# Arch Game Engine

0.2

Generated by Doxygen 1.8.11

# Contents

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

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

# Chapter 2

# Class Index

## 2.1 Class List

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

# Chapter 3

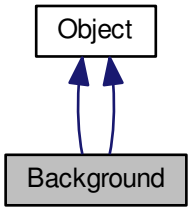
# Class Documentation

## 3.1 Background Class Reference

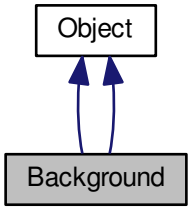
Object that is a background image that covers the screen.

`#include <arch.h>`

Inheritance diagram for Background:




Collaboration diagram for Background:



### Public Member Functions

- void setBackground (string file, int w, int h, SDL_Renderer *ren)

  *Sets the background with a path to the file name, the width and height of the screen, and the renderer.*
- void setBackground (string file, int w, int h, SDL_Renderer *ren)

  *Sets the background with a path to the file name, the width and height of the screen, and the renderer.*

#### 3.1.1 Detailed Description

Object that is a background image that covers the screen.

Definition at line 7 of file arch.h.

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

- arch.h
- background.h
- background.cpp

## 3.2 Collision Class Reference

Class used for calculating different types of collision between given Objects.

`#include <arch.h>`

### Public Member Functions

- bool isTouching (Object a, Object b)

  *Check if two objects are touching.*
- bool outOfBoundsOf (Object a, Object b)

  *Check if two object are not touching.*
- bool isAbove (Object a, Object b)

  *Check if the first object is above the second object.*
- bool isBelow (Object a, Object b)

  *Check if the first object is below the second object.*
- bool isRightOf (Object a, Object b)

  *Check if the first object is to the right of the second object.*
- bool isLeftOf (Object a, Object b)

  *Check if the first object is to the left of the second object.*
- bool isTouching (Object a, Object b)

  *Check if two objects are touching.*
- bool outOfBoundsOf (Object a, Object b)

  *Check if two object are not touching.*
- bool isAbove (Object a, Object b)

  *Check if the first object is above the second object.*
- bool isBelow (Object a, Object b)

  *Check if the first object is below the second object.*
- bool isRightOf (Object a, Object b)

  *Check if the first object is to the right of the second object.*
- bool isLeftOf (Object a, Object b)

  *Check if the first object is to the left of the second object.*

### 3.2.1 Detailed Description

Class used for calculating different types of collision between given Objects.

Definition at line 23 of file arch.h.

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

- arch.h
- collision.h
- collision.cpp

## 3.3 Engine Class Reference

Class for declaring an engine, which does basic SDL commands like creating the window and renderer.

`#include <arch.h>`

### Public Member Functions

- ~Engine ()

  *Decontructs renderer and window and then quits SDL.*
- SDL_Renderer * init (string s, const int &w, const int &h, int flag)

  *Create a window with a given name, width, height, and anyother SDL_Window flags.*
- SDL_Renderer * init (string s, const int &w, const int &h, int flag, int it)

  *Create a window with a given name, width, height, SDL_Window flags, and specified SDL_Init flags.*
- SDL_Renderer * init (string s, const int &w, const int &h, int flag, int x, int y)

  *Create a window with a given name, width, height, SDL_Window flags, and specified x and y coordinate.*
- SDL_Renderer * init (string s, const int &w, const int &h, int flag, int x, int y, int it)

  *Create a window with a given name, width, height, SDL_Window flags, specified x and y coordinate, and SDL_Init flags.*
- void setName (string s)

  *Set window name.*
- void setPos (int x, int y)

  *Set window position.*
- void setSize (int w, int y)

  *Set window size.*
- SDL_Renderer * getRenderer ()

  *Returns screen renderer.*
- void setColor (Uint8 r, Uint8 g, Uint8 b)

  *Sets SDL color.*
- void loopStart ()

  *Call this at the beginning of a loop to initilaize the loop.*
- void render ()

  *Call this at the end of the game loop to render.*
- bool FPS () const

  *Get fps.*
- void update ()

  *Update loop time.*

- void setBackground (Background b)

  *Set background.*
- void setBackground (string filename)

  *Set background with filename.*
- Background getBackground () const

  *Get background.*
- void drawBackground ()

  *Draw background.*
- void draw (Object obj)

  *Draw an object on the screen.*
- void draw (vector< Object > objs)

  *Draw a vector of Objects.*
- void draw (Object obj, int key)

  *Draw an object with a pass key before/during splash.*
- void draw (Level lvl)

  *Draw the level.*
- void splash ()

  *Calls splashscreen at the beginning of the game. This is automatically called unless deactivated.*
- void bypassSplash (int key)

  *Deactives the splashscreen, requires key.*
- bool hasSplashed ()

  *Check if the splashscreen has occured.*
- bool runCustomSplash ()

  *Run custom splashscreen. This is automatically called after splash if there is a custom splashscreen.*
- void customSplash (string file, double time, int w, int h)

  *Create a custom game splashscreen to be shown after the engine splashscreen by passing in the path to the image, the duration for it be displayed, and the size of the image.*
- void debugMode (bool d)

  *Active debugger with Boolean.*
- void **hideMouse** ()
- void **showMouse** ()
- ~Engine ()

  *Decontructs renderer and window and then quits SDL.*
- SDL_Renderer * init (string s, const int &w, const int &h, int flag)

  *Create a window with a given name, width, height, and anyother SDL_Window flags.*
- SDL_Renderer * init (string s, const int &w, const int &h, int flag, int it)

  *Create a window with a given name, width, height, SDL_Window flags, and specified SDL_Init flags.*
- SDL_Renderer * init (string s, const int &w, const int &h, int flag, int x, int y)

  *Create a window with a given name, width, height, SDL_Window flags, and specified x and y coordinate.*
- SDL_Renderer * init (string s, const int &w, const int &h, int flag, int x, int y, int it)

  *Create a window with a given name, width, height, SDL_Window flags, specified x and y coordinate, and SDL_Init flags.*
- void setName (string s)

  *Set window name.*
- void setPos (int x, int y)

  *Set window position.*
- void setSize (int w, int y)

  *Set window size.*
- SDL_Renderer * getRenderer ()

  *Returns screen renderer.*
- void setColor (Uint8 r, Uint8 g, Uint8 b)

*Sets SDL color.*

- void loopStart ()

  *Call this at the beginning of a loop to initilaize the loop.*
- void render ()

  *Call this at the end of the game loop to render.*
- bool FPS () const

  *Get fps.*
- void update ()

  *Update loop time.*
- void setBackground (Background b)

  *Set background.*
- void setBackground (string filename)

  *Set background with filename.*
- Background getBackground () const

  *Get background.*
- void drawBackground ()

  *Draw background.*
- void draw (Object obj)

  *Draw an object on the screen.*
- void draw (vector< Object > objs)

  *Draw a vector of Objects.*
- void draw (Object obj, int key)

  *Draw an object with a pass key before/during splash.*
- void draw (Level lvl)

  *Draw the level.*
- void splash ()

  *Calls splashscreen at the beginning of the game. This is automatically called unless deactivated.*
- void bypassSplash (int key)

  *Deactives the splashscreen, requires key.*
- bool hasSplashed ()

  *Check if the splashscreen has occured.*
- bool runCustomSplash ()

  *Run custom splashscreen. This is automatically called after splash if there is a custom splashscreen.*
- void customSplash (string file, double time, int w, int h)

  *Create a custom game splashscreen to be shown after the engine splashscreen by passing in the path to the image, the duration for it be displayed, and the size of the image.*
- void debugMode (bool d)

  *Active debugger with Boolean.*
- void **hideMouse** ()
- void **showMouse** ()

### 3.3.1 Detailed Description

Class for declaring an engine, which does basic SDL commands like creating the window and renderer.

Definition at line 70 of file arch.h.

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

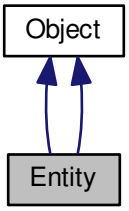
- arch.h
- engine.h
- engine.cpp

## 3.4 Entity Class Reference

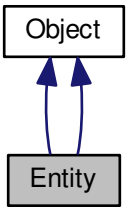
Class for storing health, emotion, team, etc. of an Object.

`#include <arch.h>`

Inheritance diagram for Entity:




Collaboration diagram for Entity:




### Public Member Functions

- double getHealth () const

  *Get Entity's health.*
- void setHealth (double h)

  *Set the Entity's health. If the health is higher then the max health it will set it to the max health.*
- double getMaxHealth () const

  *Get max health.*
- void setMaxHealth (double mh)

  *Set max health.*
- void damage (double d)

  *Deal damage. Subtracted from health. If health is less then zero it kills the entity.*
- void heal (double h)

*Give health to the Entity.*

- int getEmotion () const

  *Get current emotion state.*
- void setEmotion (int e)

  *Set current emotion state.*
- int getTeam () const

  *Get Entity's team.*
- void setTeam (int t)

  *Set Entity's team.*
- bool isActive () const

  *Check if Entity is active.*
- void kill ()

  *Sets health to zero and deactives the Entity.*
- void deactivate ()

  *Sets active to false.*
- void activate ()

  *Sets active to true.*
- void checkDisplayable (Object screen)

  *Checks if an the Entity is in the current screen by passing the screen to it.*
- SDL_Rect getDetect () const

  *Returns the detection radius.*
- void setDetect (SDL_Rect d)

  *Sets the detection with another SDL_Rect.*
- void setDetectRange (int r)

  *Sets the detection radius with a single given distance.*
- void setDetectRange (int w, int h)

  *Sets the detection radius with two given distances in both directions.*
- double getHealth () const

  *Get Entity's health.*
- void setHealth (double h)

  *Set the Entity's health. If the health is higher then the max health it will set it to the max health.*
- double getMaxHealth () const

  *Get max health.*
- void setMaxHealth (double mh)

  *Set max health.*
- void damage (double d)

  *Deal damage. Subtracted from health. If health is less then zero it kills the entity.*
- void heal (double h)

  *Give health to the Entity.*
- int getEmotion () const

  *Get current emotion state.*
- void setEmotion (int e)

  *Set current emotion state.*
- int getTeam () const

  *Get Entity's team.*
- void setTeam (int t)

  *Set Entity's team.*
- bool isActive () const

  *Check if Entity is active.*
- void kill ()

  *Sets health to zero and deactives the Entity.*

- void deactivate ()

  *Sets active to false.*
- void activate ()

  *Sets active to true.*
- void checkDisplayable (Object screen)

  *Checks if an the Entity is in the current screen by passing the screen to it.*
- SDL_Rect getDetect () const

  *Returns the detection radius.*
- void setDetect (SDL_Rect d)

  *Sets the detection with another SDL_Rect.*
- void setDetectRange (int r)

  *Sets the detection radius with a single given distance.*
- void setDetectRange (int w, int h)

  *Sets the detection radius with two given distances in both directions.*

### 3.4.1 Detailed Description

Class for storing health, emotion, team, etc. of an Object.

Definition at line 157 of file arch.h.

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

- arch.h
- entity.h
- entity.cpp

## 3.5 GameState Class Reference

### Public Member Functions

- int **getGameState** ()
- void **setGameState** (int)
- int **getGameState** ()
- void **setGameState** (int)

### Public Attributes

- int **SPLASH** = 0
- int **MENU** = 1
- int **INGAME** = 2
- int **GAMEOVER** = 3
- int **PAUSE** = 4

### 3.5.1 Detailed Description

Definition at line 214 of file arch.h.

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

- arch.h
- gamestate.h
- gamestate.cpp

## 3.6 Image Class Reference

Class for loading in SDL Textures.

`#include <arch.h>`

### Public Member Functions

- void loadImage (string file, SDL_Renderer *ren)
  *Load in either a BMP or PNG file with the path and renderer.*
- void loadPNG (string file, SDL_Renderer *ren)
  *Load in a PNG image with the path to the PNG file and the renderer.*
- void loadBMP (string file, SDL_Renderer *ren)
  *Load in a BMP image with the path to the BMP file and the renderer.*
- SDL_Texture * getTexture ()
  *Get SDL_Texture.*
- void setImage (SDL_Texture *t)
  *Set new, preloaded texture, to Image.*
- string getFile () const
  *Get path file of the image.*
- void setFile (string f)
  *Set path file to the image.*
- void loadImage (string file, SDL_Renderer *ren)
  *Load in either a BMP or PNG file with the path and renderer.*
- void loadPNG (string file, SDL_Renderer *ren)
  *Load in a PNG image with the path to the PNG file and the renderer.*
- void loadBMP (string file, SDL_Renderer *ren)
  *Load in a BMP image with the path to the BMP file and the renderer.*
- SDL_Texture * getTexture ()
  *Get SDL_Texture.*
- void setImage (SDL_Texture *t)
  *Set new, preloaded texture, to Image.*
- string getFile () const
  *Get path file of the image.*
- void setFile (string f)
  *Set path file to the image.*

### 3.6.1 Detailed Description

Class for loading in SDL Textures.

Definition at line 236 of file arch.h.

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

- arch.h
- image.h
- image.cpp

## 3.7 Input Class Reference

Class for checking and storing keyboard and mouse input.

`#include <arch.h>`

### Public Member Functions

- void logPress ()
  *Log all current keys and buttons being pressed.*
- bool checkKey (int k)
  *Check if a key has been pressed using a given key from this class. Ex: Input i; i.checkKey(i.up);.*
- bool reset ()
  *Reset all pressed keystrokes and other inputs to false. Automatically down at the beginning of each logPress().*
- int **getMouseX** () const
- int **getMouseY** () const
- void logPress ()
  *Log all current keys and buttons being pressed.*
- bool checkKey (int k)
  *Check if a key has been pressed using a given key from this class. Ex: Input i; i.checkKey(i.up);.*
- bool reset ()
  *Reset all pressed keystrokes and other inputs to false. Automatically down at the beginning of each logPress().*
- int **getMouseX** () const
- int **getMouseY** () const

### Public Attributes

- int left
  *Log ID for left.*
- int right
  *Log ID for right.*
- int up
  *Log ID for up.*
- int down
  *Log ID for down.*
- int q

*Log ID for q.*

- int w

  *Log ID for w.*

- int e

  *Log ID for e.*

- int r

  *Log ID for r.*

- int t

  *Log ID for t.*

- int y

  *Log ID for y.*

- int u

  *Log ID for u.*

- int i

  *Log ID for i.*

- int o

  *Log ID for o.*

- int p

  *Log ID for p.*

- int a

  *Log ID for a.*

- int s

  *Log ID for s.*

- int d

  *Log ID for d.*

- int f

  *Log ID for f.*

- int g

  *Log ID for g.*

- int h

  *Log ID for h.*

- int j

  *Log ID for j.*

- int k

  *Log ID for k.*

- int l

  *Log ID for l.*

- int z

  *Log ID for z.*

- int x

  *Log ID for x.*

- int c

  *Log ID for c.*

- int v

  *Log ID for v.*

- int b

  *Log ID for b.*

- int n

  *Log ID for n.*

- int m

  *Log ID for m.*

- int lshift

  *Log ID for left shift.*
- int rshift

  *Log ID for right shift.*
- int shift

  *Shift ID for shift.*
- int quit

  *Log ID for quit.*
- int esc

  *Log ID for esc.*
- int mouseleft

  *Log ID for left mouse click.*
- int mousemiddle

  *Log ID for middle mouse click.*
- int mouseright

  *Log ID for right mouse click.*
- int mouseup

  *Log ID for scroll up on mouse wheel.*
- int mousedown

  *Log ID for scroll down on mouse wheel.*

### 3.7.1 Detailed Description

Class for checking and storing keyboard and mouse input.

Definition at line 268 of file arch.h.

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

- arch.h
- input.h
- input.cpp

## 3.8 Level Class Reference

This class stores a Stage and Objects and can move them and display them.

`#include <level.h>`

## Public Member Functions

- void create ()

  *Create the Level based on the given stage.*
- void setStage (Stage s)

  *Give a Stage to the Level.*
- void setStage (Map m, Tileset t)

  *Create a Stage for the Level by giving a Map and a Tileset.*
- void setScale (int w, int h)

  *Scale the Level by giving it the width and height to scale by.*
- void setScale (int s)

  *Scale the Level by giving it a single integer to scale by.*
- void calcPos ()

  *Calculate the position of the level based on coordinates.*
- vector< Tile > getTilesToRender ()

  *Return the Tiles that are currently on the screen.*
- vector< Object > getObjectsToRender ()

  *Return the Objects that are currently on the screen.*
- vector< Entity > getEntitiesToRender ()

  *Return the Entities that are currently on the screen.*
- void move (int mx, int my)

  *Move the screen by passing in how much to move on the x and y coordinates.*
- void **moveEntity** (int id, int mx, int my)
- void setCoord (double x, double y)

  *Set the coordinate for the screen with a given x and y.*
- void setX (double x)

  *Set the x coordinate.*
- void setY (double y)

  *Set the y coordinate.*
- double getX () const

  *Get the x coordinate.*
- double getY () const

  *Get the y coordinate.*
- Object **getScreen** () const
- void setScreenSize (int w, int h)

  *Set the size of the screen by passing in the width and height.*
- void setPrecise (bool p)

  *Active precise if you want the coordinates in a map file to go to that exact pixel, or leave it off if you want it to go to that Tile.*
- void addObject (Object o)

  *Add Object to Level.*
- void addObject (vector< Object > o)

  *Add vector of Objects to Level.*
- int addEntity (Entity e)

  *Add Entity to Level.*
- void addEntity (vector< Entity > e)

  *Add vector of Entity's to Level.*
- int setMainEntity (Entity e)

  *Set main Entity.*
- int setMainEntity (int m)

  *Tell Level which one Entity is the main one.*

- void **setCameraMargin** (int wm, int hm)
- void **centerCamera** (int percentage)
- void **setLensMargin** (int wn, int hm)
- void **centerLens** (int percentage)
- Object **getCamera** ()
- Object **getLens** ()

### 3.8.1 Detailed Description

This class stores a Stage and Objects and can move them and display them.

Definition at line 10 of file level.h.

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

- level.h
- level.cpp

## 3.9 Map Class Reference

This class takes in a file and loads it in for the map.

```
#include <arch.h>
```

### Public Member Functions

- void loadMap (string filename)

  *Read in map file with given path to the file.*
- int getX () const

  *Get the start x coordinate found in the file.*
- int getY () const

  *Get the start y coordinate found in the file.*
- vector< vector< int > > getMap () const

  *Get the vector of integers found in the file.*
- void loadMap (string filename)

  *Read in map file with given path to the file.*
- int getX () const

  *Get the start x coordinate found in the file.*
- int getY () const

  *Get the start y coordinate found in the file.*
- vector< vector< int > > getMap () const

  *Get the vector of integers found in the file.*

### 3.9.1 Detailed Description

This class takes in a file and loads it in for the map.

Definition at line 421 of file arch.h.

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

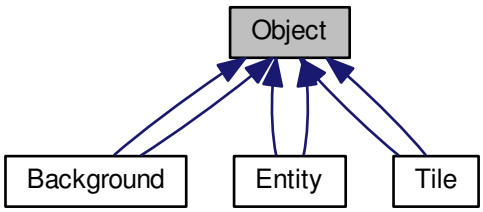
- arch.h
- map.h
- map.cpp

## 3.10 Object Class Reference

This class stores information for an Object in the game.

`#include <arch.h>`

Inheritance diagram for Object:




### Public Member Functions

- void setDisplayable (bool d)

  *This sets if you want the Object to visible on the screen by passing in a boolean.*
- bool isDisplayable (Object screen)

  *Check if the Object is displayable by seeing if it is in a given screen.*
- virtual void checkDisplayable (Object screen)

  *Checks if the Object is in the given screen.*
- void setCoord (double x, double y)

  *Set the coordinate of the Object with a given x and y.*
- void setX (double sx)

  *Set the x coordinate with a given x.*
- void setY (double sy)

  *Set the y coordinate with a given y.*
- void move (double x, double y)

  *Move along the x and y coordinate with a given x and y amount.*

- void moveX (double mx)

  *Move along the x coordinate with a given x amount.*
- void moveY (double my)

  *Move along the y coordinate with a given y amount.*
- double getX () const

  *Get the current x coordinate.*
- double getY () const

  *Get the current y coordinate.*
- Image getImage () const

  *Get the Object's Image.*
- void setImage (Image i)

  *Set the Object's Image with a given Image.*
- void setImage (string file, SDL_Renderer *ren)

  *Give the path and renderer to create the Object's Image.*
- double getAngle () const

  *Get the Object's angle.*
- void setAngle (double a)

  *Set the angle.*
- void center (int w, int h)

  *Center the Object based on a width and height.*
- SDL_Rect getFrame () const

  *Get the frame that the Object parses from the Image.*
- SDL_Rect getDest () const

  *Get the destination for the Object to be displayed on screen.*
- SDL_Rect getPos () const

  *Get the position of the Object in the world.*
- void setFrame (SDL_Rect i)

  *Set the frame with a given SDL_Rect.*
- void setDest (SDL_Rect i)

  *Set the destination with a given SDL_Rect.*
- void setPos (SDL_Rect i)

  *Set the position with a given SDL_Rect.*
- void setFrame (int x, int y, int w, int h)

  *Set the frame with a given x and y coordinate and width and height.*
- void setFrameCoord (int x, int y)

  *Set the x and y coordinate of the frame.*
- void setFrameSize (int w, int h)

  *Set the size of the frame with a width and height.*
- void setFrameX (int x)

  *Set the x coordinate of the frame.*
- void setVelTo (Object o)

  *Set the object's velocity toward another object.*
- void setFrameY (int y)

  *Set the y coordinate of the frame.*
- void **setFrameW** (int w)
- void **setFrameH** (int h)
- int **getFrameX** () const
- int **getFrameY** () const
- int **getFrameW** () const
- int **getFrameH** () const
- void **setDest** (int x, int y, int w, int h)

- void **setDestCoord** (int x, int y)
- void **setDestSize** (int w, int h)
- void **setDestX** (int x)
- void **setDestY** (int y)
- void **setDestW** (int w)
- void **setDestH** (int h)
- int **getDestX** () const
- int **getDestY** () const
- int **getDestW** () const
- int **getDestH** () const
- void **setPos** (int x, int y, int w, int h)
- void **setPosCoord** (int x, int y)
- void **setPosSize** (int w, int h)
- void **setPosX** (int x)
- void **setPosY** (int y)
- void **setPosW** (int w)
- void **setPosH** (int h)
- int **getPosX** () const
- int **getPosY** () const
- int **getPosW** () const
- int **getPosH** () const
- void **moveFrame** (int x, int y)
- void **moveFrameX** (int x)
- void **moveFrameY** (int y)
- void **moveDest** (int x, int y)
- void **moveDestX** (int x)
- void **moveDestY** (int y)
- void **movePos** (int x, int y)
- void **movePosX** (int x)
- void **movePosY** (int y)
- double **getVelX** ()
- double **getVelY** ()
- void **setVelX** (double vx)
- void **setVelY** (double vy)
- double **getSpeed** ()
- void **setSpeed** (double s)
- void **setName** (string s)
- string **getName** ()
- void **centerOnMouse** (Input i)
- void **centerOnPoint** (int cx, int cy)
- void setDisplayable (bool d)

  *This sets if you want the Object to visible on the screen by passing in a boolean.*
- bool isDisplayable (Object screen)

  *Check if the Object is displayable by seeing if it is in a given screen.*
- virtual void checkDisplayable (Object screen)

  *Checks if the Object is in the given screen.*
- void setCoord (double x, double y)

  *Set the coordinate of the Object with a given x and y.*
- void setX (double sx)

  *Set the x coordinate with a given x.*
- void setY (double sy)

  *Set the y coordinate with a given y.*
- void move (double x, double y)

*Move along the x and y coordinate with a given x and y amount.*

- void moveX (double mx)

  *Move along the x coordinate with a given x amount.*
- void moveY (double my)

  *Move along the y coordinate with a given y amount.*
- double getX () const

  *Get the current x coordinate.*
- double getY () const

  *Get the current y coordinate.*
- Image getImage () const

  *Get the Object's Image.*
- void setImage (Image i)

  *Set the Object's Image with a given Image.*
- void setImage (string file, SDL_Renderer *ren)

  *Give the path and renderer to create the Object's Image.*
- double getAngle () const

  *Get the Object's angle.*
- void setAngle (double a)

  *Set the angle.*
- void center (int w, int h)

  *Center the Object based on a width and height.*
- SDL_Rect getFrame () const

  *Get the frame that the Object parses from the Image.*
- SDL_Rect getDest () const

  *Get the destination for the Object to be displayed on screen.*
- SDL_Rect getPos () const

  *Get the position of the Object in the world.*
- void setFrame (SDL_Rect i)

  *Set the frame with a given SDL_Rect.*
- void setDest (SDL_Rect i)

  *Set the destination with a given SDL_Rect.*
- void setPos (SDL_Rect i)

  *Set the position with a given SDL_Rect.*
- void setFrame (int x, int y, int w, int h)

  *Set the frame with a given x and y coordinate and width and height.*
- void setFrameCoord (int x, int y)

  *Set the x and y coordinate of the frame.*
- void setFrameSize (int w, int h)

  *Set the size of the frame with a width and height.*
- void setFrameX (int x)

  *Set the x coordinate of the frame.*
- void setVelTo (Object o)

  *Set the object's velocity toward another object.*
- void setFrameY (int y)

  *Set the y coordinate of the frame.*
- void **setFrameW** (int w)
- void **setFrameH** (int h)
- int **getFrameX** () const
- int **getFrameY** () const
- int **getFrameW** () const
- int **getFrameH** () const

- void **setDest** (int x, int y, int w, int h)
- void **setDestCoord** (int x, int y)
- void **setDestSize** (int w, int h)
- void **setDestX** (int x)
- void **setDestY** (int y)
- void **setDestW** (int w)
- void **setDestH** (int h)
- int **getDestX** () const
- int **getDestY** () const
- int **getDestW** () const
- int **getDestH** () const
- void **setPos** (int x, int y, int w, int h)
- void **setPosCoord** (int x, int y)
- void **setPosSize** (int w, int h)
- void **setPosX** (int x)
- void **setPosY** (int y)
- void **setPosW** (int w)
- void **setPosH** (int h)
- int **getPosX** () const
- int **getPosY** () const
- int **getPosW** () const
- int **getPosH** () const
- void **moveFrame** (int x, int y)
- void **moveFrameX** (int x)
- void **moveFrameY** (int y)
- void **moveDest** (int x, int y)
- void **moveDestX** (int x)
- void **moveDestY** (int y)
- void **movePos** (int x, int y)
- void **movePosX** (int x)
- void **movePosY** (int y)
- double **getVelX** ()
- double **getVelY** ()
- void **setVelX** (double vx)
- void **setVelY** (double vy)
- double **getSpeed** ()
- void **setSpeed** (double s)
- void **setName** (string s)
- string **getName** ()
- void **centerOnMouse** (Input i)
- void **centerOnPoint** (int cx, int cy)

### 3.10.1 Detailed Description

This class stores information for an Object in the game.

Definition at line 450 of file arch.h.

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

- arch.h
- object.h
- object.cpp

## 3.11 Physics Class Reference

Class for doing physics functions.

`#include <arch.h>`

### Public Member Functions

- Object moveTowards (Object cur, Object des)

  *Returns modified first Object that is moving towards the second object (I THINK).*
- Object moveTowards (Object cur, Object des)

  *Returns modified first Object that is moving towards the second object (I THINK).*

### 3.11.1 Detailed Description

Class for doing physics functions.

Definition at line 592 of file arch.h.

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

- arch.h
- physics-tmp.h
- physics-tmp.cpp

## 3.12 Stage Class Reference

The Stage class stores a Map and Tileset.

`#include <arch.h>`

### Public Member Functions

- void createStage (Map m, Tileset t)

  *Create a stage by passing in a Map and Tileset.*
- void createStage (string filename, string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)

  *Create a stage by passing in the maps file, a name for the tiles, file of the tile image, the renderer, width and height of a tile, what row of the image the tiles are onem and how many tiles there are.*
- void **createStage** (string filename, string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)
- void **createStage** (string filename, int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)
- void **createStage** (string filename, int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)
- void setMap (Map m)

  *Set the Map by passing in a Map.*
- Map setMap (string filename)

*Load in a new map by passing in the map file.*

- Map getMap () const

  *Get the Map.*

- void setTileset (Tileset t)

  *Set the Tileset with a given Tileset.*

- Tileset **setTileset** (string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)
- Tileset **setTileset** (string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)
- Tileset **setTileset** (int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)
- Tileset **setTileset** (int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)
- Tileset getTileset () const

  *Get the Tileset.*

- void createStage (Map m, Tileset t)

  *Create a stage by passing in a Map and Tileset.*

- void createStage (string filename, string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)

  *Create a stage by passing in the maps file, a name for the tiles, file of the tile image, the renderer, width and height of a tile, what row of the image the tiles are onem and how many tiles there are.*

- void **createStage** (string filename, string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)
- void **createStage** (string filename, int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)
- void **createStage** (string filename, int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)
- void setMap (Map m)

  *Set the Map by passing in a Map.*

- Map setMap (string filename)

  *Load in a new map by passing in the map file.*

- Map getMap () const

  *Get the Map.*

- void setTileset (Tileset t)

  *Set the Tileset with a given Tileset.*

- Tileset **setTileset** (string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)
- Tileset **setTileset** (string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)
- Tileset **setTileset** (int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)
- Tileset **setTileset** (int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)
- Tileset getTileset () const

  *Get the Tileset.*

### 3.12.1 Detailed Description

The Stage class stores a Map and Tileset.

Definition at line 619 of file arch.h.

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

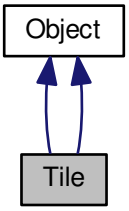
- arch.h
- stage.h
- stage.cpp

## 3.13 Tile Class Reference

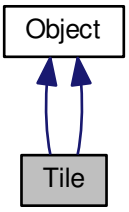
An Object class that stores the a tile value and name.

`#include <arch.h>`

Inheritance diagram for Tile:




Collaboration diagram for Tile:




### Public Member Functions

- void setValue (int v)

  *Set value of the tile. This is used when reading from a map file, etc.*
- int getValue () const

  *Get the value of the Tile.*
- void setSolid ()

  *Set if the Tile is solid.*
- void setPassable ()

  *Set if the Tile is passable (not solid).*
- bool isSolid () const

  *Check if the Tile is solid.*
- void setValue (int v)

*Set value of the tile. This is used when reading from a map file, etc.*

- int getValue () const

  *Get the value of the Tile.*

- void setSolid ()

  *Set if the Tile is solid.*

- void setPassable ()

  *Set if the Tile is passable (not solid).*

- bool isSolid () const

  *Check if the Tile is solid.*

### 3.13.1 Detailed Description

An Object class that stores the a tile value and name.

Definition at line 656 of file arch.h.

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

- arch.h
- tile.h
- tile.cpp

## 3.14 Tileset Class Reference

Class for loading in multiple Tiles.

`#include <arch.h>`

### Public Member Functions

- vector< Tile > **getTileset** () const
- SDL_Rect **getFrame** (int i)
- vector< Tile > create (string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)

  *Load in a map file with the name for all the tiles, the path to the map file, path to the tileset image, the SDL renderer, width and height of a tile, row to begin from on the image, how many tiles there are in the image.*

- vector< Tile > create (string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)

  *Load a map with a given name for the tiles, the file path to the map, the path to the tileset image, SDL renderer, width and height of a tile, row to begin on in the image, how many tiles on a certain row in the image, total amount of tiles in the image.*

- vector< Tile > create (int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)

  *Load in a map file with the name for all the tiles, the path to the map file, path to the tileset image, the SDL renderer, width and height of a tile, row to begin from on the image, how many tiles there are in the image.*

- vector< Tile > create (int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)

  *Load a map with a given name for the tiles, the file path to the map, the path to the tileset image, SDL renderer, width and height of a tile, row to begin on in the image, how many tiles on a certain row in the image, total amount of tiles in the image.*

- void addTile (Tile t)

*Push Tile in tile with given Tile.*

- Tile addTile (string name, string file, SDL_Renderer *ren, int value, int r, int c, int width, int height)

  *Generate and push Tile with tile name, path tot he tile image, SDL renderer, tile value, row and columg the tile as on in the image, the tiles width and height.*
- Tile addTile (string name, string file, SDL_Renderer *ren, int value, int width, int height)

  *Generate and push Tile with a given name, path to image file, SDL renderer, given value, and tile width and height.*
- Tile addTile (string name, string file, SDL_Renderer *ren, int value, int size)

  *Generate and push Tile with a given name, path to the image, SDL renderer, value, and size (used for width and height).*
- void setAngle (int ang)

  *Set the angle of all the tiles. Calls pushAng().*
- void **setSolid** ()
- void **setSolid** (int t)
- void **setSolid** (int s, int e)
- void **setPassable** ()
- void **setPassable** (int t)
- void **setPassable** (int s, int e)
- void **setName** (string n, int id)
- vector< Tile > **getTileset** () const
- SDL_Rect **getFrame** (int i)
- vector< Tile > create (string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)

  *Load in a map file with the name for all the tiles, the path to the map file, path to the tileset image, the SDL renderer, width and height of a tile, row to begin from on the image, how many tiles there are in the image.*
- vector< Tile > create (string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)

  *Load a map with a given name for the tiles, the file path to the map, the path to the tileset image, SDL renderer, width and height of a tile, row to begin on in the image, how many tiles on a certain row in the image, total amount of tiles in the image.*
- vector< Tile > create (int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int count)

  *Load in a map file with the name for all the tiles, the path to the map file, path to the tileset image, the SDL renderer, width and height of a tile, row to begin from on the image, how many tiles there are in the image.*
- vector< Tile > create (int startid, string name, string img, SDL_Renderer *ren, int width, int height, int r, int rcount, int count)

  *Load a map with a given name for the tiles, the file path to the map, the path to the tileset image, SDL renderer, width and height of a tile, row to begin on in the image, how many tiles on a certain row in the image, total amount of tiles in the image.*
- void addTile (Tile t)

  *Push Tile in tile with given Tile.*
- Tile addTile (string name, string file, SDL_Renderer *ren, int value, int r, int c, int width, int height)

  *Generate and push Tile with tile name, path tot he tile image, SDL renderer, tile value, row and columg the tile as on in the image, the tiles width and height.*
- Tile addTile (string name, string file, SDL_Renderer *ren, int value, int width, int height)

  *Generate and push Tile with a given name, path to image file, SDL renderer, given value, and tile width and height.*
- Tile addTile (string name, string file, SDL_Renderer *ren, int value, int size)

  *Generate and push Tile with a given name, path to the image, SDL renderer, value, and size (used for width and height).*
- void setAngle (int ang)

  *Set the angle of all the tiles. Calls pushAng().*
- void **setSolid** ()
- void **setSolid** (int t)
- void **setSolid** (int s, int e)
- void **setPassable** ()
- void **setPassable** (int t)
- void **setPassable** (int s, int e)
- void **setName** (string n, int id)

### 3.14.1 Detailed Description

Class for loading in multiple Tiles.

Definition at line 683 of file arch.h.

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

- arch.h
- tileset.h
- tileset.cpp

# Index