DELTAFORCE

CPSC2720PROJECT

Generated by Doxygen 1.6.1

Tue Apr 19 00:40:37 2016

Contents

Chapter 1

Bug List

File Allegro.cc No known bugs.

File Allegro.h No known bugs.

File bullet.cc No known bugs.

File bullet.h No known bugs.

File Enemy.cc No known bugs.

File Enemy.h No known bugs.

File EnemyOne.cc No known bugs.

File EnemyOne.h No known bugs.

File Keyboard.cc No known bugs.

File Keyboard.h No known bugs.

File main.cc No known bugs.

File Plane.cc No known bugs.

File Plane.h No known bugs.

File Player.cc No known bugs.

File Player.h No known bugs.

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Class Documentation

5.1 Allegro Class Reference

`#include <Allegro.h>`

Public Member Functions

- Allegro ()

  Constructor : Creates the various object and object states that will be used in this game.

- ~Allegro ()

  Destructor : Destroys the various objects used in the game.

- int init ()

  Initial function.

- int createWindow (float FPS, int w, int h)

  Creates the display window using the width and height and initializes all the devices, objects, images, keyboard, audio, background and timer that are used in this game.

- void gameLoop ()

  This function defines the game itself. It loads the start image, the losing image, the game sounds, and the conditions of play, losing and collision.

- bool collision (Enemy *, int, int, int, int, int, int)

  Defines the collision between the enemy and the player.

- void collision1 (Enemy *aa, Player *bb, int a, int b, int c, int d)

  Defines the collision between the bullet and the enemy.

Private Attributes

- ALLEGRO_DISPLAY * display
- ALLEGRO_TIMER * timer

- ALLEGRO_EVENT_QUEUE * event_queue
- ALLEGRO_FONT * font
- Keyboard keyboard
- Player * player
- Enemy * enemy
- ALLEGRO_BITMAP * BAbitmap
- ALLEGRO_BITMAP * LObitmap
- ALLEGRO_BITMAP * EXbitmap
- Enemy * boss
- bool looping
- bool redraw

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Allegro::Allegro ()

Constructor : Creates the various object and object states that will be used in this game.

Parameters:

No parameters

Returns:

No return value

5.1.1.2 Allegro::~Allegro ()

Destructor : Destroys the various objects used in the game.

Parameters:

No parameters

Returns:

No return value

5.1.2 Member Function Documentation

5.1.2.1 bool Allegro::collision (Enemy * *aa*, int *x1*, int *y1*, int *a*, int *b*, int *c*, int *d*)

Defines the collision between the enemy and the player.

Parameters:

Enemy* : takes in the enemy object

int : x coordinate

int : y coordinate

int : value used to calculate the boundary between the enemy and the player

int : value used to calculate the boundary between the enemy and the player

int : value used to calculate the boundary between the enemy and the player

int : value used to calculate the boundary between the enemy and the player

Returns:

boolean value: returns true if there is a collision between the player and the enemy and false if there isn't

5.1.2.2 Allegro::collision1 (Enemy * *aa*, Player * *bb*, int *a*, int *b*, int *c*, int *d*)

Defines the collision between the bullet and the enemy.

Parameters:

Enemy* : takes in the enemy object

Player* : takes in a player object(which inherits from a bullet object)

int : value used to calculate the boundary between the enemy and the bullet

int : value used to calculate the boundary between the enemy and the bullet

int : value used to calculate the boundary between the enemy and the bullet

int : value used to calculate the boundary between the enemy and the bullet

Returns:

No return value

5.1.2.3 int Allegro::createWindow (float *FPS*, int *w*, int *h*)

Creates the display window using the width and height and initializes all the devices, objects, images, keyboard, audio, background and timer that are used in this game.

Parameters:

w : the width of the display

h : the height of the display

FPS

Returns:

int : returns 0 if the display is open, returns -1 if the display stops

5.1.2.4 void Allegro::gameLoop ()

This function defines the game itself. It loads the start image, the losing image, the game sounds, and the conditions of play, losing and collision.

Parameters:

No parameters

Returns:

No return value

5.1.2.5 int Allegro::init ()

Initial function.

Parameters:

No parameters

Returns:

int : 0 or -1

5.1.3 Member Data Documentation

5.1.3.1 ALLEGRO_BITMAP* Allegro::BAbitmap [private]

Image of the start page of the game

5.1.3.2 Enemy* Allegro::boss [private]

The larger enemy

5.1.3.3 ALLEGRO_DISPLAY* Allegro::display [private]

display window

5.1.3.4 Enemy* Allegro::enemy [private]

The smaller enemy

5.1.3.5 ALLEGRO_EVENT_QUEUE* Allegro::event_queue [private]

5.1.3.6 ALLEGRO_BITMAP* Allegro::EXbitmap [private]

Image of an explosion

5.1.3.7 ALLEGRO_FONT* Allegro::font [private]

5.1.3.8 Keyboard Allegro::keyboard [private]

5.1.3.9 ALLEGRO_BITMAP* Allegro::LObitmap [private]

Image of the page displayed when the user gets killed in the game

5.1.3.10 bool Allegro::looping [private]

5.1.3.11 Player* Allegro::player [private]

The player object

5.1.3.12 bool Allegro::redraw [private]

5.1.3.13 ALLEGRO_TIMER* Allegro::timer [private]

game timer

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

- Allegro.h
- Allegro.cc

5.2 Background Struct Reference

`#include <Allegro.h>`

Public Attributes

- float x
- float y
- float velX
- float velY
- int dirX
- int dirY
- int width
- int height
- ALLEGRO_BITMAP * image

5.2.1 Member Data Documentation

5.2.1.1 int Background::dirX

Direction of the background on the x axis

5.2.1.2 int Background::dirY

Direction of the background on the y axis

5.2.1.3 int Background::height

height of the background image

5.2.1.4 ALLEGRO_BITMAP* Background::image

The background image

5.2.1.5 float Background::velX

Velocity of the background moving in the X direction

5.2.1.6 float Background::velY

Velocity of the background moving in the Y direction

5.2.1.7 int Background::width

width of the background image

5.2.1.8 float Background::x

x coordinate

5.2.1.9 float Background::y

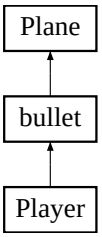
y coordinate

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

- Allegro.h

5.3 bullet Class Reference

`#include <bullet.h>`Inheritance diagram for bullet::



Public Member Functions

- bullet ()

  Constructor.

- ~bullet ()

  Destructor.

- void addBullet (int, int)

  Creates the bullet and sets it at the x and y coordinates given with the two int parameters.

- void setBullet (std::string fileName)

  Loads the bitmap image of the bullet.

- ALLEGRO_BITMAP * getBulletMap ()

  getter function to get the bitmat image of the bullet

- void drawB ()

  Draw function that actually draws the bullet image.

Public Attributes

- list< bulletType * > Blist

Protected Attributes

- ALLEGRO_BITMAP * Bbitmap

5.3.1 Constructor & Destructor Documentation

5.3.1.1 bullet::bullet ()

Constructor.

Parameters:

No parameters

Returns:

No return value

5.3.1.2 bullet::~bullet ()

Destructor.

Parameters:

No parameters

Returns:

No return value

5.3.2 Member Function Documentation

5.3.2.1 void bullet::addBullet (int *e*, int *f*)

Creates the bullet and sets it at the x and y coordinates given with the two int parameters.

Parameters:

int : representing the x coordinate of the bullet
int : representing the y coordinate of the bullet

Returns:

No return value

5.3.2.2 bullet::drawB ()

Draw function that actually draws the bullet image.

Parameters:

No parameters

Returns:

No return value

5.3.2.3 ALLEGRO_BITMAP * bullet::getBulletMap ()

getter function to get the bitmat image of the bullet

Parameters:

No parameters

Returns:

Bitmap image of the bullet

5.3.2.4 bullet::setBullet (std::string *fileName*)

Loads the bitmap image of the bullet.

Parameters:

fileName : the file name/path that contains the image of the bullet

Returns:

No return value

5.3.3 Member Data Documentation

5.3.3.1 ALLEGRO_BITMAP* bullet::Bbitmap [protected]

Creates a Bitmap object that will hold the bullet image

5.3.3.2 list<bulletType*> bullet::Blist

Creates a list of the bullets

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

- bullet.h
- bullet.cc

5.4 bulletType Class Reference

`#include <bullet.h>`

Public Member Functions

- bulletType (int a, int b)
  Constructor.
- int getbX ()
  Gets the x coordinate of the bullet.
- void setbY (int a)
  Sets the Y coordinate of the bullet to the value of the int parameter.
- void setbX (int b)
  Sets the X coordinate of the bullet to the value of the int parameter.
- int getbY ()
  Gets the y coordinate of the bullet.

Private Attributes

- int bX
- int bY

5.4.1 Constructor & Destructor Documentation

5.4.1.1 bulletType::bulletType (int *a*, int *b*) `[inline]`

Constructor.

Parameters:

a X coordinate

b Y coordinate

Returns:

No return value

5.4.2 Member Function Documentation

5.4.2.1 int bulletType::getbX () `[inline]`

Gets the x coordinate of the bullet.

Parameters:

No parameters

Returns:

Integer value

5.4.2.2 int bulletType::getbY () [inline]

Gets the y coordinate of the bullet.

Parameters:

No parameters

Returns:

Integer value

5.4.2.3 void bulletType::setbX (int *b*) [inline]

Sets the X coordinate of the bullet to the value of the int parameter.

Parameters:

b : int value of the new X coordinate of the bullet

Returns:

No return value

5.4.2.4 void bulletType::setbY (int *a*) [inline]

Sets the Y coordinate of the bullet to the value of the int parameter.

Parameters:

a : int value of the new Y coordinate of the bullet

Returns:

No return value

5.4.3 Member Data Documentation

5.4.3.1 int bulletType::bX [private]

X coordinate of the bullet

5.4.3.2 int bulletType::bY [private]

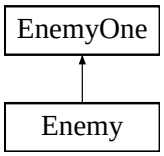
Y coordinate of the bullet

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

- bullet.h

5.5 Enemy Class Reference

`#include <Enemy.h>`Inheritance diagram for Enemy::



Public Member Functions

- Enemy ()

  Constructor.

- ~Enemy ()

  Destructor.

- void moveEnemy ()

  Defines the movement of the first enemy type to move from right to left by subtracting the 'moveSpeed' from the x coordinate. if enemy goes out of the screen on the left, it is reset to the right side.

- void moveboss ()

  Defines the movement of the second enemy type to move from right to left by subracting the 'moveSpeed' from the x coordinate.

Protected Attributes

- int health

5.5.1 Constructor & Destructor Documentation

5.5.1.1 Enemy::Enemy ()

Constructor.

Parameters:

No parameters

Returns:

No return value

5.5.1.2 Enemy::~Enemy ()

Destructor.

Parameters:

No parameters

Returns:

No return value

5.5.2 Member Function Documentation

5.5.2.1 void Enemy::moveboss ()

Defines the movement of the second enemy type to move from right to left by subracting the 'moveSpeed' from the x coordinate.

Parameters:

No parameters

Returns:

No return value

5.5.2.2 void Enemy::moveEnemy ()

Defines the movement of the first enemy type to move from right to left by subtracting the 'moveSpeed' from the x coordinate. if enemy goes out of the screen on the left, it is reset to the right side.

Parameters:

No parameters

Returns:

No return value

5.5.3 Member Data Documentation

5.5.3.1 int Enemy::health [protected]

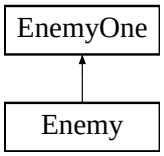
the health of the enemy which is set to 0 by the constructor

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

- Enemy.h
- Enemy.cc

5.6 EnemyOne Class Reference

`#include <EnemyOne.h>`Inheritance diagram for EnemyOne::



Public Member Functions

- EnemyOne ()

  Constructor : Sets the images for the two enemy types to NULL before they are actually initialized.

- virtual ~EnemyOne ()

  Destructor : Destroys any enemy type created dynamically.

- void addEnemy (int, int)

  Creates an Enemy object dynamically at the coordinates given by b & s which represent the x and y coordinates respectively.

- void setEnemy (std::string name)

  Loads the image name in the parameter list onto the Ebitmap object.

- ALLEGRO_BITMAP * getEnemy ()

  Getter function to get the set Enemy image.

- void drawE ()

  A draw function that draws the image loaded onto the enemy bitmap.

Public Attributes

- list< Enemytype * > Elist

Protected Attributes

- ALLEGRO_BITMAP * Ebitmap
- ALLEGRO_BITMAP * Ebitmap2

5.6.1 Constructor & Destructor Documentation

5.6.1.1 EnemyOne::EnemyOne ()

Constructor : Sets the images for the two enemy types to NULL before they are actually initialized.

Parameters:

No parameters

Returns:

No return value

5.6.1.2 EnemyOne::~EnemyOne () [virtual]

Destructor : Destroys any enemy type created dynamically.

Parameters:

No parameters

Returns:

No return value

5.6.2 Member Function Documentation

5.6.2.1 void EnemyOne::addEnemy (int *b*, int *s*)

Creates an Enemy object dynamically at the coordinates given by b & s which represent the x and y coordinates respectively.

Parameters:

b : x coordinate

s : y coordinate

Returns:

No return value

5.6.2.2 void EnemyOne::drawE ()

A draw function that draws the image loaded onto the enemy bitmap.

Parameters:

No parameters

Returns:

No return value

5.6.2.3 ALLEGRO_BITMAP * EnemyOne::getEnemy ()

Getter function to get the set Enemy image.

Parameters:

No parameters

Returns:

ALLEGRO_BITMAP type : the enemy image

5.6.2.4 void EnemyOne::setEnemy (std::string *name*)

Loads the image name in the parameter list onto the Ebitmap object.

Parameters:

name : the name of the image as stored in the folder

Returns:

No return value

5.6.3 Member Data Documentation

5.6.3.1 ALLEGRO_BITMAP* EnemyOne::Ebitmap [protected]

The first enemy type

5.6.3.2 ALLEGRO_BITMAP* EnemyOne::Ebitmap2 [protected]

The second enemy type

5.6.3.3 list<Enemytype*> EnemyOne::Elist

A list that allows for the creation of multiple enemies dynamically

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

- EnemyOne.h
- EnemyOne.cc

5.7 Enemytype Class Reference

`#include <EnemyOne.h>`

Public Member Functions

- int getX ()

  Getter function for the X coordinate of the enemy.

- int getY ()

  Getter function for the Y coordinate of the enemy.

- void setX (int a)

  Setter function which sets the x coordinate of the enemy.

- void setY (int b)

  Setter function which sets the Y coordinate of the enemy.

- int getMS ()

  a getter function which gets the set speed of the enemy

- Enemytype (int a, int s)

  Constructor : Creates the enemy to come from the right side of the screen(the x coordinate is automatically set to 640) and sets the speed at which they move.

Private Attributes

- int x
- int y
- int moveSpeed

5.7.1 Constructor & Destructor Documentation

5.7.1.1 Enemytype::Enemytype (int *a*, int *s*) `[inline]`

Constructor : Creates the enemy to come from the right side of the screen(the x coordinate is automatically set to 640) and sets the speed at which they move.

Parameters:

a,: the y coordinate of the enemy

s,: the speed at which the enemies move

Returns:

No return value

5.7.2 Member Function Documentation

5.7.2.1 int Enemytype::getMS () [inline]

a getter function which gets the set speed of the enemy

Parameters:

No parameters

Returns:

moveSpeed : the speed of the enemy

5.7.2.2 int Enemytype::getX () [inline]

Getter function for the X coordinate of the enemy.

Parameters:

No parameters

Returns:

No return value

5.7.2.3 int Enemytype::getY () [inline]

Getter function for the Y coordinate of the enemy.

Parameters:

No parameters

Returns:

int : the y coordinate of the enemy

5.7.2.4 void Enemytype::setX (int *a*) [inline]

Setter function which sets the x coordinate of the enemy.

Parameters:

a : the value which the X coordinate of the enemy will be set to

Returns:

No return value

5.7.2.5 void Enemytype::setY (int *b*) [inline]

Setter function which sets the Y coordinate of the enemy.

Parameters:

b : the value which the Y coordinate of the enemy will be set to

Returns:

No return value

5.7.3 Member Data Documentation

5.7.3.1 int Enemytype::moveSpeed [private]

the speed at which the enemy object is moving

5.7.3.2 int Enemytype::x [private]

the x coordinate of the enemy object

5.7.3.3 int Enemytype::y [private]

the y coordinate of the enemy object

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

- EnemyOne.h

5.8 Keyboard Class Reference

`#include <Keyboard.h>`

Public Member Functions

- Keyboard ()
- ~Keyboard ()

Public Attributes

- bool key [6]

5.8.1 Constructor & Destructor Documentation

5.8.1.1 Keyboard::Keyboard ()

5.8.1.2 Keyboard::~Keyboard ()

5.8.2 Member Data Documentation

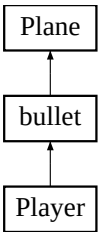
5.8.2.1 bool Keyboard::key[6]

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

- Keyboard.h
- Keyboard.cc

5.9 Plane Class Reference

`#include <Plane.h>`Inheritance diagram for Plane::



Public Member Functions

- Plane ()

  Constructor.

- virtual ~Plane ()

  Destructor.

- int getX ()

  this function gets the X coordinate of the object

- int getY ()

  this function gets the Y coordinate of the object

- void setBitmap (std::string filePath)
- ALLEGRO_BITMAP * getBitmap ()

  This function is used to get the object loaded onto the bitmap object by the function 'setBitmap'.

- void initialPlane ()

  This function set the starting point of the player by setting both the x and y coordinates to 10 and 240 respectively.

- void draw ()

  this function draws the loaded bitmap to the screen on the point provided by the x and y coordinates provided

Protected Attributes

- ALLEGRO_BITMAP * bitmap
- int x
- int y

5.9.1 Constructor & Destructor Documentation

5.9.1.1 Plane::Plane ()

Constructor.

Parameters:

This function doesn't take any parameters

Returns:

No return value

5.9.1.2 Plane::∼Plane () `[virtual]`

Destructor.

Parameters:

this function doesn't take any parameters

Returns:

no return value

5.9.2 Member Function Documentation

5.9.2.1 void Plane::draw ()

this function draws the loaded bitmap to the screen on the point provided by the x and y coordinates provided

Parameters:

takes no parameters

Returns:

No return value

5.9.2.2 ALLEGRO_BITMAP ∗ Plane::getBitmap ()

This function is used to get the object loaded onto the bitmap object by the function 'setBitmap'.

Parameters:

This function takes in no parameters

Returns:

This function returns a bitmap object

5.9.2.3 int Plane::getX ()

this function gets the X coordinate of the object

Parameters:

this function takes no parameters

Returns:

this function returns the X coordinate of the object

5.9.2.4 int Plane::getY ()

this function gets the Y coordinate of the object

Parameters:

this function takes no parameters

Returns:

this function returns the Y coordinate of the object

5.9.2.5 void Plane::initialPlane ()

This function set the starting point of the player by setting both the x and y coordinates to 10 and 240 respectively.

Parameters:

This function takes in no parameters

Returns:

No return value

5.9.2.6 void Plane::setBitmap (std::string *filePath*)

5.9.3 Member Data Documentation

5.9.3.1 ALLEGRO_BITMAP* Plane::bitmap [protected]

The picture of the player

5.9.3.2 int Plane::x [protected]

x coordinate

5.9.3.3 int Plane::y [protected]

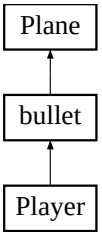
y coordinate

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

- Plane.h
- Plane.cc

5.10 Player Class Reference

`#include <Player.h>`Inheritance diagram for Player::



Public Member Functions

- Player ()

  Constructor.

- ~Player ()

  Destructor.

- void doLogic (Keyboard keyboard)

  Defines the movement key controls for the player object so it can be moved using a keyboard.

- void moveBullet ()

  This function initializes the movement of the bullet through the display.

Private Attributes

- int health
- int moveSpeed
- int bMoveSpeed

5.10.1 Constructor & Destructor Documentation

5.10.1.1 Player::Player ()

Constructor.

Parameters:

No parameters

Returns:

No return value

5.10.1.2 Player::~Player ()

Destructor.

Parameters:

No parameters

Returns:

No return value

5.10.2 Member Function Documentation

5.10.2.1 Void Player::doLogic (Keyboard *keyboard*)

Defines the movement key controls for the player object so it can be moved using a keyboard.

Parameters:

This function takes in a Keyboard object

Returns:

No return value

5.10.2.2 void Player::moveBullet ()

This function initializes the movement of the bullet through the display.

Parameters:

This function takes no parameters

Returns:

No return value

5.10.3 Member Data Documentation

5.10.3.1 int Player::bMoveSpeed `[private]`

speed of the bullet

5.10.3.2 int Player::health `[private]`

5.10.3.3 int Player::moveSpeed `[private]`

Rate at which the player can be moved

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

- Player.h
- Player.cc

Chapter 6

File Documentation

6.1 Allegro.cc File Reference

Implementation of the Allegro class. `#include "Allegro.h"`

`#include <stdio.h>`

Functions

- void InitBackground (Background &back, float x, float y, float velx, float vely, int width, int height, int dirX, int dirY, ALLEGRO_BITMAP *image)

  This function initializes the various background images : acts like a constructor for the background struct.

- void UpdateBackground (Background &back)

  Initializes the speed at which the background moves.

- void DrawBackground (Background &back)

  Draws the background and makes sure it is always cycles through.

Variables

- const int WIDTH = 640
- const int HEIGHT = 480
- Background BG
- Background MG
- Background FG
- ALLEGRO_BITMAP * bgImage = NULL
- ALLEGRO_BITMAP * mgImage = NULL
- ALLEGRO_BITMAP * fgImage = NULL
- ALLEGRO_SAMPLE * shot = NULL
- ALLEGRO_SAMPLE * boom = NULL
- ALLEGRO_SAMPLE * song = NULL
- ALLEGRO_SAMPLE_INSTANCE * songInstance = NULL

6.1.1 Detailed Description

Implementation of the Allegro class. This contains the implementation of member variables and functions of the Allegro class

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi.

Bug

No known bugs.

6.1.2 Function Documentation

6.1.2.1 void DrawBackground (Background & *back*)

Draws the background and makes sure it is always cycles through.

Parameters:

&back : Background object with the properties of the background struct

Returns:

No return value

6.1.2.2 void InitBackground (Background & *back*, float *x*, float *y*, float *velx*, float *vely*, int *width*, int *height*, int *dirX*, int *dirY*, ALLEGRO_BITMAP * *image*)

This function initializes the various background images : acts like a constructor for the background struct.

Parameters:

&back : a background object with the properties of the background struct

x : x coordinate

y : y coordinate

velx : velocity of the background moving in the x direction

vely : velocity of the background moving in the y direction

width : width of the background

height : height of the background

dirX : direction of x

dirY : direction of y

ALLEGRO_BITMAP : the background image

Returns:

No return value

6.1.2.3 void UpdateBackground (Background & *back*)

Initializes the speed at which the background moves.

Parameters:

&back : Background object with the properties of the background struct

Returns:

No return value

6.1.3 Variable Documentation

6.1.3.1 Background BG

6.1.3.2 ALLEGRO_BITMAP* bgImage = NULL

6.1.3.3 ALLEGRO_SAMPLE* boom = NULL

6.1.3.4 Background FG

6.1.3.5 ALLEGRO_BITMAP* fgImage = NULL

6.1.3.6 const int HEIGHT = 480

6.1.3.7 Background MG

6.1.3.8 ALLEGRO_BITMAP* mgImage = NULL

6.1.3.9 ALLEGRO_SAMPLE* shot = NULL

6.1.3.10 ALLEGRO_SAMPLE* song = NULL

6.1.3.11 ALLEGRO_SAMPLE_INSTANCE* songInstance = NULL

6.1.3.12 const int WIDTH = 640

6.2 Allegro.h File Reference

Definition of the Allegro class. `#include <allegro5/allegro.h>`

```
#include <allegro5/allegro_image.h>
#include <allegro5/allegro_primitives.h>
#include <allegro5/allegro_font.h>
#include <allegro5/allegro_ttf.h>
#include <allegro5/allegro_audio.h>
#include <allegro5/allegro_acodec.h>
#include "Player.h"
#include "Keyboard.h"
#include "Enemy.h"
```

Classes

- class Allegro
- struct Background

Enumerations

- enum States { TITLE, PLAY, LOST }

6.2.1 Detailed Description

Definition of the Allegro class. This contains the public and private member variables and functions of the Allegro class

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi.

Bug

No known bugs.

6.2.2 Enumeration Type Documentation

6.2.2.1 enum States

Enumerator:

TITLE

PLAY

LOST

6.3 bullet.cc File Reference

Implementation of the bullet class. `#include "bullet.h"`

6.3.1 Detailed Description

Implementation of the bullet class. This contains the implementation of member variables and functions of the bullet class

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi.

Bug

No known bugs.

6.4 bullet.h File Reference

Definition of the bulleyType and bullet class (which inherits from the Plane class). This contains the public and private member variables and functions of the bullet class. `#include <string>`

```
#include <allegro5/allegro.h>
#include <allegro5/allegro_image.h>
#include <allegro5/allegro_primitives.h>
#include "Plane.h"
#include <list>
```

Classes

- class bulletType
- class bullet

6.4.1 Detailed Description

Definition of the bulleyType and bullet class (which inherits from the Plane class). This contains the public and private member variables and functions of the bullet class.

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi.

Bug

No known bugs.

6.5 Enemy.cc File Reference

Implementation of the Enemy class. `#include "Enemy.h"`

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
```

6.5.1 Detailed Description

Implementation of the Enemy class. This contains the implementation of member variables and functions of the Enemy class

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi.

Bug

No known bugs.

6.6 Enemy.h File Reference

Definition of the Enemy class. `#include "EnemyOne.h"`

Classes

- class Enemy

6.6.1 Detailed Description

Definition of the Enemy class. This contains the public and private member variables and functions of the Enemy class

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi.

Bug

No known bugs.

6.7 EnemyOne.cc File Reference

Implementation of the Enemytype and EnemyOne classes. `#include "EnemyOne.h"`

`#include <allegro5/allegro_primitives.h>`

6.7.1 Detailed Description

Implementation of the Enemytype and EnemyOne classes. This contains the implementation of member variables and functions of the Enemytype and EnemyOne classes.

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi

Bug

No known bugs.

6.8 EnemyOne.h File Reference

Definition of the Enemytype and EnemyOne classes. `#include <string>`

```
#include <allegro5/allegro.h>
#include <allegro5/allegro_image.h>
#include <stdlib.h>
#include <time.h>
#include <iostream>
#include <list>
```

Classes

- class Enemytype
- class EnemyOne

6.8.1 Detailed Description

Definition of the Enemytype and EnemyOne classes. This contains thepublic and private member variables and functions of the Enemytype and EnemyOne classes

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi

Bug

No known bugs.

6.9 Keyboard.cc File Reference

Implementation of the Keyboard class. `#include "Keyboard.h"`

6.9.1 Detailed Description

Implementation of the Keyboard class. This contains the implementation of member variables and functions of the Keyboard class

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi.

Bug

No known bugs.

6.10 Keyboard.h File Reference

Definition of the Keyboard Class.

Classes

- class Keyboard

Enumerations

- enum keys {
  UP, LEFT, DOWN, RIGHT,
  SPACE, ENTER }

6.10.1 Detailed Description

Definition of the Keyboard Class. This contains the public and private member variables and functions of the Keyboard class

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi.

Bug

No known bugs.

6.10.2 Enumeration Type Documentation

6.10.2.1 enum keys

Enumerator:

UP

LEFT

DOWN

RIGHT

SPACE

ENTER

6.11 main.cc File Reference

This is the main function that calls the functions and initiates the gameplay. `#include "Allegro.h"`

Functions

- int main ()

6.11.1 Detailed Description

This is the main function that calls the functions and initiates the gameplay.

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi.

Bug

No known bugs.

6.11.2 Function Documentation

6.11.2.1 int main ()

6.12 Plane.cc File Reference

Implementation of the Plane class. `#include "Plane.h"`

`#include <allegro5/allegro_primitives.h>`

6.12.1 Detailed Description

Implementation of the Plane class. This contains the implementation of member variables and functions of the Plane class

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi.

Bug

No known bugs.

6.13 Plane.h File Reference

Definition of the Plane class. `#include <string>`

`#include <allegro5/allegro.h>`

`#include <allegro5/allegro_image.h>`

Classes

- class Plane

6.13.1 Detailed Description

Definition of the Plane class. This contains the public and private member variables and functions of the Plane class.

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi

Bug

No known bugs.

6.14 Player.cc File Reference

Implementation of the Player class. `#include "Player.h"`

6.14.1 Detailed Description

Implementation of the Player class. This contains the implementation of member variables and functions of the Player class

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi.

Bug

No known bugs.

6.15 Player.h File Reference

Definition of the Player class which inherits from the bullet class. `#include "Keyboard.h"`

`#include "bullet.h"`

Classes

- class Player

6.15.1 Detailed Description

Definition of the Player class which inherits from the bullet class. This contains the public and private member variables and functions of the Player class

Author:

Wang Kangning, Jefferson Sylva-Iriogbe and Yuhai Shi

Bug

No known bugs.

Index