
    }
    for(int i=5;i<10;i++)
    {
        int temp;
        fscanf(fp,"%d",&temp);
        timeText.str( "" );

timeText<<((temp/1000)/3600)<<":"<<((temp/1000)/60)%60<<":"<<(temp/1000)%60<<"."<<temp
%1000;
        speech[i].loadFromRenderedText( timeText.str().c_str(), textColor);
        speech[i].render(600,150+60*(i-5));
    }
    fclose(fp);
    speech[13].loadFromRenderedText("Back to Main Menu",textColor);
    speech[13].render(700,500);
    render_outline(695,495,280,50);

    SDL_RenderPresent( gRenderer );
    SDL_Delay(25);

}
int dynamic_credit_postion=600,c=0,gameplaytime;
bool checkscorername=false;
char scorername[63];
bool scoreresetchec=false,endgametimetaken=false,speachtaken=false;

void endgame()
{


    if(!endgametimetaken)
    {
        endgametimetaken=true;
        gameplaytime=SDL_GetTicks()-startTime;
    }
    if(!speachtaken)
    {
        speachtaken=true;
        dynamic_credit_postion=600;
    }
    SDL_Color textColor = { 0, 0, 0, 255 };
```

```
    speech[0].loadFromRenderedText( "Thanks for playing this game", textColor);
    speech[1].loadFromRenderedText("This game is developed by Rafid and
Nayeem",textColor);
    speech[2].loadFromRenderedText("Enter Your Name Here",textColor);
    speech[4].loadFromRenderedText("Your gameplay time was:",textColor);
    timeText.str("");

timeText<<(gameplaytime/1000)/3600<<":"<<((gameplaytime/1000)/60)%60<<":"<<(gameplayti
me/1000)%60<<"."<<gameplaytime%1000;
    speech[5].loadFromRenderedText(timeText.str().c_str(),textColor);
    SDL_SetRenderDrawColor( gRenderer, 0x00, 0x00, 0x00, 0xFF  );

    SDL_RenderClear( gRenderer );

    menuraw.render(0,0);
    speech[0].render(300,dynamic_credit_postion);
    speech[1].render(200,dynamic_credit_postion+50);
    if(dynamic_credit_postion>-100)
        dynamic_credit_postion--;
    else
    {
        speech[4].render((SCREEN_WIDTH-speech[4].getWidth())/2-100,200);
        speech[5].render(SCREEN_WIDTH-(SCREEN_WIDTH-speech[4].getWidth())/2-100,200);
        speech[2].render((SCREEN_WIDTH-speech[2].getWidth())/2-45,250);
        checkscorername=true;
        if(c==0)speech[3].loadFromRenderedText(" ",textColor);
        else speech[3].loadFromRenderedText(scorername,textColor);
        speech[3].render((SCREEN_WIDTH-speech[3].getWidth())/2-70,300);

        if(scoreresetchec)
        {
            leaderboardreset();
            memset(scorername,0,sizeof(scorername));
            dynamic_credit_postion=600;
            c=0;
            gameplaytime=0;
            endgametimetaken=false;
            check_menu=true;
            checkscorername=false;
            endgamecheck=false;
            scoreresetchec=false;
            speachtaken=false;
            reset();
        }
    }

    SDL_RenderPresent( gRenderer );


}
struct name_and_score
{
    char name[100];
    int score;
};
void leaderboardreset()
{
    name_and_score temp[6];
    FILE *fp,*fpo;
    fp=fopen("saveddoc/leaderboard.txt","r");
    for(int i=0;i<5;i++)
        fgets(temp[i].name,100,fp);
    for(int i=0;i<5;i++)
```

```
    {
        fscanf(fp,"%d",&temp[i].score);
    }

    fclose(fp);
    int len=strlen(scorername);
    if(len<1)
    {
        char kjlskkljdfs[100]="(no name)";
        len=9;
    }
    scorername[len++]='\n';scorername[len]=0;
    for(int i=0;i<len;i++)
        temp[5].name[i]=scorername[i];
    temp[5].name[len]=0;

    temp[5].score=gameplaytime;
    for(int i=5;i>=0;i--)
    {
        if(temp[i].score<temp[i-1].score)
        {
            name_and_score tmp;
            tmp=temp[i];
            temp[i]=temp[i-1];
            temp[i-1]=tmp;
        }
    }
    fpo=fopen("saveddoc/leaderboard.txt","w");
    for(int i=0;i<5;i++)
    {
        fprintf(fpo, "%s", temp[i].name);
    }
    for(int i=0;i<5;i++)
    {
        fprintf(fpo, "%d\n",temp[i].score );
    }
    fclose(fpo);
}

void enterscorername(SDL_Event* e)
{
    if( e->type == SDL_KEYDOWN )
    {
        switch( e->key.keysym.sym )
        {
            case SDLK_a:
                scorername[c++]='a';
            break;
            case SDLK_b:
                scorername[c++]='b';
            break;
            case SDLK_c:
                scorername[c++]='c';
            break;
            case SDLK_d:
                scorername[c++]='d';
            break;
            case SDLK_e:
                scorername[c++]='e';
            break;
            case SDLK_f:
                scorername[c++]='f';
            break;
```

```
case SDLK_g:
    scorername[c++]='g';
break;
case SDLK_h:
    scorername[c++]='h';
break;
case SDLK_i:
    scorername[c++]='i';
break;
case SDLK_j:
    scorername[c++]='j';
break;
case SDLK_k:
    scorername[c++]='k';
break;
case SDLK_l:
    scorername[c++]='l';
break;
case SDLK_m:
    scorername[c++]='m';
break;
case SDLK_n:
    scorername[c++]='n';
break;
case SDLK_o:
    scorername[c++]='o';
break;
case SDLK_p:
    scorername[c++]='p';
break;
case SDLK_q:
    scorername[c++]='q';
break;
case SDLK_r:
    scorername[c++]='r';
break;
case SDLK_s:
    scorername[c++]='s';
break;
case SDLK_t:
    scorername[c++]='t';
break;
case SDLK_u:
    scorername[c++]='u';
break;
case SDLK_v:
    scorername[c++]='v';
break;
case SDLK_w:
    scorername[c++]='w';
break;
case SDLK_x:
    scorername[c++]='x';
break;
case SDLK_y:
    scorername[c++]='y';
break;
case SDLK_z:
    scorername[c++]='z';
break;
case SDLK_SPACE:
    scorername[c++]=' ';
break;
```

```
            case SDLK_BACKSPACE:
                if(c!=0)scorername[--c]=0;
            break;
            case SDLK_RETURN:
            {
                if(strlen(scorername)!=0)
                    scoreresetchec=true;
            }
            break;
        }
    }
}
void difficultyset(bool *quit,bool *esaped)
{
    bool difficulty_selection_check=false,textresetchec=false;
    SDL_Event eee;
    bool render_help=true;
    int difficultybar=0;
    SDL_Rect easy ={ 435, 155, 190, 55 };
    SDL_Rect hard ={ 435, 240, 190, 55 };
    SDL_Rect easey ={ 430, 150, 200, 65 };
    SDL_Rect haard ={ 430, 235, 200, 65 };
    SDL_Color coloroftext={0,0,0};
    char line[1000]="Welcome. In this game You are to find some person, talk with
them\n and do some tasks for them. And then when you\n are done . You have to get out
of there.\n Press H while playing if you need any help\nBest of luck. \nPress Enter to
continue";
    int tempdifbar=-1;
    while( !*quit && !difficulty_selection_check)
    {
        while( SDL_PollEvent( &eee ) != 0 )
        {
            if( eee.type == SDL_QUIT )
            {
                *quit = true;
            }
            if(eee.type == SDL_KEYDOWN && eee.key.keysym.sym == SDLK_RETURN)
            {
                render_help=false;
            }
            if(!render_help)
            {
                if(easybutton.handleEvent(&eee, 200, 65, 1,&difficultybar) ||(eee.type
== SDL_KEYDOWN && eee.key.keysym.sym == SDLK_RETURN && difficultybar==1))
                {
                    Mix_PlayChannel( -1, click , 0 );
                    SDL_Delay(500);
                    easyplay=true;difficulty_selection_check=true;
                }
                if(hardbutton.handleEvent(&eee,200,65,2,&difficultybar)|| (eee.type ==
SDL_KEYDOWN && eee.key.keysym.sym == SDLK_RETURN && difficultybar==2))
                {
                    Mix_PlayChannel( -1, click , 0 );
                    SDL_Delay(500);
                    easyplay=false;difficulty_selection_check=true;
                }
                if(eee.type == SDL_KEYDOWN && eee.key.keysym.sym == SDLK_UP)
                    difficultybar=1;
                if(eee.type == SDL_KEYDOWN && eee.key.keysym.sym == SDLK_DOWN)
                    difficultybar=2;
                if(difficultybar>0 && tempdifbar!=difficultybar)
                {
                    Mix_PlayChannel( -1, selected , 0 );
```

```
                    tempdifbar=difficultybar;

                }
            }
            if(eee.type == SDL_KEYDOWN && eee.key.keysym.sym == SDLK_ESCAPE)
            {
                Mix_PlayChannel( -1, click , 0 );
                SDL_Delay(500);
                *esaped=true;
                speak_helps(line,1,0,50,100,true);
                return;
            }


        }
        SDL_SetRenderDrawColor( gRenderer, 0xFF, 0xFF, 0xFF, 0x00 );
        SDL_RenderClear( gRenderer );
        menuraw.render(0,0);
        if(render_help)
        {

            speak_helps(line,1,0,50,100,!render_help);
            SDL_Delay(10);

        }
        else
        {
            if(!textresetchec)
            {
                speak_helps(line,1,0,50,100,!render_help);
                textresetchec=true;
            }
            SDL_SetRenderDrawColor( gRenderer, 0x00, 0x00, 0x00, 0x05 );

            SDL_RenderDrawRect( gRenderer, &easy );
            SDL_RenderDrawRect( gRenderer, &hard );
            if(difficultybar==1)
            {

                SDL_SetRenderDrawColor( gRenderer, 0x11, 0x11, 0x11, 0x05 );
                //SDL_RenderClear(gRenderer);
                SDL_RenderFillRect( gRenderer, &easey );
                dificultytext[0].loadFromRenderedText("In this difficulty,You have to
collect less items",coloroftext);
                dificultytext[1].loadFromRenderedText("And you can save while
collecting items",coloroftext);
                dificultytext[0].render((SCREEN_WIDTH-
dificultytext[0].getWidth())/2,350);
                dificultytext[1].render((SCREEN_WIDTH-
dificultytext[1].getWidth())/2,390);

            }
            if(difficultybar==2)
            {

                SDL_SetRenderDrawColor( gRenderer, 0x00, 0x00, 0x00, 0x05 );
                ///SDL_RenderClear(gRenderer);
                //haard.setAlpha(4);
                SDL_RenderFillRect( gRenderer, &haard );
                dificultytext[0].loadFromRenderedText("In this difficulty,You have to
collect more items",coloroftext);
                dificultytext[1].loadFromRenderedText("And you cant save while
collecting items",coloroftext);
```

```
                dificultytext[0].render((SCREEN_WIDTH-
dificultytext[0].getWidth())/2,350);
                dificultytext[1].render((SCREEN_WIDTH-
dificultytext[1].getWidth())/2,390);
            }

            SDL_Color design={23,44,144};
            dificultytext[2].loadFromRenderedText("EASY",design);
            dificultytext[3].loadFromRenderedText("HARD",design);
            dificultytext[2].render(480,170);
            dificultytext[3].render(480,255);
            SDL_Delay(50);

        }
        //SDL_Delay(10);
        SDL_RenderPresent( gRenderer );
        //SDL_RenderClear(gRenderer);

    }
}



void render_saves(int *ith,bool *quit)
{
    bool selecte=false;
    LButton save1,save2,save3;
    save1.setPosition(400,200);
    save2.setPosition(400,280);
    save3.setPosition(400,360);
    int savebar=0,tempsavebar=-1;
    SDL_Event eee;
    SDL_Color clrrrr1,clrrrr2,clrrrr3;
    while( !*quit && !selecte)
    {
        //Handle events on queue
        while( SDL_PollEvent( &eee ) != 0 )
        {
            //User requests quit
            if( eee.type == SDL_QUIT )
            {
                *quit = true;
            }
            if(save1.handleEvent(&eee, 200, 65, 1,&savebar)||(eee.type == SDL_KEYDOWN
&& eee.key.keysym.sym == SDLK_RETURN && savebar==1))
            {
                Mix_PlayChannel( -1, click , 0 );
                SDL_Delay(500);
                *ith=1;selecte=true;
            }
            if(save2.handleEvent(&eee, 200, 65, 2,&savebar) ||(eee.type == SDL_KEYDOWN
&& eee.key.keysym.sym == SDLK_RETURN && savebar==2))
            {
                Mix_PlayChannel( -1, click , 0 );
                SDL_Delay(500);
                *ith=2;selecte=true;
            }
            if(save3.handleEvent(&eee, 200, 65, 3,&savebar)||(eee.type == SDL_KEYDOWN
&& eee.key.keysym.sym == SDLK_RETURN && savebar==3))
            {
                Mix_PlayChannel( -1, click , 0 );
                SDL_Delay(500);
                *ith=3;selecte=true;
```

```
            }
            if(eee.type == SDL_KEYDOWN && eee.key.keysym.sym == SDLK_UP && savebar>1)
                savebar--;
            if(eee.type == SDL_KEYDOWN && eee.key.keysym.sym == SDLK_DOWN &&
savebar<3)
                savebar++;
            if(eee.type == SDL_KEYDOWN && eee.key.keysym.sym == SDLK_UP && savebar<1)
                savebar=3;
            if(eee.type == SDL_KEYDOWN && eee.key.keysym.sym == SDLK_ESCAPE)
            {
                Mix_PlayChannel( -1, click , 0 );
                SDL_Delay(500);
                return;
            }
            if(savebar>0 && tempsavebar!=savebar)
            {
                Mix_PlayChannel( -1, selected , 0 );
                tempsavebar=savebar;

            }
        }
        SDL_SetRenderDrawColor( gRenderer, 0xFF, 0xFF, 0xFF, 0x00 );
        SDL_RenderClear(gRenderer);
        clrrrr1={255,255,255};
        clrrrr2={255,255,255};
        clrrrr3={255,255,255};
        menuraw.render(0,0);

        if(savebar==1){clrrrr1={255,0,0};}
        else if(savebar==2){clrrrr2={255,0,0};}
        else if(savebar==3){clrrrr3={255,0,0};}
        speech[0].loadFromRenderedText("Save file 1",clrrrr1);
        speech[1].loadFromRenderedText("Save file 2",clrrrr2);
        speech[2].loadFromRenderedText("Save file 3",clrrrr3);
        speech[0].render(420,220);
        speech[1].render(420,300);
        speech[2].render(420,380);
        SDL_RenderPresent( gRenderer );
    }
}
bool ashholei(bool *quit)
{
    LButton yes;
    LButton no;
    int azairabar=0,tempazaira=-1;
    yes.setPosition(400,300);
    no.setPosition(400,380);
    SDL_Color azaira1,azaira2;
    while(!*quit)
    {
        SDL_Event ee;
        while(SDL_PollEvent( &ee ) != 0)
        {
            if( ee.type == SDL_QUIT )
            {
                *quit = true;
            }
            if(yes.handleEvent(&ee, 100, 65, 1,&azairabar)||(ee.type == SDL_KEYDOWN &&
ee.key.keysym.sym == SDLK_RETURN && azairabar==1))
            {
                return true;
            }
```

```
            if(no.handleEvent(&ee, 100, 65, 2,&azairabar)||(ee.type == SDL_KEYDOWN &&
ee.key.keysym.sym == SDLK_RETURN && azairabar==2))
            {
                return false;
            }
            if(ee.type == SDL_KEYDOWN && ee.key.keysym.sym == SDLK_UP)
                azairabar=1;
            if(ee.type == SDL_KEYDOWN && ee.key.keysym.sym == SDLK_DOWN)
                azairabar=2;
            if(azairabar>0 && tempazaira!=azairabar)
            {
                Mix_PlayChannel( -1, selected , 0 );
                tempazaira=azairabar;

            }

        }
        SDL_RenderClear(gRenderer);
        azaira1={255,255,255};
        azaira2={255,255,255};
        speech[2].loadFromRenderedText("Any unsaved progress will be
deleted.",azaira1);
        speech[3].loadFromRenderedText("Are you sure you want to go back to the main
menu?",azaira2);
        speech[2].render((SCREEN_WIDTH-speech[2].getWidth())/2,200);
        speech[3].render((SCREEN_WIDTH-speech[3].getWidth())/2,250);

        if(azairabar==1)azaira1={255,0,255};
        else if(azairabar==2)azaira2={255,0,255};
        speech[0].loadFromRenderedText("yes",azaira1);
        speech[1].loadFromRenderedText("no",azaira2);
        speech[0].render(420,330);
        speech[1].render(423,390);
        SDL_RenderPresent(gRenderer);
    }
  return true;
}
void helpline(bool *quit)
{
    SDL_Event f;
    SDL_Color helpcolor={0,255,255};
    while(!*quit)
    {
        while(SDL_PollEvent( &f ) != 0)
        {
            if( f.type == SDL_QUIT )
            {
                *quit = true;
            }
            if(f.type == SDL_KEYDOWN && f.key.keysym.sym == SDLK_ESCAPE)
                return;
        }

        SDL_RenderClear(gRenderer);
        speech[0].loadFromRenderedText("HELPS",helpcolor);
        speech[0].render((SCREEN_WIDTH-speech[0].getWidth())/2,10);
        speech[1].loadFromRenderedText("Press M to turn on and off music",helpcolor);
        speech[1].render((SCREEN_WIDTH-speech[1].getWidth())/2,50);
        speech[2].loadFromRenderedText("Press N to turn on and off nitro shoes when
available",helpcolor);
        speech[2].render((SCREEN_WIDTH-speech[2].getWidth())/2,90);
        speech[3].loadFromRenderedText("Press Escape to go to the main
menu",helpcolor);
```

```
        speech[3].render((SCREEN_WIDTH-speech[3].getWidth())/2,130);
        speech[4].loadFromRenderedText("Press direction keypad to move
around",helpcolor);
        speech[4].render((SCREEN_WIDTH-speech[4].getWidth())/2,170);
        speech[5].loadFromRenderedText("Press and hold Left shift to move
faster",helpcolor);
        speech[5].render((SCREEN_WIDTH-speech[5].getWidth())/2,210);
        speech[6].loadFromRenderedText("Read carefullly what people says",helpcolor);
        speech[6].render((SCREEN_WIDTH-speech[6].getWidth())/2,250);
        speech[7].loadFromRenderedText("Press Backspace to get out conversation and
white colored texts",helpcolor);
        speech[7].render((SCREEN_WIDTH-speech[7].getWidth())/2,290);
        speech[8].loadFromRenderedText("While gamplay,time keeps on moving unless you
go back to the main menu",helpcolor);
        speech[8].render((SCREEN_WIDTH-speech[8].getWidth())/2,330);
        speech[9].loadFromRenderedText("This means it is increasing now too. Press
Escape to go back playing",helpcolor);
        speech[9].render((SCREEN_WIDTH-speech[9].getWidth())/2,370);
        speech[10].loadFromRenderedText("And explore more trying different things,or
exit the place.Your wish",helpcolor);
        speech[10].render((SCREEN_WIDTH-speech[10].getWidth())/2,410);
        speech[11].loadFromRenderedText("Oh and press Enter in outlined boxes if you
are confused",helpcolor);
        speech[11].render((SCREEN_WIDTH-speech[11].getWidth())/2,450);
   speech[12].loadFromRenderedText("And to save, click the save button at the right
bottom corner", helpcolor);
   speech[12].render((SCREEN_WIDTH - speech[12].getWidth()) / 2, 490);
        SDL_RenderPresent(gRenderer);
        SDL_Delay(50);
    }
}
```

The End