My Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

Labyrinth

A rudimentary video game. All sprites and backgrounds courtesy of Hugh Keene.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 Background Class Reference

`#include <background.h>`

Public Member Functions

- Background (vector< SpriteLayer * > layers, SDL_Renderer *ren)
- Background (SDL_Renderer *ren)
- ~Background ()
- void add_layer (SpriteLayer *layer)
- void left (uint speed)
- void right (uint speed)
- void reset ()
- void reset (int offset)
- void draw ()
- void blank ()

3.1.1 Detailed Description

A background class

Author

Jared Allen

Version

8 February 2019

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Background() [1/2]

```
Background::Background (
            vector< SpriteLayer * > layers,
            SDL_Renderer * ren )
```

Constructor

Parameters

layers	the sprites that layer the background

3.1.2.2 Background() [2/2]

```
Background::Background (
            SDL_Renderer * ren )
```

Constructor

3.1.2.3 ~Background()

```
Background::~Background ( )
```

Destructor

3.1.3 Member Function Documentation

3.1.3.1 add_layer()

```
void Background::add_layer (
            SpriteLayer * layer )
```

add a layer to background

Parameters

layer	

3.1.3.2 blank()

```
void Background::blank ( )
```

draw a blank background

3.1.3.3 draw()

```
void Background::draw ( )
```

draw the background

Parameters

renderer	the renderer

3.1.3.4 left()

```
void Background::left (
            uint speed )
```

move background left

3.1.3.5 reset() [1/2]

```
void Background::reset ( )
```

reset the background

3.1.3.6 reset() [2/2]

```
void Background::reset (
            int offset )
```

reset the background with offset

3.1.3.7 right()

```
void Background::right (
            uint speed )
```

move background right

The documentation for this class was generated from the following files:

- background.h
- background.cpp

3.2 Character Class Reference

```
#include <character.h>
```

Public Member Functions

- Character (string name, Sprite *full_body, Sprite *torso, uint force, uint diversion, uint stealth, uint num_↩ walking_sprites, uint num_talking_sprites)
- ~Character ()
- void happy ()
- void gasp ()
- void walk_left (uint speed, uint count)
- void walk_right (uint speed, uint count)
- void jump (int y_velocity)
- void jump (int x_velocity, int y_velocity)
- void stand ()
- void set_position (int x, int y)
- void set_screen_position (int x, int y)
- vector< int > get_screen_position ()
- vector< int > get_position ()
- void follow (Character *leader, uint speed, uint count)
- void update_pos (bool left, uint speed)
- void reset ()
- void reset (uint offset)
- bool equals (Character *character)
- string get_name ()
- void set_stage_pos (int screen_pos, int pos)
- void set_recruited ()
- bool is_recruited ()

3.2.1 Detailed Description

A character class

Author

Jared Allen

Version

9 February 2019

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Character()

```
Character::Character (
            string name,
            Sprite * full_body,
            Sprite * torso,
            uint force,
            uint diversion,
            uint stealth,
            uint num_walking_sprites,
            uint num_talking_sprites )
```

Constructor

Parameters

full_body	the full body sprite
torso	the torso sprite

3.2.2.2 ~Character()

```
Character::~Character ( )
```

Destructor

3.2.3 Member Function Documentation

3.2.3.1 equals()

```
bool Character::equals (
            Character * character )
```

determine if two characters are equal

3.2.3.2 follow()

```
void Character::follow (
            Character * leader,
            uint speed,
            uint count )
```

make NPCs follow the character

3.2.3.3 gasp()

```
void Character::gasp ( )
```

set character gasp expression

3.2.3.4 get_name()

```
string Character::get_name ( )
```

gets the name of the character

3.2.3.5 get_position()

```
vector< int > Character::get_position ( )
```

get the characters coordinates

3.2.3.6 get_screen_position()

```
vector< int > Character::get_screen_position ( )
```

get the characters coordinates

3.2.3.7 happy()

```
void Character::happy ( )
```

set character happy expression

3.2.3.8 is_recruited()

```
bool Character::is_recruited ( )
```

determine if character has been recruited

3.2.3.9 jump() [1/2]

```
void Character::jump (
            int y_velocity )
```

make the character jump

3.2.3.10 jump() [2/2]

```
void Character::jump (
            int x_velocity,
            int y_velocity )
```

make the character jump laterally

3.2.3.11 reset() [1/2]

```
void Character::reset ( )
```

reset characters position

3.2.3.12 reset() [2/2]

```
void Character::reset (
            uint offset )
```

reset characters position with offset

3.2.3.13 set_position()

```
void Character::set_position (
            int x,
            int y )
```

set the characters coordinates

3.2.3.14 set_recruited()

```
void Character::set_recruited ( )
```

recruit a character

3.2.3.15 set_screen_position()

```
void Character::set_screen_position (
            int x,
            int y )
```

set the characters screen coordinates

3.2.3.16 set_stage_pos()

```
void Character::set_stage_pos (
            int screen_pos,
            int pos )
```

set stage position

3.2.3.17 stand()

```
void Character::stand ( )
```

make the character stand

3.2.3.18 update_pos()

```
void Character::update_pos (
            bool left,
            uint speed )
```

update characters position

3.2.3.19 walk_left()

```
void Character::walk_left (
            uint speed,
            uint count )
```

make the character walk left

Parameters

speed	the speed that the character walks
count	the characters cycle count

3.2.3.20 walk_right()

```
void Character::walk_right (
            uint speed,
            uint count )
```

make the character walk right

Parameters

speed	the speed that the character walks
count	the characters walking cycle count

The documentation for this class was generated from the following files:

- character.h
- character.cpp

3.3 Conversation Class Reference

```
#include <conversation.h>
```

Public Member Functions

- Conversation (string filepath, SDL_Renderer ∗renderer)
- Conversation ()
- string get_dialogue (uint index)
- void add_dialogue (char ∗words, SDL_Renderer ∗renderer)
- uint get_length ()
- void set_angry_response (char ∗words, SDL_Renderer ∗renderer)
- string get_angry_response ()

3.3.1 Detailed Description

A conversation class

Author

Jared Allen

Version

30 March 2019

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Conversation() [1/2]

```
Conversation::Conversation (
            string filepath,
            SDL_Renderer * renderer )
```

Constructor

3.3.2.2 Conversation() [2/2]

```
Conversation::Conversation ( )
```

default constructor

3.3.3 Member Function Documentation

3.3.3.1 add_dialogue()

```
void Conversation::add_dialogue (
            char * words,
            SDL_Renderer * renderer )
```

add dialogue to conversation

3.3.3.2 get_angry_response()

```
string Conversation::get_angry_response ( )
```

get angry response

3.3.3.3 get_dialogue()

```
string Conversation::get_dialogue (
            uint index )
```

get dialogue at the index provided

3.3.3.4 get_length()

```
uint Conversation::get_length ( )
```

get length of the dialogue vector

3.3.3.5 set_angry_response()

```
void Conversation::set_angry_response (
            char * words,
            SDL_Renderer * renderer )
```

set an angry response

The documentation for this class was generated from the following files:

- conversation.h
- conversation.cpp

3.4 Game Class Reference

```
#include <game.h>
```

Public Member Functions

- Game ()
- ~Game ()
- void play ()
- void set_introduction (Panel *panel)
- void add_scene (Scene *scene, uint scene_track)
- void join_scenes (uint track1_index, uint track2_index, uint scene1_pos, uint scene2_pos, int scene1_junction_pos, int scene2_junction_pos)

3.4.1 Detailed Description

A game class

Author

Jared Allen

Version

8 February 2019

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Game()

```
Game::Game ( )
```

Constructer

3.4.2.2 ~Game()

```
Game::~Game ( )
```

Destructor

3.4.3 Member Function Documentation

3.4.3.1 add_scene()

```
void Game::add_scene (
            Scene * scene,
            uint scene_track )
```

add scene to game

3.4.3.2 join_scenes()

```
void Game::join_scenes (
            uint track1_index,
            uint track2_index,
            uint scene1_pos,
            uint scene2_pos,
            int scene1_junction_pos,
            int scene2_junction_pos )
```

join two scenes

Parameters

scene1	the first scene
scene2	the second scene

3.4.3.3 play()

```
void Game::play ( )
```

plays the game

3.4.3.4 set_introduction()

```
void Game::set_introduction (
            Panel * panel )
```

sets the introduction panel of the game

Parameters

panel	the introduction panel

The documentation for this class was generated from the following files:

- game.h
- game.cpp

3.5 Interaction Class Reference

```
#include <interaction.h>
```

Public Member Functions

- Interaction (vector< string > messages, int param_scene_position, SDL_Renderer *renderer)
- int get_scene_position ()
- uint get_interaction_length ()
- string get_message (uint index)

3.5.1 Detailed Description

An interaction class Jared Allen

Version

20 July 2019

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Interaction()

```
Interaction::Interaction (
            vector< string > messages,
            int param_scene_position,
            SDL_Renderer * renderer )
```

Constructor

3.5.3 Member Function Documentation

3.5.3.1 get_interaction_length()

```
uint Interaction::get_interaction_length ( )
```

get the length of the interaction

Returns

the length of the interaction

3.5.3.2 get_message()

```
string Interaction::get_message (
            uint index )
```

get the message at the given position

Parameters

index	the index of the message

Returns

the message

3.5.3.3 get_scene_position()

```
int Interaction::get_scene_position ( )
```

get the scene position

Returns

the scene position

The documentation for this class was generated from the following files:

- interaction.h
- interaction.cpp

3.6 Panel Class Reference

`#include <panel.h>`

Public Member Functions

- Panel (Background *background, SDL_Renderer *renderer)
- bool display ()

3.6.1 Detailed Description

A panel class

Author

Jared Allen

Version

5 July 2019

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Panel()

```
Panel::Panel (
            Background * background,
            SDL_Renderer * renderer )
```

Constructor

3.6.3 Member Function Documentation

3.6.3.1 display()

```
bool Panel::display ( )
```

displays the panel

The documentation for this class was generated from the following files:

- panel.h
- panel.cpp

3.7 Report Struct Reference

Public Attributes

- Scene_States **status**
- int **character_position**

The documentation for this struct was generated from the following file:

- scene.h

3.8 Scene Class Reference

`#include <scene.h>`

Public Member Functions

- void add_interaction (vector< string > messages, int scene_position, SDL_Renderer *renderer)
- void prompt_interact ()
- void interact ()
- void stage_junction (int junction_position)
- Report play ()
- bool movement_key_pressed (const Uint8 *state)
- void marionette (int x_velocity, int y_velocity, uint cadence)
- void set_previous_junction_position (int position)
- int get_previous_junction_position ()
- void reset ()
- void add_follower (Character *character)
- void stage_left ()
- void stage_left_barrier ()
- void stage_right ()
- void stage_right_barrier ()
- void add_conversation (Conversation *conversation)
- Scene (SDL_Renderer *renderer, Background *background, vector< Character * > characters, Character *main_character, Script *scene_dialogue, uint speed, uint maximum_stage_displacement)
- ~Scene ()
- void set_junction (int position)
- bool enter ()

3.8.1 Detailed Description

A scene class

Author

Jared Allen

Version

8 February 2019

3.8.2 Constructor & Destructor Documentation

3.8.2.1 Scene()

```
Scene::Scene (
            SDL_Renderer * renderer,
            Background * background,
            vector< Character * > characters,
            Character * main_character,
            Script * scene_dialogue,
            uint speed,
            uint maximum_stage_displacement )
```

Constructor

Parameters

renderer	the renderer
background	the background
characters	the characters
main_character	the main character
speed	the speed

3.8.2.2 ~Scene()

```
Scene::~Scene ( )
```

Destructor

3.8.3 Member Function Documentation

3.8.3.1 add_conversation()

```
void Scene::add_conversation (
            Conversation * conversation )
```

adds conversation to scene

3.8.3.2 add_follower()

```
void Scene::add_follower (
            Character * character )
```

adds a following character

3.8.3.3 add_interaction()

```
void Scene::add_interaction (
            vector< string > messages,
            int scene_position,
            SDL_Renderer * renderer )
```

add an interaction

3.8.3.4 enter()

```
bool Scene::enter ( )
```

determine entry to linked scene

3.8.3.5 get_previous_junction_position()

```
int Scene::get_previous_junction_position ( )
```

gets the previous scene junction position

3.8.3.6 interact()

```
void Scene::interact ( )
```

perform interaction if within proximity

3.8.3.7 marionette()

```
void Scene::marionette (
            int x_velocity,
            int y_velocity,
            uint cadence )
```

moves the main character

3.8.3.8 movement_key_pressed()

```
bool Scene::movement_key_pressed (
            const Uint8 * state )
```

determines whether a movement key is being held down

3.8.3.9 play()

```
Report Scene::play ( )
```

plays the scene

3.8.3.10 prompt_interact()

```
void Scene::prompt_interact ( )
```

prompt interaction

3.8.3.11 reset()

```
void Scene::reset ( )
```

resets the scene

3.8.3.12 set_junction()

```
void Scene::set_junction (
            int position )
```

set scene junction

Parameters

pos	the junction position

3.8.3.13 set_previous_junction_position()

```
void Scene::set_previous_junction_position (
            int position )
```

sets previous scene junction position

3.8.3.14 stage_junction()

```
void Scene::stage_junction (
            int junction_position )
```

Set the stage to a position based on the junction position

Parameters

junction	position the position to set the stage to

3.8.3.15 stage_left()

```
void Scene::stage_left ( )
```

sets the scene to the left

3.8.3.16 stage_left_barrier()

```
void Scene::stage_left_barrier ( )
```

sets the scene to the left barrier

3.8.3.17 stage_right()

```
void Scene::stage_right ( )
```

sets the scene to the right

3.8.3.18 stage_right_barrier()

```
void Scene::stage_right_barrier ( )
```

sets the scene to the right barrier

The documentation for this class was generated from the following files:

- scene.h
- scene.cpp

3.9 SceneJunction Class Reference

```
#include <sceneJunction.h>
```

Public Member Functions

- SceneJunction (uint track1_index, uint track2_index, uint scene1_index, uint scene2_index, int scene1_link↩ _position, int scene2_link_position)
- SceneJunction ()
- void link_scenes (uint track1_index, uint track2_index, uint scene1_index, uint scene2_index, int scene1_↩ link_position, int scene2_link_position)
- uint get_next_scene (uint scene, uint track)
- uint get_next_track (uint scene, uint track)
- int get_next_character_position (uint scene, uint track)
- bool contains (uint scene_pos, uint track_pos, int link_position)

3.9.1 Detailed Description

A scene junction class

Author

Jared Allen

Version

17 May 2019

3.9.2 Constructor & Destructor Documentation

3.9.2.1 SceneJunction() [1/2]

```
SceneJunction::SceneJunction (
            uint track1_index,
            uint track2_index,
            uint scene1_index,
            uint scene2_index,
            int scene1_link_position,
            int scene2_link_position )
```

Constructor

Parameters

position	the position of the junction

3.9.2.2 SceneJunction() [2/2]

```
SceneJunction::SceneJunction ( )
```

default constructor

3.9.3 Member Function Documentation

3.9.3.1 contains()

```
bool SceneJunction::contains (
            uint scene_pos,
            uint track_pos,
            int link_position )
```

determine whether sceneJunction contains a scene

3.9.3.2 get_next_character_position()

```
int SceneJunction::get_next_character_position (
            uint scene,
            uint track )
```

get next character position updates the character position for the next scene

Parameters

scene	the scene of the current track
track	the track of the current scene

3.9.3.3 get_next_scene()

```
uint SceneJunction::get_next_scene (
            uint scene,
            uint track )
```

get next scene

Parameters

scene	the scene of the current track
track	the track of the current scene

3.9.3.4 get_next_track()

```
uint SceneJunction::get_next_track (
            uint scene,
            uint track )
```

get next track

Parameters

scene	the scene of the current track
track	the track of the current scene

3.9.3.5 link_scenes()

```
void SceneJunction::link_scenes (
            uint track1_index,
            uint track2_index,
            uint scene1_index,
            uint scene2_index,
            int scene1_link_position,
            int scene2_link_position )
```

set the junction position

The documentation for this class was generated from the following files:

- sceneJunction.h
- sceneJunction.cpp

3.10 Script Class Reference

```
#include <script.h>
```

Public Member Functions

- Script (vector< Character * > characters)
- Conversation * speak_to (Character *character)
- void insert_conversation (Character *character, Conversation *conversation)
- void insert_recruit_conversation (Character *character, Conversation *conversation)

3.10.1 Detailed Description

A script class

Author

Jared Allen

Version

8 February 2019

3.10.2 Constructor & Destructor Documentation

3.10.2.1 Script()

```
Script::Script (
            vector< Character * > characters )
```

Constructor

3.10.3 Member Function Documentation

3.10.3.1 insert_conversation()

```
void Script::insert_conversation (
            Character * character,
            Conversation * conversation )
```

insert new conversation into a character's dialogue

Parameters

character	the character pointer
conversation	the conversation pointer

3.10.3.2 insert_recruit_conversation()

```
void Script::insert_recruit_conversation (
            Character * character,
            Conversation * conversation )
```

insert recruitment conversation into a character's dialogue

Parameters

character	the character pointer
conversation	the conversation pointer

3.10.3.3 speak_to()

```
Conversation * Script::speak_to (
            Character * character )
```

talk to this character

Parameters

character	the character to speak to

Returns

the conversation to be had

The documentation for this class was generated from the following files:

- script.h
- script.cpp

3.11 Sprite Class Reference

`#include <sprite.h>`

Public Member Functions

- Sprite (std::string image_path, SDL_Renderer *renderer, int x, int y)
- ~Sprite ()
- void set_source (SDL_Rect *source)
- void set_destination (SDL_Rect *destination)
- void draw ()
- void draw (SDL_Rect *destination)
- void flip_draw ()
- void flip_draw (SDL_Rect *destination)
- void set_position (int new_x, int new_y)
- void set_position (int new_x)
- void reset_position ()
- void reset_position (int offset)
- int get_x ()
- int get_y ()
- int get_height ()
- int get_width ()

3.11.1 Detailed Description

A sprite class

Author

Jared Allen

Version

8 February 2019

3.11.2 Constructor & Destructor Documentation

3.11.2.1 Sprite()

```
Sprite::Sprite (
            std::string image_path,
            SDL_Renderer * renderer,
            int x,
            int y )
```

Constructor

Parameters

texture_path	the texture file name for the entire texture
x	the x coordinate
y	the y coordinate

3.11.2.2 ~Sprite()

```
Sprite::~Sprite ( )
```

Destructor

3.11.3 Member Function Documentation

3.11.3.1 draw() [1/2]

```
void Sprite::draw ( )
```

method to draw sprite

Parameters

renderer	the pointer to renderer

3.11.3.2 draw() [2/2]

```
void Sprite::draw (
```

```
            SDL_Rect * destination )
```

method to draw sprite

Parameters

renderer	the pointer to renderer
destination	the destination to be drawn to

3.11.3.3 flip_draw() [1/2]

```
void Sprite::flip_draw ( )
```

method to draw mirror sprite

Parameters

renderer	the pointer to the renderer

3.11.3.4 flip_draw() [2/2]

```
void Sprite::flip_draw (
            SDL_Rect * destination )
```

method to draw mirror sprite

Parameters

renderer	the pointer to the renderer
destination	the destination for sprite

3.11.3.5 get_height()

```
int Sprite::get_height ( )
```

get the sprite's height

3.11.3.6 get_width()

```
int Sprite::get_width ( )
```

get the sprite's width

3.11.3.7 get_x()

```
int Sprite::get_x ( )
```

method to get sprite x position

3.11.3.8 get_y()

```
int Sprite::get_y ( )
```

method to get sprite y position

3.11.3.9 reset_position() [1/2]

```
void Sprite::reset_position ( )
```

method to reset sprite position

3.11.3.10 reset_position() [2/2]

```
void Sprite::reset_position (
            int offset )
```

method to reset sprite position with offset

Parameters

offset	the offset

3.11.3.11 set_destination()

```
void Sprite::set_destination (
            SDL_Rect * destination )
```

method to change destination rectangle

Parameters

destination	the new destination

3.11.3.12 set_position() [1/2]

```
void Sprite::set_position (
```

```
            int new_x,
            int new_y )
```

method to change sprite position

Parameters

new_x	the new x coordinate
new_y	the new y coordinate

3.11.3.13 set_position() [2/2]

```
void Sprite::set_position (
            int new_x )
```

method to change sprite position

Parameters

new_x	the new x coordinate

3.11.3.14 set_source()

```
void Sprite::set_source (
            SDL_Rect * source )
```

method to change source rectangle

Parameters

source	the new source

The documentation for this class was generated from the following files:

- sprite.h
- sprite.cpp

3.12 SpriteLayer Class Reference

```
#include <sprite_layer.h>
```

Public Member Functions

- SpriteLayer (std::string image_path, SDL_Renderer ∗renderer, int x, int y, uint distance)
- ∼SpriteLayer ()
- void left (uint speed)
- void right (uint speed)
- void reset ()
- void reset (int offset)
- void draw ()

3.12.1 Detailed Description

A sprite layer class

Author

Jared Allen

Version

8 February 2019

3.12.2 Constructor & Destructor Documentation

3.12.2.1 SpriteLayer()

```
SpriteLayer::SpriteLayer (
            std::string image_path,
            SDL_Renderer * renderer,
            int x,
            int y,
            uint distance )
```

Constructor

Parameters

sprite	the sprite
distance	the distance

3.12.2.2 ∼SpriteLayer()

```
SpriteLayer::~SpriteLayer ( )
```

Destructor

3.12.3 Member Function Documentation

3.12.3.1 draw()

```
void SpriteLayer::draw ( )
```

draw the sprite layer

Parameters

renderer	the renderer

3.12.3.2 left()

```
void SpriteLayer::left (
            uint speed )
```

move layer to the left

Parameters

speed	

3.12.3.3 reset() [1/2]

```
void SpriteLayer::reset ( )
```

reset sprite layer

3.12.3.4 reset() [2/2]

```
void SpriteLayer::reset (
            int offset )
```

reset sprite layer

Parameters

offset	the offset

3.12.3.5 right()

```
void SpriteLayer::right (
            uint speed )
```

move layer to the right

Parameters

speed	

The documentation for this class was generated from the following files:

- sprite_layer.h
- sprite_layer.cpp

3.13 Text_box Class Reference

```
#include <text_box.h>
```

Public Member Functions

- Text_box (uint param_x, uint param_y, uint param_h, uint param_w)
- void display (string message, SDL_Renderer *renderer, TTF_Font *font, uint letters)

3.13.1 Detailed Description

a class to represent a text box

Author

Jared Allen

Version

1 June 2019

3.13.2 Constructor & Destructor Documentation

3.13.2.1 Text_box()

```
Text_box::Text_box (
            uint param_x,
            uint param_y,
            uint param_h,
            uint param_w )
```

constructor

Parameters

param_x	the x pos
param_y	the y pos
param_h	the height
param_w	the width

3.13.3 Member Function Documentation

3.13.3.1 display()

```
void Text_box::display (
            string message,
            SDL_Renderer * renderer,
            TTF_Font * font,
            uint letters )
```

display message on text box

Parameters

message	the message to be displayed

The documentation for this class was generated from the following files:

- text_box.h
- text_box.cpp

Index