Mini2DEngine

Generated by Doxygen 1.9.2

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

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 GameObj::GameManager Struct Reference

```
#include <GameObjects.h>
```

Public Member Functions

- **GameManager** (ObjectManager ∗objMan)

4.1.1 Detailed Description

The object that will begin every game - add rooms here

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

- GameObjects.h
- GameObjects.cpp

4.2 IO::KeyboardInput Struct Reference

Public Member Functions

- KeyboardInput ()
- ∼KeyboardInput ()
- bool getKeyDown (int key_scancode)
- bool getKeyUp (int key_scancode)
- bool getKey (int key_scancode)
- void setKeyDown (int key_scancode, bool down)
- void setKeyUp (int key_scancode, bool up)
- void endUpdate ()

4.2.1 Constructor & Destructor Documentation

4.2.1.1 KeyboardInput()

```
KeyboardInput::KeyboardInput ( )
```

Construct new KeyboardInput object

4.2.1.2 ∼KeyboardInput()

```
KeyboardInput::∼KeyboardInput ( )
```

Destroy KeyboardInput object

4.2.2 Member Function Documentation

4.2.2.1 endUpdate()

```
void KeyboardInput::endUpdate ( )
```

Engine utility function, do not use

4.2.2.2 getKey()

```
bool KeyboardInput::getKey (
              int key_scancode )
```

Check state of key

Parameters

key_scancode	the code of the key to check. Refer to SDL Scancodes online

4.2.2.3 getKeyDown()

```
bool KeyboardInput::getKeyDown (
              int key_scancode )
```

Check if key pressed

Parameters

key_scancode	the code of the key to check. Refer to SDL Scancodes online

4.2.2.4 getKeyUp()

```
bool KeyboardInput::getKeyUp (
             int key_scancode )
```

Check if key released

Parameters

key_scancode	the code of the key to check. Refer to SDL Scancodes online

4.2.2.5 setKeyDown()

```
void KeyboardInput::setKeyDown (
             int key_scancode,
             bool down )
```

Engine utility function, do not use

4.2.2.6 setKeyUp()

```
void KeyboardInput::setKeyUp (
             int key_scancode,
             bool up )
```

Engine utility function, do not use

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

- IOHandlers.h
- InputHandlers.cpp

4.3 IO::MouseInput Struct Reference

```
#include <IOHandlers.h>
```

Public Member Functions

- MouseInput ()
- bool getLeftButtonDown ()
- bool getRightButtonDown ()
- bool getLeftButtonUp ()
- bool getRightButtonUp ()
- bool getLeftButton ()
- bool getRightButton ()
- int getX ()
- int getY ()
- void setLeftButtonDown (bool down)
- void setRightButtonDown (bool down)
- void setLeftButtonUp (bool Up)
- void setRightButtonUp (bool Up)
- void setLeftButton (bool held)
- void setRightButton (bool held)
- void setX (int newx)
- void setY (int newy)
- void endUpdate ()

4.3.1 Detailed Description

Handle mouse input

4.3.2 Constructor & Destructor Documentation

4.3.2.1 MouseInput()

```
MouseInput::MouseInput ( )
```

Construct new MouseInput object

4.3.3 Member Function Documentation

4.3.3.1 endUpdate()

```
void MouseInput::endUpdate ( )
```

Engine utility function, do not use

4.3.3.2 getLeftButton()

```
bool MouseInput::getLeftButton ( )
```

Check state of left mouse button

4.3.3.3 getLeftButtonDown()

```
bool MouseInput::getLeftButtonDown ( )
```

Check if left mouse button pressed

4.3.3.4 getLeftButtonUp()

```
bool MouseInput::getLeftButtonUp ( )
```

Check if left mouse button released

4.3.3.5 getRightButton()

```
bool MouseInput::getRightButton ( )
```

Check state of left mouse button

4.3.3.6 getRightButtonDown()

```
bool MouseInput::getRightButtonDown ( )
```

Check if right mouse button pressed

4.3.3.7 getRightButtonUp()

```
bool MouseInput::getRightButtonUp ( )
```

Check if right mouse button released

4.3.3.8 getX()

```
int MouseInput::getX ( )
```

Get x position of mouse in room

4.3.3.9 getY()

```
int MouseInput::getY ( )
```

Get y position of mouse in room

4.3.3.10 setLeftButton()

```
void MouseInput::setLeftButton (
            bool held )
```

Engine utility function, do not use

4.3.3.11 setLeftButtonDown()

```
void MouseInput::setLeftButtonDown (
            bool down )
```

Engine utility function, do not use

4.3.3.12 setLeftButtonUp()

```
void MouseInput::setLeftButtonUp (
            bool Up )
```

Engine utility function, do not use

4.3.3.13 setRightButton()

```
void MouseInput::setRightButton (
            bool held )
```

Engine utility function, do not use

4.3.3.14 setRightButtonDown()

```
void MouseInput::setRightButtonDown (
            bool down )
```

Engine utility function, do not use

4.3.3.15 setRightButtonUp()

```
void MouseInput::setRightButtonUp (
            bool Up )
```

Engine utility function, do not use

4.3.3.16 setX()

```
void MouseInput::setX (
            int newx )
```

Engine utility function, do not use

4.3.3.17 setY()

```
void MouseInput::setY (
            int newy )
```

Engine utility function, do not use

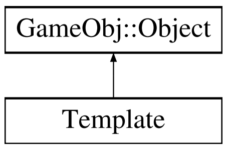
The documentation for this struct was generated from the following files:

- IOHandlers.h
- InputHandlers.cpp

4.4 GameObj::Object Struct Reference

`#include <GameObjects.h>`

Inheritance diagram for GameObj::Object:



Public Member Functions

- Object (ObjectManager ∗objMan, GameRoom::Room ∗room, double X, double Y)
- virtual ∼Object ()
- virtual void create ()
- virtual void update ()
- virtual void endUpdate ()
- virtual void draw ()
- virtual void destroy ()
- double getX ()
- void setX (double newx)
- double getY ()
- void setY (double newy)
- void setXY (double newx, double newy)
- void addX (double add)
- void addY (double add)
- double getDirection ()
- void setDirection (double newdirection)
- SDL_Rect ∗ getColMask ()
- bool setSprite (std::string ∗path)
- bool getHasSpr ()
- const char ∗ getSpritePath ()
- GameSpr::Sprite ∗ getSprite ()
- void setRoom (GameRoom::Room ∗newroom)
- void setRoomEnd ()
- GameRoom::Room ∗ getRoom ()
- void selfDestruct ()
- bool getPersistent ()
- bool getToDestruct ()
- template<typename T >
  T ∗ check_collision ()

Public Attributes

- size_t **index**

Protected Attributes

- ObjectManager ∗ **objMan**
- GameRoom::Room ∗ **room**
- bool **persistent**
- bool **room_end**

4.4.1 Detailed Description

A GameObject, the base unit on which all objects within a game are built on

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Object()

```
Object::Object (
            ObjectManager ∗ objMan,
            GameRoom::Room ∗ room,
            double X,
            double Y )
```

Constructs an Object, use within the class prototype of a derived class

Parameters

objMan	Pointer to the ObjectManager
room	pointer to the room that the object is in
X	the x value in the room that the object starts at
Y	the y value in the room that the object starts at

4.4.2.2 ∼Object()

```
Object::∼Object ( ) [virtual]
```

Destroy an object. Overrideable, required for overriding Object.destroy() with the derived class. See Game↩ Objects.cpp for an example (Asteroid)

4.4.3 Member Function Documentation

4.4.3.1 addX()

```
void Object::addX (
            double add )
```

Add a value to the x position of the object

Parameters

add	the amount to add

4.4.3.2 addY()

```
void Object::addY (
            double add )
```

Add a value to the y position of the object

Parameters

add	the amount to add

4.4.3.3 check_collision()

```
template<typename T >
T * Object::check_collision
```

Check if the object is colliding with an object of type T using both object's collision masks

Template Parameters

T	the type of the GameObject to check for collisions with

4.4.3.4 create()

```
void Object::create ( )  [virtual]
```

The Create event of an object. Overrideable

4.4.3.5 destroy()

```
void Object::destroy ( )  [virtual]
```

The Destroy event of an object. Overrideable. Use in conjunction with an overriden ~Object() for derived classes

4.4.3.6 draw()

```
void Object::draw ( )  [virtual]
```

The Draw event of an object. Overrideable

4.4.3.7 endUpdate()

```
void Object::endUpdate ( )  [virtual]
```

The End update event of an object. Overrideable

4.4.3.8 getColMask()

```
SDL_Rect * Object::getColMask ( )
```

Return the collision mask of the object, which is based on its sprite data

4.4.3.9 getDirection()

```
double Object::getDirection ( )
```

Get the direction of the object in degrees (it's rotation)

4.4.3.10 getHasSpr()

```
bool Object::getHasSpr ( )
```

Return whether the object has a sprite

4.4.3.11 getPersistent()

```
bool Object::getPersistent ( )
```

Return whether the object is persistsent

4.4.3.12 getRoom()

```
GameRoom::Room * Object::getRoom ( )
```

Get a pointer to the room the object is tied to

4.4.3.13 getSprite()

```
Sprite * Object::getSprite ( )
```

Get a pointer to the object's sprite

4.4.3.14 getSpritePath()

```
const char * Object::getSpritePath ( )
```

Get the path to the object's sprite

4.4.3.15 getToDestruct()

```
bool Object::getToDestruct ( )
```

Return whether the object is going to be destroyed in the next update

4.4.3.16 getX()

```
double Object::getX ( )
```

Get the x position of the object in the room

4.4.3.17 getY()

```
double Object::getY ( )
```

Get the y position of the object in the room

4.4.3.18 selfDestruct()

```
void Object::selfDestruct ( )
```

Trigger the object to destroy itself in the next update

4.4.3.19 setDirection()

```
void Object::setDirection (
             double newdirection )
```

Set the direction of the object

Parameters

newdirection	the angle of direction

4.4.3.20 setRoom()

```
void Object::setRoom (
             GameRoom::Room * newroom )
```

Change to room the object is 'in'. This means which object array the object is in. This is set to nullptr for a persistent object that has lost its original room

Parameters

newroom	a pointer to the room

4.4.3.21 setRoomEnd()

```
void Object::setRoomEnd ( )
```

Set the room_end flag to true - this flag is helpful in indicating that an object should act differently depending on whether it is being destroyed as part of game code, or simply the room/game ending. It is important to set this appropriately

4.4.3.22 setSprite()

```
bool Object::setSprite (
             std::string * path )
```

Set the sprite of the object

Parameters

path	the path of the new sprite

4.4.3.23 setX()

```
void Object::setX (
             double newx )
```

Set the x position of the object

Parameters

newx	the x value in the room

4.4.3.24 setXY()

```
void Object::setXY (
             double newx,
             double newy )
```

Set the x and y position of the object

Parameters

newx	the x value in the room
newx	the y value in the room

4.4.3.25 setY()

```
void Object::setY (
             double newy )
```

Set the y position of the object

Parameters

newx	the y value in the room

4.4.3.26 update()

```
void Object::update ( )  [virtual]
```

The Update event of an object. Overrideable

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

- GameObjects.h
- GameObjects.cpp

4.5 GameObj::ObjectManager Struct Reference

```
#include <GameObjects.h>
```

Public Member Functions

- **ObjectManager** (SDL_Renderer ∗renderer, IO::MouseInput ∗mouse, IO::KeyboardInput ∗keyboard)
- bool update ()
- void draw ()
- SDL_Renderer ∗ getRenderer ()
- size_t createObject (Object ∗obj)
- void destroyObject (size_t index, bool room_end)
- void nextRoom ()
- void gotoRoom (const char ∗name)
- void restartRoom ()
- size_t getNumObj ()
- template<typename T >
  T ∗ getObject (size_t index)
- template<typename T >
  T ∗ instance_find (size_t num)
- template<typename T >
  size_t instance_list (size_t ∗arraytofill)

Public Attributes

- IO::MouseInput ∗ **mouse**
- IO::KeyboardInput ∗ **keyboard**
- GameRoom::RoomManager ∗ **roomMan**
- GameText::TextManager ∗ **textMan**

4.5.1 Detailed Description

A management object that handles object lifespan, creation, and storage. It also connects the keyboard, mouse, renderer, text and room managers to allow GameObjects to access them

4.5.2 Member Function Documentation

4.5.2.1 createObject()

```
size_t ObjectManager::createObject (
            Object ∗ obj )
```

Connects a freshly created object to the ObjectManager and returns its index in the array of GameObjects

Parameters

obj	pointer to created game object

4.5.2.2 destroyObject()

```
void ObjectManager::destroyObject (
            size_t index,
            bool room_end )
```

Destroys an object, free the relevant memory and calling its destroy() event

Parameters

index	the index of the object to remove
room_end	flag to indicate if this is a room_end destruction - this effects memory management, only use true if you know what you're doing

4.5.2.3 draw()

```
void ObjectManager::draw ( )
```

Calls draw event for all GameObjects

4.5.2.4 getNumObj()

```
size_t ObjectManager::getNumObj ( )
```

Get the number of objects in the game

4.5.2.5 getObject()

```
template<typename T >
T * ObjectManager::getObject (
            size_t index )
```

Get a pointer to an object in the given type based on it's index in the array

Template Parameters

T	the type of the GameObject to get a pointer to

Parameters

index	the index of the object to get

4.5.2.6 getRenderer()

```
SDL_Renderer * ObjectManager::getRenderer ( )
```

Returns a pointer to the render target

4.5.2.7 gotoRoom()

```
void ObjectManager::gotoRoom (
             const char * name )
```

Go to a specified room

Parameters

name	the name of the room to go to

4.5.2.8 instance_find()

```
template<typename T >
T * ObjectManager::instance_find (
             size_t num )
```

Get a pointer to the nth object of the given type if it exists

Template Parameters

T	the type of the GameObject to get a pointer to

Parameters

num	the ordinal object to get (1st, 2nd etc)

4.5.2.9 instance_list()

```
template<typename T >
size_t ObjectManager::instance_list (
             size_t * arraytofill )
```

Fill a list with the indexes of objects of the given type, and return the size of the list

Template Parameters

T	the type of the GameObject to get a list of

Parameters

arrayToFill	a pointer to the array that will be filled

4.5.2.10 nextRoom()

```
void ObjectManager::nextRoom ( )
```

Goes to the next room. Use over RoomManager functions

4.5.2.11 restartRoom()

```
void ObjectManager::restartRoom ( )
```

Restart the current room

4.5.2.12 update()

```
bool ObjectManager::update ( )
```

Calls update event for all GameObjects

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

- GameObjects.h
- GameObjects.cpp

4.6 GameRoom::Room Struct Reference

```
#include <GameRooms.h>
```

Public Member Functions

- Room (std::string ∗room_path, std::function< void(Room ∗)> creationCode, rapidjson::Document ∗document, RoomManager ∗roomMan)
- ∼Room ()
- void roomStart ()
- bool addObject (size_t ∗obj_index)
- void removeObject (size_t ∗objToSearch)
- size_t ∗ getObjectIndexFromArray (size_t index)
- size_t getNumObjs ()
- std::string ∗ getName ()
- int getWidth ()
- int getHeight ()
- std::string ∗ getPath ()

Public Attributes

- std::function< void(Room ∗)> **creationCode**

4.6.1 Detailed Description

A Room object - essentially a level that contains objects, some create code, and a width and height

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Room()

```
Room::Room (
            std::string * room_path,
            std::function< void(Room *)> creationCode,
            rapidjson::Document * document,
            RoomManager * roomMan )
```

Construct new Room object - use RoomManager.addRoom() for safer room creation!

Parameters

room_path	the path to the rooms json property file
creationCode	a function that runs at the start of the room. Put objects to store in each room here
document	A structured interpretation of the rooms json property file
roomMan	pointer to it's manager. Dependency injection

4.6.2.2 ∼Room()

```
Room::~Room ( )
```

Destruct room. Use RoomManager.destroyRoom() for a safer function

4.6.3 Member Function Documentation

4.6.3.1 addObject()

```
bool Room::addObject (
            size_t * obj_index )
```

Add an object to the room - Unneccessary. Creating objects with ObjectManager already handles this function.

Parameters

obj_index	a pointer to the objects index in the obj_array of ObjectManager

4.6.3.2 getHeight()

```
int Room::getHeight ( )
```

Get height of the room

4.6.3.3 getName()

```
std::string * Room::getName ( )
```

Get name of the room

4.6.3.4 getNumObjs()

```
size_t Room::getNumObjs ( )
```

Get number of objects in the room

4.6.3.5 getObjectIndexFromArray()

```
size_t * Room::getObjectIndexFromArray (
              size_t index )
```

Return a pointer to an objects index in the obj_array of ObjectManager from the room

Parameters

index	index in the room's array of object index pointers

4.6.3.6 getPath()

```
std::string * Room::getPath ( )
```

Get path to the rooms json file

4.6.3.7 getWidth()

```
int Room::getWidth ( )
```

Get width of the room

4.6.3.8 removeObject()

```
void Room::removeObject (
             size_t * objToSearch )
```

Remove an object from the room - Unneccessary. Destroying objects with ObjectManager already handles this function.

Parameters

objToSearch	a pointer to the objects index in the obj_array of ObjectManager

4.6.3.9 roomStart()

```
void Room::roomStart ( )
```

Run creation code of room

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

- GameRooms.h
- GameRooms.cpp

4.7 GameRoom::RoomManager Struct Reference

```
#include <GameRooms.h>
```

Public Member Functions

- RoomManager ()
- ∼RoomManager ()
- Room ∗ getRoomPointer (size_t index)
- size_t getCurrentRoom ()
- std::string ∗ getRoomName (size_t index)
- bool destroyRoom ()
- size_t getNumRooms ()
- bool selectRoom (const char ∗name)
- bool restartRoom ()
- bool addRoom (std::string ∗roomJSON, std::function< void(Room ∗)> creationCode)

4.7.1 Detailed Description

A RoomManager object - stores the rooms, handles their lifetime, and manages movement between each room

4.7.2 Constructor & Destructor Documentation

4.7.2.1 RoomManager()

```
RoomManager::RoomManager ( )
```

Construct a RoomManager object

4.7.2.2 ∼RoomManager()

```
RoomManager::∼RoomManager ( )
```

Destruct a RoomManager object

4.7.3 Member Function Documentation

4.7.3.1 addRoom()

```
bool RoomManager::addRoom (
            std::string * roomJSON,
            std::function< void(Room *)> creationCode )
```

Create a new room safely

Parameters

room_path	the path to the rooms json property file
creationCode	a function that runs at the start of the room. Put objects to store in each room here

4.7.3.2 destroyRoom()

```
bool RoomManager::destroyRoom ( )
```

Destroy a room safely

4.7.3.3 getCurrentRoom()

```
size_t RoomManager::getCurrentRoom ( )
```

Get the index of the current room

4.7.3.4 getNumRooms()

```
size_t RoomManager::getNumRooms ( )
```

Get the number of rooms in the game

4.7.3.5 getRoomName()

```
std::string * RoomManager::getRoomName (
             size_t index )
```

Get the name of a room

4.7.3.6 getRoomPointer()

```
Room * RoomManager::getRoomPointer (
             size_t index )
```

Get a pointer to the room

Parameters

index	the index of the room in the room_array

4.7.3.7 restartRoom()

```
bool RoomManager::restartRoom ( )
```

Restart a room - use ObjectManager.restartRoom() for a safer function

4.7.3.8 selectRoom()

```
bool RoomManager::selectRoom (
             const char * name )
```

Go to a specific room - use ObjectManager.gotoRoom() for a safer function

Parameters

name	name of the room to go to

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

- GameRooms.h
- GameRooms.cpp

4.8 GameSpr::Sprite Struct Reference

```
#include <GameSprites.h>
```

Public Member Functions

- Sprite (const char ∗name, const char ∗path, const char ∗detailspath, SDL_Renderer ∗renderer)
- ∼Sprite ()
- void draw (double newx, double newy)
- void draw (double newx, double newy, double angle)
- const char ∗ getPath ()
- void setAngle (double newangle)
- double getAngle ()
- rapidjson::Document ∗ getSpriteDetails ()
- int getWidth ()
- int getHeight ()
- const char ∗ getName ()

4.8.1 Detailed Description

Represents a texture to draw to the screen, storing important details about its position in-world

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Sprite()

```
Sprite::Sprite (
            const char ∗ name,
            const char ∗ path,
            const char ∗ detailspath,
            SDL_Renderer ∗ renderer )
```

Construct new Sprite

Parameters

name	the name of the sprite
path	the path to the sprite's image file
path	the path to the sprite's properties file
renderer	the SDL_Renderer to use, as a pointer

4.8.2.2 ∼Sprite()

```
Sprite::∼Sprite ( )
```

Destroys the sprite

4.8.3 Member Function Documentation

4.8.3.1 draw() [1/2]

```
void Sprite::draw (
             double newx,
             double newy )
```

Draw the Sprite object at the given x and y

Parameters

newx	the x coordinate to draw at
newy	the y coordinate to draw at

4.8.3.2 draw() [2/2]

```
void Sprite::draw (
             double newx,
             double newy,
             double angle )
```

Draw the sprite object at the given x and y, at an angle

Parameters

newx	the x coordinate to draw at
newy	the y coordinate to draw at
angle	the angle to draw the sprite at

4.8.3.3 getAngle()

```
double Sprite::getAngle ( )
```

Get the angle of the sprite object

4.8.3.4 getHeight()

```
int Sprite::getHeight ( )
```

Get height of sprite

4.8.3.5 getName()

```
const char * Sprite::getName ( )
```

Get name of sprite

4.8.3.6 getPath()

```
const char * Sprite::getPath ( )
```

Get path to the sprite

4.8.3.7 getSpriteDetails()

```
Document * Sprite::getSpriteDetails ( )
```

Get a pointer to the details of the sprite in a Document object

4.8.3.8 getWidth()

```
int Sprite::getWidth ( )
```

Get width of sprite

4.8.3.9 setAngle()

```
void Sprite::setAngle (
             double newangle )
```

Change angle of sprite object

Parameters

angle	the angle to draw the sprite at

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

- GameSprites.h
- GameSprites.cpp

4.9 Sprite Struct Reference

Public Member Functions

- **Sprite** (const char ∗path, int x, int y)
- void **update** ()
- void **set_xy** (int newx, int newy)
- int **get_x** ()
- int **get_y** ()

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

- Source.cpp

4.10 SpriteManager Struct Reference

Public Member Functions

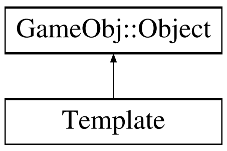
- void **update** ()
- bool **set_spr_xy** (size_t sprIndex, int x, int y)
- int **get_spr_x** (size_t sprIndex)
- int **get_spr_y** (size_t sprIndex)
- size_t **add_spr_from_path** (const char ∗path, int x, int y)

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

- Source.cpp

4.11 Template Struct Reference

Inheritance diagram for Template:



Public Member Functions

- **Template** (ObjectManager ∗objMan, Room ∗room, double X, double Y)

Additional Inherited Members

4.11.1 Detailed Description

A basic example for how objects should be structured

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

- GameObjects.cpp

4.12 GameText::Text Struct Reference

```
#include <GameText.h>
```

Public Member Functions

- Text (const char ∗fontpath, int size, const char ∗message, SDL_Color color, SDL_Renderer ∗renderer, size←
  _t index)
- Text (const char ∗fontpath, int size, const char ∗message, SDL_Color color, SDL_Renderer ∗renderer, int width, int height, size_t index)
- ∼Text ()
- void draw (double newx, double newy)
- void draw (double newx, double newy, double angle)
- void drawExt (double newx, double newy, double angle, const char ∗newmessage, bool autoSize)
- void changeMessage (const char ∗newmessage, bool autoSize)
- void setAngle (double newangle)
- double getAngle ()

Public Attributes

- size_t index

4.12.1 Detailed Description

A Text object - when initialised and drawn, will display its contained message with a defined colour and size

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Text() [1/2]

```
Text::Text (
             const char * fontpath,
             int size,
             const char * message,
             SDL_Color color,
             SDL_Renderer * renderer,
             size_t index )
```

Construct new Text object - use TextManager.createText() for safer text object creation

Parameters

fontpath	the path to the ttf font file
size	the size of the font in pixels
message	the c-string message to display
color	the color of the text
renderer	the SDL_Renderer to use, as a pointer
index	the index of the text object in the text array

4.12.2.2 Text() [2/2]

```
Text::Text (
            const char * fontpath,
            int size,
            const char * message,
            SDL_Color color,
            SDL_Renderer * renderer,
            int width,
            int height,
            size_t index )
```

Construct new Text object - use TextManager.createText() for safer text object creation

Parameters

fontpath	the path to the ttf font file
size	the size of the font in pixels
message	the c-string message to display
color	the color of the text
renderer	the SDL_Renderer to use, as a pointer
width	the width of the text area
height	the height of the text area
index	the index of the text object in the text array

4.12.2.3 ∼Text()

```
Text::∼Text ( )
```

Destroy the Text object - use TextManager.destroyText() for safer text object destruction

4.12.3 Member Function Documentation

4.12.3.1 changeMessage()

```
void Text::changeMessage (
            const char * newmessage,
            bool autoSize )
```

Change the text cstring to display

Parameters

newmessage	the new cstring to display
autoSize	whether to automatically resize text area to contain new message, or keep old size

4.12.3.2 draw() [1/2]

```
void Text::draw (
            double newx,
            double newy )
```

Draw the text object at the given x and y

Parameters

newx	the x coordinate to draw at
newy	the y coordinate to draw at

4.12.3.3 draw() [2/2]

```
void Text::draw (
            double newx,
            double newy,
            double angle )
```

Draw the text object at the given x and y, at an angle

Parameters

newx	the x coordinate to draw at
newy	the y coordinate to draw at
angle	the angle to draw the text at

4.12.3.4 drawExt()

```
void Text::drawExt (
            double newx,
            double newy,
            double angle,
            const char * newmessage,
            bool autoSize )
```

Draw the text object at the given x and y, at an angle, with a new message. Can automatically determine new size based on message, or simply keep the old size

Parameters

newx	the x coordinate to draw at
newy	the y coordinate to draw at
angle	the angle to draw the text at
newmessage	the new cstring to display
autoSize	whether to automatically resize text area to contain new message, or keep old size

4.12.3.5 getAngle()

```
double Text::getAngle ( )
```

Get the angle of the text object

4.12.3.6 setAngle()

```
void Text::setAngle (
            double newangle )
```

Change angle of text object

Parameters

angle	the angle to draw the text at

4.12.4 Member Data Documentation

4.12.4.1 index

```
size_t GameText::Text::index
```

A Text objects index position in the text_array of TextManager

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

- GameText.h
- GameText.cpp

4.13 GameText::TextManager Struct Reference

```
#include <GameText.h>
```

Public Member Functions

- TextManager (SDL_Renderer ∗renderer)
- ∼TextManager ()
- Text ∗ createText (const char ∗fontpath, int size, const char ∗message, SDL_Color color)
- Text ∗ createText (const char ∗fontpath, int size, const char ∗message, SDL_Color color, int width, int height)
- void destroyText (Text ∗text)

4.13.1 Detailed Description

Manage the lifetime and storage of Text objects

4.13.2 Constructor & Destructor Documentation

4.13.2.1 TextManager()

```
TextManager::TextManager (
            SDL_Renderer ∗ renderer )
```

Construct a new TextManager

Parameters

renderer	pointer to the SDL_Renderer target to use

4.13.2.2 ∼TextManager()

```
TextManager::∼TextManager ( )
```

Destructs the TextManager

4.13.3 Member Function Documentation

4.13.3.1 createText() [1/2]

```
Text * TextManager::createText (
            const char * fontpath,
            int size,
            const char * message,
            SDL_Color color )
```

Construct new Text object safely

Parameters

fontpath	the path to the ttf font file
size	the size of the font in pixels
message	the c-string message to display
color	the color of the text

4.13.3.2 createText() [2/2]

```
Text * TextManager::createText (
            const char * fontpath,
            int size,
            const char * message,
            SDL_Color color,
            int width,
            int height )
```

Construct new Text object safely

Parameters

fontpath	the path to the ttf font file
size	the size of the font in pixels
message	the c-string message to display
color	the color of the text
width	the width of the text area
height	the height of the text area

4.13.3.3 destroyText()

```
void TextManager::destroyText (
            Text * text )
```

Destroy the referenced Text object

Parameters

text	pointer to the Text object to destroy

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

- GameText.h
- GameText.cpp

Chapter 5

File Documentation

5.1 GameObjects.h File Reference

Handle objects.

```
#include "GameSprites.h"
#include "IOHandlers.h"
#include "GameRooms.h"
#include "GameText.h"
#include <map>
```

Classes

- struct GameObj::ObjectManager
- struct GameObj::Object
- struct GameObj::GameManager

Variables

- const size_t GameObj::max_obj { 100 }

5.1.1 Detailed Description

Handle objects.

5.1.2 Variable Documentation

5.1.2.1 max_obj

```
const size_t GameObj::max_obj { 100 }
```

The maximum number of objects that can exist concurrently

5.2 GameObjects.h

Go to the documentation of this file.

```
#pragma once

#include "GameSprites.h"
#include "IOHandlers.h"
#include "GameRooms.h"
#include "GameText.h"
#include <map>

namespace GameObj {
    const size_t max_obj{ 100 };

    struct Object;

    struct GameManager;

    struct ObjectManager {
        ObjectManager(SDL_Renderer* renderer, IO::MouseInput* mouse, IO::KeyboardInput* keyboard);
        ~ObjectManager();

        bool update();

        void draw();

        SDL_Renderer* getRenderer();

        size_t createObject(Object* obj);

        void destroyObject(size_t index, bool room_end);

        void nextRoom();

        void gotoRoom(const char* name);

        void restartRoom();

        size_t getNumObj();

        template<typename T>
        T* getObject(size_t index);

        template<typename T>
        T* instance_find(size_t num);

        template<typename T>
        size_t instance_list(size_t* arraytofill);

        IO::MouseInput* mouse;
        IO::KeyboardInput* keyboard;
        GameRoom::RoomManager* roomMan;
        GameText::TextManager* textMan;
    private:
        Object* obj_array[max_obj];
        size_t num_obj;
        bool game_end;
        SDL_Renderer* renderer;
        GameManager* gameMan;

    };

    struct Object {
         Object(ObjectManager* objMan, GameRoom::Room* room, double X, double Y);

         virtual ~Object();

         virtual void create();

         virtual void update();

         virtual void endUpdate();

        virtual void draw();

        virtual void destroy();

        double getX();

        void setX(double newx);

        double getY();

        void setY(double newy);

        void setXY(double newx, double newy);

        void addX(double add);

        void addY(double add);

        double getDirection();

        void setDirection(double newdirection);

        SDL_Rect* getColMask();

        bool setSprite(std::string* path);

        bool getHasSpr();

        const char* getSpritePath();

        GameSpr::Sprite* getSprite();

        void setRoom(GameRoom::Room* newroom);

        void setRoomEnd();

        GameRoom::Room* getRoom();

        void selfDestruct();

        bool getPersistent();

        bool getToDestruct();

        template<typename T>
        T* check_collision();

        size_t index;
    protected:
        ObjectManager* objMan;
        GameRoom::Room* room;
        bool persistent;
        bool room_end;
    private:
        bool toDestruct;
        bool hasSpr;
        GameSpr::Sprite* spr;
        SDL_Rect col_mask;
        double x;
        double y;
        double direction;

    };

    struct GameManager {
        GameManager(ObjectManager* objMan);

        ~GameManager();
    };
}
```

5.3 GameRooms.h File Reference

Handle rooms/levels.

```
#include <string>
#include <map>
#include <functional>
#include <rapidjson/document.h>
```

Classes

- struct GameRoom::Room
- struct GameRoom::RoomManager

Variables

- const size_t GameRoom::max_room_obj { 100 }
- const size_t GameRoom::max_rooms { 100 }

5.3.1 Detailed Description

Handle rooms/levels.

5.3.2 Variable Documentation

5.3.2.1 max_room_obj

```
const size_t GameRoom::max_room_obj { 100 }
```

Maximum number of objects per room

5.3.2.2 max_rooms

```
const size_t GameRoom::max_rooms { 100 }
```

Maximum number of rooms

5.4 GameRooms.h

Go to the documentation of this file.

```
#pragma once

#include <string>
#include <map>
#include <functional>
#include <rapidjson/document.h>

namespace GameRoom {
    const size_t max_room_obj{ 100 };

    const size_t max_rooms{ 100 };

    struct RoomManager;

    struct Room {
        Room(std::string* room_path, std::function<void(Room*)> creationCode, rapidjson::Document*
    document, RoomManager* roomMan);

        ~Room();

        void roomStart();
        bool addObject(size_t* obj_index);

        void removeObject(size_t* objToSearch);

        size_t* getObjectIndexFromArray(size_t index);

        size_t getNumObjs();

        std::string* getName();

        int getWidth();

        int getHeight();

        std::string* getPath();

        std::function<void(Room*)> creationCode;
    private:
        int w;
        int h;
        RoomManager* roomMan;

        size_t* obj_array[max_room_obj];
        size_t num_objs;
        std::string name;
        std::string* path;
        rapidjson::Document* room_data;
    };

    struct RoomManager {
        RoomManager();
        ~RoomManager();

        Room* getRoomPointer(size_t index);

        size_t getCurrentRoom();

        std::string* getRoomName(size_t index);

        bool destroyRoom();

        size_t getNumRooms();

        bool selectRoom(const char* name);

        bool restartRoom();

        bool addRoom(std::string* roomJSON, std::function<void(Room*)> creationCode);
    private:
        Room* room_array[max_rooms];
        std::map<std::string, size_t> room_names;
        size_t current_room;
        size_t num_rooms;
    };


}
```

5.5 GameSprites.h File Reference

Handle sprites.

```
#include <SDL.h>
#include <rapidjson/document.h>
```

Classes

- struct GameSpr::Sprite

Functions

- bool GameSpr::validateDetails (rapidjson::Document ∗details)
- bool GameSpr::rectIntersect (SDL_Rect ∗a, SDL_Rect ∗b)

5.5.1 Detailed Description

Handle sprites.

5.5.2 Function Documentation

5.5.2.1 rectIntersect()

```
bool GameSpr::rectIntersect (
            SDL_Rect * a,
            SDL_Rect * b )
```

Check if two rectangles intersect - for collisions

Parameters

a	the first rectangle
b	the second rectangle

5.5.2.2 validateDetails()

```
bool GameSpr::validateDetails (
            rapidjson::Document * details )
```

Confirm that details are in correct format and exist

Parameters

details	the details Document object to look at

5.6 GameSprites.h

Go to the documentation of this file.

```
#pragma once

#include <SDL.h>
#include <rapidjson/document.h>

namespace GameSpr {

    struct Sprite {
        Sprite(const char* name, const char* path, const char* detailspath, SDL_Renderer* renderer);

        ~Sprite();

        void draw(double newx, double newy);

        void draw(double newx, double newy, double angle);

        const char* getPath();

        void setAngle(double newangle);

        double getAngle();

        rapidjson::Document* getSpriteDetails();

        int getWidth();

        int getHeight();

        const char* getName();
    private:
        SDL_Texture* texture;
        SDL_Surface* temp;
        SDL_Renderer* renderer;
        SDL_Rect rect;
        double angle;
        const char* name;
        const char* path;
        const char* detailspath;
        rapidjson::Document* sprite_details;
    };

    bool validateDetails(rapidjson::Document* details);

    bool rectIntersect(SDL_Rect* a, SDL_Rect* b);
}
```

5.7 GameText.h File Reference

Handle text objects.

```
#include <SDL.h>
#include <SDL_ttf.h>
#include <string>
```

Classes

- struct GameText::Text
- struct GameText::TextManager

Variables

- const size_t GameText::max_texts { 100 }

5.7.1 Detailed Description

Handle text objects.

5.7.2 Variable Documentation

5.7.2.1 max_texts

```
const size_t GameText::max_texts { 100 }
```

Maximum number of text objects the engine allows

5.8 GameText.h

Go to the documentation of this file.

```
#pragma once

#include <SDL.h>
#include <SDL_ttf.h>
#include <string>

namespace GameText {
    const size_t max_texts{ 100 };

    struct Text {
        Text(const char* fontpath, int size, const char* message, SDL_Color color, SDL_Renderer* renderer, size_t index);

        Text(const char* fontpath, int size, const char* message, SDL_Color color, SDL_Renderer* renderer, int width, int height, size_t index);

        ~Text();

        void draw(double newx, double newy);

        void draw(double newx, double newy, double angle);

        void drawExt(double newx, double newy, double angle, const char* newmessage, bool autoSize);

        void changeMessage(const char* newmessage, bool autoSize);

        void setAngle(double newangle);

        double getAngle();

        size_t index;
    private:
        SDL_Texture* texture;
        SDL_Surface* temp;
        SDL_Renderer* renderer;
        TTF_Font* font;
        SDL_Rect rect;
        SDL_Color color;
        std::string message;
        int width;
        int height;
        int size;
        double angle;
    };

    struct TextManager {
        TextManager(SDL_Renderer* renderer);
        ~TextManager();

        Text* createText(const char* fontpath, int size, const char* message, SDL_Color color);

        Text* createText(const char* fontpath, int size, const char* message, SDL_Color color, int width, int height);

        void destroyText(Text* text);

    private:
        Text* text_array[max_texts];
        size_t num_texts;
        SDL_Renderer* renderer;
    };
}
```

5.9 IOHandlers.h File Reference

Handle the keyboard and mouse IO, connecting user-defined code to SDL events.

```
#include <SDL.h>
```

Classes

- struct IO::MouseInput
- struct IO::KeyboardInput

5.9.1 Detailed Description

Handle the keyboard and mouse IO, connecting user-defined code to SDL events.

5.10 IOHandlers.h

Go to the documentation of this file.

```
#pragma once

#include <SDL.h>

namespace IO {
    struct MouseInput {
        MouseInput();

        bool getLeftButtonDown();

        bool getRightButtonDown();

        bool getLeftButtonUp();

        bool getRightButtonUp();

        bool getLeftButton();

        bool getRightButton();

        int getX();

        int getY();

        /* Following functions are engine ONLY - used to set values */

        void setLeftButtonDown(bool down);

        void setRightButtonDown(bool down);

        void setLeftButtonUp(bool Up);

        void setRightButtonUp(bool Up);

        void setLeftButton(bool held);

        void setRightButton(bool held);

        void setX(int newx);

        void setY(int newy);

        void endUpdate();
    private:
        int x;
        int y;
        bool leftButton[3];
        bool rightButton[3];
    };

    struct KeyboardInput {
        KeyboardInput();

        ~KeyboardInput();

        bool getKeyDown(int key_scancode);

        bool getKeyUp(int key_scancode);

        bool getKey(int key_scancode);

        void setKeyDown(int key_scancode, bool down);

        void setKeyUp(int key_scancode, bool up);

        void endUpdate();
    private:
        const Uint8* keys;
        int numKeys;
        bool* keysDown;
        bool* keysUp;
    };
}
```

5.11 Main.cpp File Reference

Handles core engine functions and main game loop. Remember to add libpng.dll, jpeg.dll etc to the builds because of nuget not working!

```
#include <SDL.h>
#include <SDL_image.h>
#include <cstdio>
#include <stdexcept>
#include <climits>
#include <ctime>
#include "GameObjects.h"
```

Functions

- bool init ()
- bool update (ObjectManager ∗obj_manager, MouseInput ∗mouse, KeyboardInput ∗keyboard)
- bool draw (ObjectManager ∗obj_manager)
- void kill (ObjectManager ∗obj_manager)
- int main (int argc, char ∗∗args)

Variables

- SDL_Window ∗ **window** {}
- SDL_Renderer ∗ **renderer** {}

5.11.1 Detailed Description

Handles core engine functions and main game loop. Remember to add libpng.dll, jpeg.dll etc to the builds because of nuget not working!

5.11.2 Function Documentation

5.11.2.1 draw()

```
bool draw (
            ObjectManager * obj_manager )
```

Call draw event of all objects through the object manager, and update screen with the render target

5.11.2.2 init()

```
bool init ( )
```

Initialise game engine, including SDL, the window, and the renderer

5.11.2.3 kill()

```
void kill (
            ObjectManager * obj_manager )
```

Safely exit the SDL environment and free resources from memory

5.11.2.4 main()

```
int main (
            int argc,
            char ** args )
```

Main function - contains game loop

5.11.2.5 update()

```
bool update (
            ObjectManager * obj_manager,
            MouseInput * mouse,
            KeyboardInput * keyboard )
```

Handle update (per-frame) of game logic. Includes event pipeline

5.12 UserObjects.h

```
#pragma once
```

5.13 Utilities.h File Reference

Provide several basic utilities to the engine.

```
#include <utility>
#include <cmath>
#include <fstream>
#include <rapidjson/document.h>
```

Functions

- constexpr double Utils::degToRad (double deg)
- std::pair< double, double > Utils::dirLenToVector (double direction, double length)
- std::string * Utils::getStringFromFile (const char *filename)
- rapidjson::Document * Utils::parseJSON (const char *pathToFile)
- double Utils::randDouble (double max)
- double Utils::choose (double i, double j)

Variables

- const double Utils::pi = 3.14159265358979323846

5.13.1 Detailed Description

Provide several basic utilities to the engine.

5.13.2 Function Documentation

5.13.2.1 choose()

```
double Utils::choose (
            double i,
            double j )
```

Choose between two options at random and return one of them

Parameters

i	the first option that could be picked
j	the second option that could be picked

5.13.2.2 degToRad()

```
constexpr double Utils::degToRad (
            double deg )  [constexpr]
```

Convert degrees to radians

Parameters

deg	the value in degrees to convert

5.13.2.3 dirLenToVector()

```
std::pair< double, double > Utils::dirLenToVector (
            double direction,
            double length )
```

Convert a direction and length to a vector

Parameters

dir	direction in degrees
length	the length of the target vector

5.13.2.4 getStringFromFile()

```
std::string * Utils::getStringFromFile (
            const char * filename )
```

Return a string of a files contents

Parameters

filename	path of the file to read from

5.13.2.5 parseJSON()

```
rapidjson::Document * Utils::parseJSON (
            const char * pathToFile )
```

Parse a json file into a Document object

Parameters

pathToFile	path of the file to read from

5.13.2.6 randDouble()

```
double Utils::randDouble (
            double max )
```

Get a random double between 0 and a maximum

Parameters

max	the maximum value of the double

5.13.3 Variable Documentation

5.13.3.1 pi

```
const double Utils::pi = 3.14159265358979323846
```

The mathematical constant of pi, to 20 decimal places

5.14 Utilities.h

Go to the documentation of this file.

```
#ifndef H_UTILITIES
#define H_UTILITIES

#include <utility>
#include <cmath>
#include <fstream>
#include <rapidjson/document.h>


namespace Utils {
    const double pi = 3.14159265358979323846;

    constexpr double degToRad(double deg);

    std::pair<double, double> dirLenToVector(double direction, double length);

    std::string* getStringFromFile(const char* filename);

    rapidjson::Document* parseJSON(const char* pathToFile);

    double randDouble(double max);

    double choose(double i, double j);
}

#endif // !H_UTILITIES
```

Index