DSdl Framwork

Version 2.0 4/5/2016 8:37:00 PM

Table of Contents

Namespace Index	2
Hierarchical Index	3
Class Index	4
File Index	5
DsdlEngine	7
Class Documentation	
DsdlEngine::AudioManager	11
DsdlEngine::Button	
DsdlEngine::CollisionShape	
DsdlEngine::DsdlGui	
DsdlEngine::EngineBaseNode	
DsdlEngine::EngineMaster	
DsdlEngine::FileIO	
DsdlEngine::FpsLimiter	
DsdlEngine::IMainGame	
DsdlEngine::InputManager	
DsdlEngine::IScene	
DsdlEngine::Label	
DsdlEngine::Layer	
DsdlEngine::Music	
DsdlEngine::Particles	
DsdlEngine::ResourceTexture	
DsdlEngine::SceneManager	
DsdlEngine::SFX	
DsdlEngine::Size	70
DsdlEngine::Sprite	72
DsdlEngine::Vec2	75
DsdlEngine::Window	77
DsdlEngine::XmlLocalStorage	80
File Documentation	84
AudioManager.cpp	84
AudioManager.cpp	84
AudioManager.h	
AudioManager.h	
Button.cpp	
Button.cpp	
Button.h	91
Button.h	
CollisionShane.cpp	
CollisionShane.cpp	93
CollisionShape.h	94
CollisionShape.h	94
DsdlEngine.cpp	95
DsdlEngine.cpp	95
DsdlEngine.h	
DsdlEngine.h	
EngineBaseNode.cpp	
EngineBaseNode.cpp	
EngineBaseNode.h	
EngineBaseNode.h	
EngineDefines.h	103
EngineDefines h	105

EngineError.cpp	
EngineError.cpp	107
EngineError.h	108
EngineError.h	108
EngineMaster.cpp	109
EngineMaster.cpp	109
Engine Master.h	110
EngineMaster.h	
Engine Math.cpp	
EngineMath.cpp	
Engine Math.h	
Engine Math.h	
FileIO.cpp	
FileIO.cpp	
FileIO.h	
FileIO.h	
Gui.cpp	
Gui.cpp	
Gui.h	
Gui.h	
IMainGame.cpp	
IMainGame.cpp	
IMainGame.h	
IMainGame.h	
InputManager.cpp	
InputManager.cpp	
InputManager.h	
InputManager.h	
IScene.h	
IScene.h	
Label.cpp	
Label.cpp	
Label.h	
Label.h	
Layer.cpp	
Layer.cpp	
Layer.h	
Layer.h	
Particles.cpp	
Particles.cpp	139
Particles.h	
Particles.h	140
ResourceTexture.cpp	141
ResourceTexture.cpp	141
ResourceTexture.h	144
ResourceTexture.h	144
Scene.cpp	145
Scene.cpp	145
SceneManager.cpp	
SceneManager.cpp	
SceneManager.h	
SceneManager.h	
Sprite.cpp	
Sprite.cpp	
Sprite.h	
Sprite.h	
~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	102

	Timing.cpp	153
	Timing.cpp	153
	Timing.h	155
	Timing.h	155
	Window.cpp	156
	Window.cpp	156
	Window.h	
	Window.h	158
	XmlLocalStorage.cpp	
	XmlLocalStorage.cpp	159
	XmlLocalStorage.h.	163
	XmlLocalStorage.h	163
In	dex	164

Namespace Index

NIOM	ACD A	\sim	10+
Nam	E204	L.E.	_151
	JUPU	•	

Here is a list of all	namespaces with brief descriptions:	
DsdlEngine		7

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically: DsdlEngine::FileIO.......31 DsdlEngine::XmlLocalStorage 80

Class Index

Class List

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

<u>DsdlEngine::AudioManager</u>	11
<u>DsdlEngine::Button</u>	13
<u>DsdlEngine::CollisionShape</u>	16
DsdlEngine::DsdlGui	19
DsdlEngine::EngineBaseNode	22
DsdlEngine::EngineMaster	30
DsdlEngine::FileIO	31
<u>DsdlEngine::FpsLimiter</u>	35
<u>DsdlEngine::IMainGame</u>	38
<u>DsdlEngine::InputManager</u>	43
<u>DsdlEngine::IScene</u>	48
<u>DsdlEngine::Label</u>	52
<u>DsdlEngine::Layer</u>	54
<u>DsdlEngine::Music</u>	57
DsdlEngine::Particles	59
<u>DsdlEngine::ResourceTexture</u>	62
<u>DsdlEngine::SceneManager</u>	65
<u>DsdlEngine::SFX</u>	68
<u>DsdlEngine::Size</u>	70
<u>DsdlEngine::Sprite</u>	72
DsdlEngine::Vec2	75
DsdlEngine::Window	77
DsdlEngine::XmlLocalStorage	80

File Index

File List

Here is a list of all files with brief descriptions:

<u>AudioManager.cpp</u>	84
AudioManager.h	86
Button.cpp	88
Button.h	91
<u>CollisionShane.cpp</u>	93
CollisionShape.h	94
<u>DsdlEngine.cpp</u>	95
<u>DsdlEngine.h</u>	96
EngineBaseNode.cpp	98
EngineBaseNode.h	101
EngineDefines.h	103
EngineError.cpp	107
EngineError.h	108
EngineMaster.cpp	109
EngineMaster.h	110
EngineMath.cpp	111
EngineMath.h	112
FileIO.cpp	113
<u>FileIO.h</u>	117
<u>Gui.cpp</u>	119
Gui.h	121
IMainGame.cpp	122
IMainGame.h	126
InputManager.cpp	128
InputManager.h	131
<u>IScene.h</u>	132
<u>Label.cpp</u>	134
<u>Label.h</u>	135
<u>Layer.cpp</u>	136
<u>Layer.h</u>	138
Particles.cpp	139
<u>Particles.h</u>	140
ResourceTexture.cpp	141
ResourceTexture.h	144
Scene.cpp	145
SceneManager.cpp	147
SceneManager.h	149
Sprite.cpp	150
Sprite.h	152
Timing.cpp	153

Timing.h	155
Window.cpp	156
Window.h	158
XmlLocalStorage.cpp	159
XmlLocalStorage.h	163

Namespace Documentation

DsdlEngine Namespace Reference

Classes

- class AudioManager
- class Button
- class CollisionShape
- class DsdlGui
- class EngineBaseNode
- class EngineMaster
- class FileIO
- class FpsLimiter
- class IMainGame
- class <u>InputManager</u>
- class <u>IScene</u>
- class Label
- class <u>Layer</u>
- class <u>Music</u>
- class Particles
- class <u>ResourceTexture</u>
- class <u>SceneManager</u>
- class SFX
- class Size
- class Sprite
- class <u>Vec2</u>
- class Window
- class XmlLocalStorage

Typedefs

- typedef SDL_TimerID <u>CallBackTimer</u>
- typedef SDL_TimerCallback <u>CallBack</u>

Enumerations

- enum <u>NodeType</u> { <u>NodeType::BASENODE</u>, <u>NodeType::SPRITE</u>, <u>NodeType::LABEL</u>, <u>NodeType::BUTTON</u>, NodeType::PARTICLE }
- enum ButtonState { ButtonState::NORMAL, ButtonState::PRESSED, ButtonState::HOVERING }
- enum <u>ButtonType</u> { <u>ButtonType::LABEL_BTN</u>, <u>ButtonType::SPRITE_BTN</u> }
- enum LableType { LableType::LABEL STATIC, LableType::LABEL DYNAMIC }
- enum <u>SceneState</u> { <u>SceneState::NONE</u>, <u>SceneState::RUNNING</u>, <u>SceneState::EXIT_APP</u>,
 <u>SceneState::CHANGE_NEXT</u>, <u>SceneState::CHANGE_PREVIOUS</u> }

Functions

- int init ()
- template<typename T, typename... Args> std::unique_ptr< T > make_unique (Args &&...args)

Variables

• static <u>EngineMaster</u> * <u>Instance</u> = nullptr

- static <u>FileIO</u> * <u>Instance</u> = nullptr
- static <u>XmlLocalStorage</u> * <u>Instance</u> = nullptr

Detailed Description

Author:

Derek O Brien Derek O Brien

Class Based of

 $\frac{https://www.youtube.com/watch?v=Epyih-LEbig\&list=PLSPw4ASQYyymu3PfG9gxywSPghnSMiOAW}{\&index=26\ tutorial}$

Typedef Documentation

typedef SDL_TimerCallback DsdlEngine::CallBack

Definition at line <u>75</u> of file <u>EngineDefines.h.</u>

typedef SDL_TimerID <u>DsdlEngine::CallBackTimer</u>

Timer Call Back

Definition at line <u>74</u> of file <u>EngineDefines.h</u>.

Enumeration Type Documentation

enum DsdlEngine::ButtonState [strong]

Button State Enum

Enumerator

NORMAL PRESSED HOVERING

Definition at line 49 of file EngineDefines.h.

enum DsdlEngine::ButtonType [strong]

Button Type Enum

Enumerator

LABEL_BTN SPRITE BTN

Definition at line 58 of file EngineDefines.h.

enum DsdlEngine::LableType [strong]

Label Type Enum

Enumerator

```
LABEL_STATIC
LABEL_DYNAMIC
```

Definition at line 66 of file EngineDefines.h.

enum DsdlEngine::NodeType[strong]

```
Node Type Enum
```

Enumerator

BASENODE SPRITE LABEL BUTTON

PARTICLE

Definition at line <u>38</u> of file <u>EngineDefines.h</u>.

enum DsdlEngine::SceneState [strong]

SceneState enum class. For use when controlling the which scene is active..

Enumerator

NONE
RUNNING
EXIT_APP
CHANGE_NEXT
CHANGE_PREVIOUS

Definition at line 24 of file IScene.h.

Function Documentation

int DsdlEngine::init ()

init, Initalize SDL

Definition at line 7 of file <u>DsdlEngine.cpp</u>.

template<typename T , typename... Args> std::unique_ptr<T> DsdlEngine::make_unique (Args &&... args)

Definition at line 17 of file IMainGame.cpp.

Variable Documentation

EngineMaster* DsdlEngine::Instance = nullptr[static]

Definition at line 6 of file EngineMaster.cpp.

FileIO* DsdlEngine::Instance = nullptr[static]

Definition at line 11 of file FileIO.cpp.

XmlLocalStorage* DsdlEngine::Instance = nullptr[static]

Definition at line <u>16</u> of file <u>XmlLocalStorage.cpp</u>.

Class Documentation

DsdlEngine::AudioManager Class Reference

#include <AudioManager.h>

Public Member Functions

- AudioManager ()
- ~AudioManager ()
- void <u>init</u> ()
- void <u>destroy</u> ()
- <u>SFX loadSFX</u> (std::string audioPath)
- Music loadMusic (std::string audioPath)

Private Attributes

- std::map< std::string, Mix_Chunk * > m_sfxAudioMap
- std::map< std::string, Mix_Music * > m_bgAudioMap
- bool m_bisInitialized

Detailed Description

<u>AudioManager</u> Class. The <u>AudioManager</u> Class is responsable for loading and playing of audio within a game. The <u>AudioManager</u> will also cache any loaded audio.

Definition at line 81 of file AudioManager.h.

Constructor & Destructor Documentation

DsdlEngine::AudioManager::AudioManager()[inline]

<u>AudioManager</u> Constructor. On call will init the <u>AudioManager</u>.

Definition at line 87 of file AudioManager.h.

DsdlEngine::AudioManager::~AudioManager ()[inline]

<u>AudioManager</u> Deconstructor. On call will destroy current copy of the <u>AudioManager</u> and clear current cache

Definition at line 93 of file AudioManager.h.

Member Function Documentation

void DsdlEngine::AudioManager::destroy ()

Destroy current copy of <u>AudioManager</u> and clear cache maps.

Definition at line <u>24</u> of file <u>AudioManager.cpp</u>.

void DsdlEngine::AudioManager::init ()

init SDL Audio and set up Audio channels and frequency. Automaticaly called by Constructor.

Definition at line <u>10</u> of file <u>AudioManager.cpp</u>.

Music DsdlEngine::AudioManager::loadMusic (std::string audioPath)

Load Music.

Parameters:

std::string	for path to audio.
Stat. Strike	101 path to addio.

Returns:

Music to play.

Definition at line <u>87</u> of file <u>AudioManager.cpp</u>.

<u>SFX</u> DsdlEngine::AudioManager::loadSFX (std::string audioPath)

Load Sound effect.

Parameters:

stdvstring	for path to audio.
Biasiring	101 pain to addio.

Returns:

SFX chunk to play.

Definition at line 56 of file AudioManager.cpp.

Member Data Documentation

std::map<std::string, Mix_Music*> DsdlEngine::AudioManager::m_bgAudioMap [private]

Private cache map for storing musics.

Definition at line 129 of file AudioManager.h.

bool DsdlEngine::AudioManager::m_bisInitialized[private]

Private bool for control loading.

Definition at line <u>134</u> of file <u>AudioManager.h</u>.

std::map<std::string, Mix_Chunk*> DsdlEngine::AudioManager::m_sfxAudioMap [private]

Private cache map for storing sound effects.

Definition at line <u>124</u> of file <u>AudioManager.h</u>.

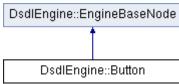
The documentation for this class was generated from the following files:

- AudioManager.h
- AudioManager.cpp

DsdlEngine::Button Class Reference

#include <Button.h>

Inheritance diagram for DsdlEngine::Button:



Public Member Functions

- Button ()
- virtual <u>~Button</u> ()
- void <u>destroy</u> ()
- void <u>createTextButton</u> (<u>Vec2</u> pos, <u>Size</u> size, std::string buttonText, std::string fontPath, SDL_Color textColor, SDL_Color bgColor)
- void <u>createSpriteButton</u> (<u>Vec2</u> spriteSize, <u>Vec2</u> position, std::string imagePath, std::string name)
- void <u>checkInput</u> (SDL_Event &e)
- std::string <u>getButtonName</u> ()

Public Attributes

• ButtonState m_eCurrentState

Private Member Functions

- void <u>onMouseEnters</u> ()
- void onMouseLeaves ()
- void onClicked ()

Private Attributes

- Label * m_label
- Sprite * m spriteBtn
- std::string <u>m_buttonName</u>

Additional Inherited Members

Detailed Description

<u>Button</u> Class subclass of <u>EngineBaseNode</u>. The button class is for creating buttons and handling events on such buttons.

Definition at line 19 of file Button.h.

Constructor & Destructor Documentation

DsdlEngine::Button::Button ()

Button Constructor.

Definition at line 6 of file Button.cpp.

DsdlEngine::Button::~Button ()[virtual]

Button Deconstructor.

Definition at line 11 of file Button.cpp.

Member Function Documentation

void DsdlEngine::Button::checkInput (SDL_Event & e)

checkInput. Check for input event on the current button.

Parameters:

e as SDL Event argument.

Definition at line <u>70</u> of file <u>Button.cpp</u>.

void DsdlEngine::Button::createSpriteButton (<u>Vec2</u> spriteSize, <u>Vec2</u> position, std::string imagePath, std::string name)

Create button as a **Sprite** node.

Parameters:

spriteSize	as a Vec2 argument.
position	as a Vec2 position argument
imagePath	as a std::string path to image
пате	as a std::string name of button

Definition at line <u>33</u> of file <u>Button.cpp</u>.

void DsdlEngine::Button::createTextButton (<u>Vec2</u> pos, <u>Size</u> size, std::string buttonText, std::string fontPath, SDL_Color textColor, SDL_Color bgColor)

Create button as Label node.

Parameters:

pos	as a <u>Vec2</u> position argument.
size	as a <u>Size</u> content size argument.
buttonText	as a std::string argument.
fontPath	as a std::string argument.
textColor	as a SDL_Color argument.
bgColor	as a SDL_Color argument.

Definition at line <u>14</u> of file <u>Button.cpp</u>.

void DsdlEngine::Button::destroy () [virtual]

destroy, responsible for cleaning up after button gose out of scope.

Reimplemented from DsdlEngine::EngineBaseNode.

Definition at line 127 of file Button.cpp.

std::string DsdlEngine::Button::getButtonName ()[inline]

getButtonName. Get the name of the button.

Returns:

std::string name of the button
Definition at line 70 of file Button.h.

void DsdlEngine::Button::onClicked ()[private]

onCLicked. Set button state to CLICKED.

Definition at line <u>64</u> of file <u>Button.cpp</u>.

void DsdlEngine::Button::onMouseEnters () [private]

onMouseEnters. Set button state to HOVERING.

Definition at line <u>53</u> of file <u>Button.cpp</u>.

void DsdlEngine::Button::onMouseLeaves ()[private]

onMouseLeaves. Set button state to NORMAL.

Definition at line <u>59</u> of file <u>Button.cpp</u>.

Member Data Documentation

std::string DsdlEngine::Button::m_buttonName[private]

std::string button name variable.

Definition at line 111 of file Button.h.

ButtonState DsdlEngine::Button::m_eCurrentState

ButtonState. Enum class for handeling the buttons state.

Definition at line <u>76</u> of file <u>Button.h</u>.

Label* DsdlEngine::Button::m_label[private]

<u>Label</u> varaible for creating label buttons.

Definition at line 101 of file Button.h.

Sprite* DsdlEngine::Button::m_spriteBtn[private]

Sprite variable for creating sprite button.

Definition at line 106 of file Button.h.

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

- Button.h
- <u>Button.cpp</u>

DsdlEngine::CollisionShape Class Reference

#include <CollisionShape.h>

Public Member Functions

- CollisionShape ()
- ~CollisionShape ()
- void <u>init</u> (b2World *world, <u>Vec2</u> position, <u>Vec2</u> dimensions, float density, float friction, bool fixedRotation)
- void <u>destroy</u> (b2World *world)
- b2Body * getBody () const
- b2Fixture * getFixture (int index) const
- const <u>Vec2 getDimensions</u> () const

Protected Attributes

- b2Body * m body = nullptr
- b2Fixture * <u>m_fixtures</u> [1]
- Vec2 m dimensions

Detailed Description

<u>CollisionShape</u> class is for creating a Box2D collision shape around the <u>Sprite</u> node.

Definition at line 12 of file CollisionShape.h.

Constructor & Destructor Documentation

DsdlEngine::CollisionShape::CollisionShape ()

Constructor.

Definition at line 6 of file CollisionShane.cpp.

DsdlEngine::CollisionShape::~CollisionShape ()

Deconstructor.

Definition at line 11 of file CollisionShane.cpp.

Member Function Documentation

void DsdlEngine::CollisionShape::destroy (b2World * world)

destroy shape in the Box2D world.

Parameters:

world as a b2World pointer argument.

Definition at line <u>45</u> of file <u>CollisionShane.cpp</u>.

b2Body* DsdlEngine::CollisionShape::getBody () const [inline]

getBody, get the body for the shape.

Returns:

b2Body pointer.

Definition at line <u>50</u> of file <u>CollisionShape.h.</u>

const Vec2 DsdlEngine::CollisionShape::getDimensions () const[inline]

getDimensions, get the dimensions of the shape.

Returns:

<u>Vec2</u> dimesions of the shape.

Definition at line 63 of file CollisionShape.h.

b2Fixture* DsdlEngine::CollisionShape::getFixture (int index) const[inline]

getFixture, get the fixture for index passed in.

Parameters:

index	as a int argument.

Returns:

b2Fixture pointer.

Definition at line <u>57</u> of file <u>CollisionShape.h.</u>

void DsdlEngine::CollisionShape::init (b2World * world, <u>Vec2</u> position, <u>Vec2</u> dimensions, float density, float friction, bool fixedRotation)

Initialize shape with arguments passed in.

Parameters:

position	as a <u>Vec2</u> position argument.
dimensions	as a <u>Vec2</u> size arguent.
density	as a float argument.
friction	as a float argument.
fixedRotation	as a bool argument.

Definition at line 16 of file CollisionShane.cpp.

Member Data Documentation

b2Body* DsdlEngine::CollisionShape::m body = nullptr[protected]

b2Body variabel.

Definition at line 69 of file CollisionShape.h.

Vec2 DsdlEngine::CollisionShape::m_dimensions[protected]

<u>Vec2</u> dimensions variable.

Definition at line 79 of file CollisionShape.h.

b2Fixture* DsdlEngine::CollisionShape::m_fixtures[1][protected]

Array of fixtures for the shape.

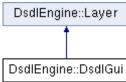
The documentation for this class was generated from the following files:

- CollisionShape.h
- CollisionShane.cpp

DsdlEngine::DsdlGui Class Reference

#include <Gui.h>

Inheritance diagram for DsdlEngine::DsdlGui:



Public Member Functions

- DsdlGui ()
- virtual <u>~DsdlGui</u> ()
- void <u>addButton</u> (<u>ButtonType</u> type, std::string name, <u>Vec2</u> pos, <u>Vec2</u> size, std::string path, SDL_Color color, SDL_Color bgColor, const char *text=NULL)
- void <u>addLabel</u> (<u>LableType</u> type, <u>Vec2</u> pos, std::string text, int fontSize, SDL_Color color, std::string fontFilePath)
- void <u>addPreDefineLabel</u> (<u>Label</u> *label, <u>LableType</u> type)
- void setGUIPos ()
- void <u>onSDLEvent</u> (SDL_Event &e)
- void destroy ()
- <u>Button</u> * <u>getButton</u> ()

Public Attributes

• std::vector< <u>Button</u> *> <u>GUIElements</u>

Protected Attributes

- Label * m_label
- Button * m_btn

Detailed Description

GUi <u>Layer</u> templtate for creating an a UI <u>Layer</u>. Inherits from layer

Definition at line 20 of file Gui.h.

Constructor & Destructor Documentation

DsdlEngine::DsdlGui::DsdlGui ()

Constructor

Definition at line <u>14</u> of file <u>Gui.cpp</u>.

DsdlEngine::DsdlGui::~DsdlGui ()[virtual]

Deconstructor

Definition at line 19 of file Gui.cpp.

Member Function Documentation

void DsdlEngine::DsdlGui::addButton (<u>ButtonType</u> type, std::string name, <u>Vec2</u> pos, <u>Vec2</u> size, std::string path, SDL_Color color, SDL_Color bgColor, const char * text = NULL) addButton, Creates and adds a button to the UI layer.

Parameters:

type,type	of button as a ButtonType
name,std::string	name of the button
pos, <u>Vec2</u>	position of the button
size, <u>Vec2</u>	size of the button
path,path	to texture to load.
color,SDL_color	of the button. for label type
bgColor,backgrou	color of the button. for label type.
nd	
text,text	to display. for label type.

Definition at line 24 of file Gui.cpp.

void DsdlEngine::DsdlGui::addLabel (<u>LableType</u> type, <u>Vec2</u> pos, std::string text, int fontSize, SDL_Color color, std::string fontFilePath)

addLabel, Creates and adds a Label to the UI layer.

Parameters:

type,type	of labe as a LabelType
pos, <u>Vec2</u>	position of the label
text,text	to display. for label type.
fontsize,as	int size of font.
color,SDL_color	of the label.
fontFilePath,file	path to the font.

Definition at line 46 of file Gui.cpp.

void DsdlEngine::DsdlGui::addPreDefineLabel (Label * label, LableType type)

addPredefinedLabel, add a pre made label to the UI layer,

Parameters:

label,the	<u>Label</u> to be added.
type,the	type of label.

Definition at line <u>57</u> of file <u>Gui.cpp</u>.

void DsdlEngine::DsdlGui::destroy () [virtual]

destroy, Clean up when left scope.

Reimplemented from DsdlEngine::Layer.

Definition at line <u>76</u> of file <u>Gui.cpp</u>.

Button* DsdlEngine::DsdlGui::getButton ()[inline]

getButton, Get button from the UIElemets vector

Returns:

Button.

Definition at line 85 of file Gui.h.

void DsdlEngine::DsdlGui::onSDLEvent (SDL_Event & e)

onSDLEvent, evnet listener for GUI buttons.

Parameters:

```
e,evnent to listen on.
```

Definition at line <u>68</u> of file <u>Gui.cpp</u>.

void DsdlEngine::DsdlGui::setGUIPos ()

setGUIPos, set the GUI position Definition at line 63 of file Gui.cpp.

Member Data Documentation

std::vector<Button*> DsdlEngine::DsdlGui::GUIElements

Vector to hold GUI Elements Definition at line <u>79</u> of file <u>Gui.h</u>.

<u>Button</u>* DsdlEngine::DsdlGui::m_btn[protected]

Definition at line 90 of file Gui.h.

Label* DsdlEngine::DsdlGui::m_label[protected]

Definition at line 89 of file Gui.h.

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

- <u>Gui.h</u>
- Gui.cpp

DsdlEngine::EngineBaseNode Class Reference

#include <EngineBaseNode.h>

Inheritance diagram for DsdlEngine::EngineBaseNode:

DsdlEngine::Button DsdlEngine::Label DsdlEngine::Particles DsdlEngine::Sprite

Public Member Functions

- EngineBaseNode ()
- virtual ~EngineBaseNode ()
- virtual void <u>destroy</u> ()
- virtual void <u>cleanup</u> ()
- bool load (SDL_Renderer *r)
- void <u>render</u> (SDL_Renderer *r)
- void renderectarray void renderectarray (SDL_Renderectarray) ** (SDL_Rendere
- void <u>setPosition</u> (const <u>Vec2</u> &pos)
- void setPositionX (int x)
- void <u>setPositionY</u> (int y)
- const <u>Vec2 getPosition</u> () const
- void <u>setSize</u> (<u>Size</u> si)
- void setWidth (int w)
- void <u>setHeight</u> (int h)
- const <u>Vec2 getContentSize</u> () const
- void scaleNode (float scale)
- void <u>scaleWidth</u> (float scale)
- void <u>scaleHeight</u> (float scale)
- void <u>setAssetPath</u> (std::string path)
- std::string <u>getAssetsPath</u> ()
- NodeType getNodeType ()
- void setEngineNodeType (NodeType type)
- void <u>setOpacity</u> (int opacity)
- ResourceTexture * getEngineTexture ()
- void <u>updateLabelText</u> (std::string text)
- SDL_Rect * <u>getBoundingBox</u> ()
- void setBoundingBox (Vec2 pos, Vec2 size)
- void setUpdateTextureTrue (bool value)
- bool <u>isTextureChanged</u> ()

Protected Attributes

- std::string <u>m_assetPath</u>
- NodeType nodeType = NodeType::BASENODE
- ResourceTexture * m_engineTexture
- SDL Rect * m objectBoundingBox
- <u>Vec2 m position</u>
- Vec2 m_size
- int m numFrames
- int m frame
- int m opacity

- bool updateTextureInfo
- SDL_Rect m gSpriteClips [14]
- SDL_Rect * m_currentFrame
- TTF_Font * m font
- std::map< std::string, TTF_Font *> m_FontMap
- std::string <u>m labelText</u>
- int m_textSize
- SDL_Color <u>m_textColor</u>
- CollisionShape * m_CollisionShape

Detailed Description

EngineBaseNode is the root for all elements in the framework

Definition at line 18 of file EngineBaseNode.h.

Constructor & Destructor Documentation

DsdlEngine::EngineBaseNode::EngineBaseNode ()

Constructor

Definition at line <u>10</u> of file <u>EngineBaseNode.cpp</u>.

DsdlEngine::EngineBaseNode::~EngineBaseNode ()[virtual]

Deconstructor

Definition at line <u>23</u> of file <u>EngineBaseNode.cpp</u>.

Member Function Documentation

void DsdlEngine::EngineBaseNode::cleanup ()[virtual]

virtual cleanup, for cleaning up node if it has to be reoved.

Reimplemented in DsdlEngine::Label.

Definition at line <u>165</u> of file <u>EngineBaseNode.cpp</u>.

void DsdlEngine::EngineBaseNode::destroy ()[virtual]

virtual destroy, destorys node when it leaves scope.

 $Reimplemented \ in \ \underline{DsdlEngine::Sprite}, \ \underline{DsdlEngine::Label}, \ and \ \underline{DsdlEngine::Button}.$

Definition at line <u>154</u> of file <u>EngineBaseNode.cpp</u>.

std::string DsdlEngine::EngineBaseNode::getAssetsPath()[inline]

getAssetsPath, get the path of the asset to be loaded.

Returns:

std::string path to asset.

Definition at line 144 of file EngineBaseNode.h.

SDL_Rect* DsdlEngine::EngineBaseNode::getBoundingBox ()[inline]

getBoundingBox, get the nodes SDL bounding box.

Returns:

SDL_Rect pointer.

Definition at line 180 of file EngineBaseNode.h.

const Vec2 DsdIEngine::EngineBaseNode::getContentSize () const [inline]

getContentSize, get size of the node.

Returns:

Vec2 size of the node

Definition at line <u>107</u> of file <u>EngineBaseNode.h</u>.

ResourceTexture* DsdlEngine::EngineBaseNode::getEngineTexture()[inline]

getEngineTexture, get the ResourceTexture for the node.

Returns:

ResourceTexture pointer.

Definition at line 168 of file EngineBaseNode.h.

NodeType DsdlEngine::EngineBaseNode::getNodeType ()[inline]

getNodeType, get the type of a specific node.

Returns:

NodeType, the type of node.

Definition at line 150 of file EngineBaseNode.h.

const Vec2 DsdlEngine::EngineBaseNode::getPosition () const [inline]

getPosition, get position of the node.

Returns:

const Vec2.

Definition at line 83 of file EngineBaseNode.h.

bool DsdlEngine::EngineBaseNode::isTextureChanged ()[inline]

isTextureChange, check if the texture was changed.

Returns:

bool.

Definition at line 200 of file EngineBaseNode.h.

bool DsdlEngine::EngineBaseNode::load (SDL_Renderer * r)

load, load node as SDL_Texture.

Parameters:

r	as SDL Renderer argument.

Returns:

bool.

Definition at line <u>61</u> of file <u>EngineBaseNode.cpp</u>.

void DsdlEngine::EngineBaseNode::render (SDL_Renderer * r)

render, Render node to window.

Parameters:

r	as SDL_Renderer argument

Definition at line <u>30</u> of file <u>EngineBaseNode.cpp</u>.

void DsdlEngine::EngineBaseNode::renderCollisionShape (SDL_Renderer * r, CollisionShape * shape)

rendereCollisionShape, Render collision shape for node.

Parameters:

r	as a SDL_Rendere argumnet.
shape	as a CollisionShape pointer argument.

Definition at line <u>170</u> of file <u>EngineBaseNode.cpp</u>.

void DsdlEngine::EngineBaseNode::scaleHeight (float scale)[inline]

scaleHeight, Scale the height of the node.

Parameters:

scale	as a float argument	
-------	---------------------	--

Definition at line <u>131</u> of file <u>EngineBaseNode.h.</u>

void DsdlEngine::EngineBaseNode::scaleNode (float scale)[inline]

scaleNode, Scale the node size by value passed in.

Parameters:

	scale	as a float argument
Definition at line 119 of file EngineBaseNode.h.		

void DsdlEngine::EngineBaseNode::scaleWidth (float scale)[inline]

scaleWidth, Scale the width of the node. scale as a float argument

Definition at line <u>125</u> of file <u>EngineBaseNode.h.</u>

void DsdlEngine::EngineBaseNode::setAssetPath (std::string path)[inline]

setAssetPath, set the path to the asset to be loaded.

Parameters:

path	as a std::string argument.	

Definition at line <u>138</u> of file <u>EngineBaseNode.h</u>.

void DsdlEngine::EngineBaseNode::setBoundingBox (Vec2 pos, Vec2 size)

setBoundingBox, the the bounding box for the node.

Parameters:

pos	as a Vec2 argument.
size	as a <u>Vec2</u> argument.

Definition at line <u>131</u> of file <u>EngineBaseNode.cpp</u>.

void DsdlEngine::EngineBaseNode::setEngineNodeType (NodeType type)[inline]

setEngineNodeType, set the type for a specific node.

Parameters:

ty	ре	as a NodeType argument.
	•	<u> </u>

Definition at line <u>156</u> of file <u>EngineBaseNode.h</u>.

void DsdlEngine::EngineBaseNode::setHeight (int h)[inline]

setHeight, Set height of the node.

Parameters:

	h	as a const int argument
_		

Definition at line 101 of file EngineBaseNode.h.

void DsdlEngine::EngineBaseNode::setOpacity (int opacity)

setOpacity, set the opacity value of a node, defaults to 255 if out of bounds value passed in.

Parameters:

opacity as an int argument between 0 - 255
--

Definition at line <u>144</u> of file <u>EngineBaseNode.cpp</u>.

void DsdlEngine::EngineBaseNode::setPosition (const Vec2 & pos)[inline]

setPosition, Set position of the node.

Parameters:

pos	as a const Vec2 argument

Definition at line <u>65</u> of file <u>EngineBaseNode.h</u>.

void DsdlEngine::EngineBaseNode::setPositionX (int x)[inline]

setPositionX, Set X position of the node.

Parameters:

x	as a const int argument

Definition at line <u>71</u> of file <u>EngineBaseNode.h</u>.

void DsdlEngine::EngineBaseNode::setPositionY (int y)[inline]

setPositionY, Set Y position of the node.

Parameters:

Y	as a const int argument

Definition at line <u>77</u> of file <u>EngineBaseNode.h.</u>

void DsdlEngine::EngineBaseNode::setSize (Size si)[inline]

setSize, Set size of the node.

Parameters:

si as a const Size argument

Definition at line 89 of file EngineBaseNode.h.

void DsdlEngine::EngineBaseNode::setUpdateTextureTrue (bool value)[inline]

setUpdateTextureTrue, control if node texture was changed after initial load.

Parameters:

value	as a bool argument.

Definition at line <u>194</u> of file <u>EngineBaseNode.h.</u>

void DsdlEngine::EngineBaseNode::setWidth (int w)[inline]

setWidth, Set width of the node.

Parameters:

		W	as a const int argument
--	--	---	-------------------------

Definition at line 95 of file EngineBaseNode.h.

void DsdlEngine::EngineBaseNode::updateLabelText (std::string text)

updateLabelText, change the display text of a label.

Parameters:

text	as a std::string argument.	

Definition at line 139 of file EngineBaseNode.cpp.

Member Data Documentation

std::string DsdlEngine::EngineBaseNode::m_assetPath[protected]

std::string asset path

Definition at line <u>205</u> of file <u>EngineBaseNode.h.</u>

<u>CollisionShape</u>* DsdlEngine::EngineBaseNode::m_CollisionShape[protected]

Box2D collision shape of the node

Definition at line <u>233</u> of file <u>EngineBaseNode.h.</u>

SDL_Rect* DsdlEngine::EngineBaseNode::m_currentFrame[protected]

the current frame rect

Definition at line 220 of file EngineBaseNode.h.

ResourceTexture* DsdlEngine::EngineBaseNode::m_engineTexture[protected]

ResourceTexture for the node

Definition at line 208 of file EngineBaseNode.h.

TTF_Font* DsdlEngine::EngineBaseNode::m_font[protected]

the font to use for labels

Definition at line 224 of file EngineBaseNode.h.

std::map<std::string, TTF_Font*> DsdlEngine::EngineBaseNode::m_FontMap [protected]

std::map for caching the font

Definition at line <u>225</u> of file <u>EngineBaseNode.h.</u>

int DsdlEngine::EngineBaseNode::m_frame[protected]

Definition at line <u>215</u> of file <u>EngineBaseNode.h.</u>

SDL_Rect DsdlEngine::EngineBaseNode::m_gSpriteClips[14][protected]

frames rect for animation

Definition at line <u>219</u> of file <u>EngineBaseNode.h.</u>

std::string DsdlEngine::EngineBaseNode::m_labelText[protected]

std::string to hold label display text

Definition at line <u>227</u> of file <u>EngineBaseNode.h</u>.

int DsdlEngine::EngineBaseNode::m_numFrames[protected]

Definition at line <u>215</u> of file <u>EngineBaseNode.h.</u>

SDL_Rect* DsdlEngine::EngineBaseNode::m_objectBoundingBox[protected]

SDL_Rect bounding box for the node

Definition at line <u>209</u> of file <u>EngineBaseNode.h</u>.

int DsdlEngine::EngineBaseNode::m_opacity[protected]

int values for frames & opacity

Definition at line <u>215</u> of file <u>EngineBaseNode.h</u>.

Vec2 DsdlEngine::EngineBaseNode::m_position [protected]

Vec2 Position of the node

Definition at line <u>212</u> of file <u>EngineBaseNode.h.</u>

Vec2 DsdlEngine::EngineBaseNode::m_size[protected]

Vec2 Size of the node

Definition at line <u>213</u> of file <u>EngineBaseNode.h</u>.

SDL_Color DsdlEngine::EngineBaseNode::m_textColor [protected]

color of the label

Definition at line 229 of file EngineBaseNode.h.

int DsdlEngine::EngineBaseNode::m_textSize[protected]

int for size of font

Definition at line <u>228</u> of file <u>EngineBaseNode.h.</u>

NodeType DsdlEngine::EngineBaseNode::nodeType = NodeType::BASENODE[protected]

NodeType for containing node type

Definition at line 206 of file EngineBaseNode.h.

bool DsdlEngine::EngineBaseNode::updateTextureInfo[protected]

bool for texture control

Definition at line <u>217</u> of file <u>EngineBaseNode.h</u>.

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

- EngineBaseNode.h
- EngineBaseNode.cpp

DsdlEngine::EngineMaster Class Reference

#include <EngineMaster.h>

Static Public Member Functions

• static <u>EngineMaster</u> * <u>getInstance</u> ()

Protected Member Functions

- <u>EngineMaster</u> ()
- virtual <u>~EngineMaster</u> ()

Detailed Description

EngineMaster is a stactic singleton helper class

Definition at line 14 of file EngineMaster.h.

Constructor & Destructor Documentation

DsdlEngine::EngineMaster::EngineMaster()[inline], [protected]

Constructor

Definition at line <u>27</u> of file <u>EngineMaster.h.</u>

virtual DsdlEngine::EngineMaster::~EngineMaster()[inline], [protected], [virtual]

Deconstructor

Definition at line 32 of file EngineMaster.h.

Member Function Documentation

EngineMaster * DsdlEngine::EngineMaster::getInstance ()[static]

getInstace, create <u>EngineMaster</u> as a Static instance. static instance of <u>EngineMaster</u> Definition at line <u>7</u> of file <u>EngineMaster.cpp</u>.

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

- EngineMaster.h
- EngineMaster.cpp

DsdlEngine::FileIO Class Reference

#include <FileIO.h>

Public Member Functions

- std::string getSuitableFOpen (const std::string &filenameUtf8) const
- std::string getWritablePath ()
- void <u>setAssetsPath</u> (std::string assetsPath)
- std::string <u>getFileToOpen</u> ()
- void setFileToOpen (std::string file)
- bool <u>loadDocument</u> (const char *filepath, char **doc_contents)
- bool <u>writeDocument</u> (const char *filepath, const char **doc_contents)
- XMLElement * getXMLNodeForKey (const char *pKey, XMLElement **rootNode, XMLDocument **doc)
- void setValueForKey (const char *value, const char *key)
- bool <u>createXMLFile</u> ()

Static Public Member Functions

• static <u>FileIO</u> * <u>getInstance</u> ()

Protected Member Functions

- <u>FileIO</u> ()
- virtual ~FileIO ()

Private Attributes

- std::string m path
- std::string <u>m_fileName</u>

Detailed Description

<u>FileIO</u> class handles open and cllosing of xml files in the framework. Handles XML parsing and Saveing Definition at line 21 of file FileIO.h.

Constructor & Destructor Documentation

DsdlEngine::FileIO::FileIO ()[inline], [protected]

Constructor

Definition at line <u>104</u> of file <u>FileIO.h</u>.

virtual DsdlEngine::FileIO::~FileIO()[inline], [protected], [virtual]

Deconstructor

Definition at line 109 of file FileIO.h.

Member Function Documentation

bool DsdlEngine::FileIO::createXMLFile ()

createXMLFile, Create a new Xml file.

Returns:

bool 1 on success.

Definition at line 207 of file FileIO.cpp.

std::string DsdlEngine::FileIO::getFileToOpen()[inline]

getFileToOpen, get the name of teh file to open.

Returns:

std::string filename.

Definition at line <u>54</u> of file <u>FileIO.h</u>.

FileIO * DsdlEngine::FileIO::getInstance ()[static]

getInstance, Creates FileIO as a static singleton

Returns:

instance of FileIO

Definition at line 13 of file FileIO.cpp.

std::string DsdlEngine::FileIO::getSuitableFOpen (const std::string & filenameUtf8) const

getSuitableFopen, the the filename of the path to open

Parameters:

std::string	file path.		
-------------	------------	--	--

Returns:

std::string file path to open.

Definition at line <u>22</u> of file <u>FileIO.cpp</u>.

std::string DsdlEngine::FileIO::getWritablePath ()

getWriteablePath, get the full path to the file.

Returns:

std::string.

Definition at line <u>37</u> of file <u>FileIO.cpp</u>.

XMLElement * DsdlEngine::FileIO::getXMLNodeForKey (const char * pKey, XMLElement ** rootNode, XMLDocument ** doc)

getXMLNodeForKey, parses the file contents in the memory buffer for a xml element that matches the key.

Parameters:

pKey	key to search for in the file.
rootNode	XML node to use for search.
doc	XML doc to hold the contents.

Returns:

XMLElement the element matching the key.

Definition at line 98 of file FileIO.cpp.

bool DsdlEngine::FileIO::loadDocument (const char * filepath, char ** doc_contents)

loadDocument, load the contents of a file into memory for parsing,

Parameters:

filepath	const char path to file,
doc_contents	buffer to hold the file contents.

Returns:

bool.

Definition at line <u>53</u> of file <u>FileIO.cpp</u>.

void DsdlEngine::FileIO::setAssetsPath (std::string assetsPath)[inline]

setAssetsPath, set the path to the file root (only applies to windows platform)

Parameters:

std::string path for file.

Definition at line <u>47</u> of file <u>FileIO.h</u>.

void DsdlEngine::FileIO::setFileToOpen (std::string file)[inline]

setFileToOpen, set the name of the file to open.

Parameters:

std::string	file name.

Definition at line <u>60</u> of file <u>FileIO.h</u>.

void DsdlEngine::FileIO::setValueForKey (const char * value, const char * key)

setValueForKey, Set or update the value of an XML element that matches the key.

Parameters:

vlaue	the value to be set.
key	the key to look for.

Definition at line <u>156</u> of file <u>FileIO.cpp</u>.

bool DsdlEngine::FileIO::writeDocument (const char * filepath, const char ** doc_contents)

writeDocument. Write the file contents from memory buffer to file and save.

Parameters:

filepath	const char path to file,
doc_contents	buffer cotaining the file contents.

Returns:

bool.

Definition at line <u>79</u> of file <u>FileIO.cpp</u>.

Member Data Documentation

std::string DsdlEngine::FileIO::m_fileName[private]

name of xml file to load

Definition at line <u>114</u> of file <u>FileIO.h</u>.

std::string DsdlEngine::FileIO::m_path[private]

path to folder which contains file Definition at line 109 of file FileIO.h.

- FileIO.h
- FileIO.cpp

DsdlEngine::FpsLimiter Class Reference

#include <Timing.h>

Public Member Functions

- <u>FpsLimiter</u> ()
- <u>~FpsLimiter</u> ()
- void <u>init</u> (float maxFPS)
- void <u>setMaxFPS</u> (float maxFPS)
- void <u>begin</u> ()
- float <u>end</u> ()

Private Member Functions

• void calculateFPS ()

Private Attributes

- float m fFps
- float m_fMaxFPS
- float <u>m fFrameTime</u>
- unsigned int <u>m_iStartTicks</u>

Detailed Description

Timming file handles setting up of caluclating and controlling frame rate of the engine.

Definition at line 12 of file Timing.h.

Constructor & Destructor Documentation

DsdlEngine::FpsLimiter::FpsLimiter ()

Constructor

Definition at line 10 of file Timing.cpp.

DsdlEngine::FpsLimiter::~FpsLimiter()

Deconstructor

Definition at line <u>13</u> of file <u>Timing.cpp</u>.

Member Function Documentation

void DsdlEngine::FpsLimiter::begin ()

Start the Frame Rate Timer

Definition at line 28 of file Timing.cpp.

void DsdlEngine::FpsLimiter::calculateFPS ()[private]

Calculate the running fps and keep it under control

Definition at line <u>45</u> of file <u>Timing.cpp</u>.

float DsdlEngine::FpsLimiter::end ()

End the frame rate timer

Returns:

float, the current fps vlaue.

Definition at line <u>33</u> of file <u>Timing.cpp</u>.

void DsdlEngine::FpsLimiter::init (float maxFPS)

Initializes the FPS limiter.

Parameters:

m	axFPS,the	max frame rate allowed.
---	-----------	-------------------------

Definition at line <u>18</u> of file <u>Timing.cpp</u>.

void DsdlEngine::FpsLimiter::setMaxFPS (float maxFPS)

Sets the desired max FPS

Parameters:

maxFPS,the	desired Frame Rate.

Definition at line 23 of file Timing.cpp.

Member Data Documentation

float DsdlEngine::FpsLimiter::m_fFps[private]

Definition at line <u>53</u> of file <u>Timing.h</u>.

float DsdlEngine::FpsLimiter::m_fFrameTime[private]

float values for claculations

Definition at line <u>53</u> of file <u>Timing.h</u>.

float DsdlEngine::FpsLimiter::m_fMaxFPS[private]

Definition at line 53 of file Timing.h.

unsigned int DsdlEngine::FpsLimiter::m_iStartTicks[private]

starting timestamp

Definition at line <u>54</u> of file <u>Timing.h</u>.

- Timing.h Timing.cpp

DsdlEngine::IMainGame Class Reference

#include <IMainGame.h>

Public Member Functions

- <u>IMainGame</u> ()
- virtual <u>~IMainGame</u> ()
- void <u>run</u> ()
- void <u>setupWindow</u> (int w, int h, std::string windowName, std::string path, int flag)
- void <u>setFps</u> (float fps)
- virtual void <u>onInit</u> ()=0
- virtual void addScenes ()=0
- virtual void <u>onExit</u> ()=0
- void <u>onSDLEvent</u> (SDL_Event &evnt)
- void <u>setPaused</u> ()
- void setRunning ()
- bool <u>checkPaused</u> ()

Public Attributes

• <u>InputManager m InputManager</u>

Protected Attributes

- std::unique_ptr< <u>SceneManager</u> > <u>m_pSceneManager</u>
- IScene * m pCurrentRunning
- bool m blsRunning
- bool m bIsPaused
- Window m Window
- SDL_Renderer * m_pGameRenderer
- AudioManager m audioManager

Private Member Functions

- const float getFps () const
- void mainLoop ()
- void <u>update</u> ()
- void <u>draw</u> ()
- bool init ()
- bool initSystems ()
- void <u>exitGame</u> ()

Private Attributes

- float m fFps
- unsigned int windowFlag
- int m_windowWidth
- int m windowHeight
- std::string <u>windowtitle</u>
- std::string <u>mainAssetsPath</u>

Detailed Description

<u>IMainGame</u> is the heart of the engine as it contians the mian game loop and ties all the engine together with the game. Users must inherit from this class to make their application entry point.

Definition at line 26 of file IMainGame.h.

Constructor & Destructor Documentation

DsdlEngine::IMainGame::IMainGame ()

Constructor

Definition at line 22 of file IMainGame.cpp.

DsdlEngine::IMainGame::~IMainGame()[virtual]

Deconstructor

Definition at line 26 of file IMainGame.cpp.

Member Function Documentation

virtual void DsdlEngine::IMainGame::addScenes () [pure virtual]

addScenes, pure virtual function for user custom logic. this is where the user can add their scenes to the game scene manager (m_pSceneManager) it is called at start of main loop.

bool DsdlEngine::IMainGame::checkPaused ()[inline]

checkPaused, check if the game is currently paused.

Returns:

bool

Definition at line 99 of file IMainGame.h.

void DsdlEngine::IMainGame::draw ()[private]

draw, the main draw function, called once every loop cycle. calls all nodes draw functions and display the node to window

Definition at line 229 of file IMainGame.cpp.

void DsdlEngine::IMainGame::exitGame ()[private]

exitGame. Cleans up and exits the game.

Definition at line <u>246</u> of file <u>IMainGame.cpp</u>.

const float DsdlEngine::IMainGame::getFps () const [inline], [private]

getFps, Get the running Frame Rate.

Returns:

float fps,

Definition at line <u>132</u> of file <u>IMainGame.h.</u>

bool DsdlEngine::IMainGame::init ()[private]

init, Initilazie the engine subsystems.

Returns:

bool.

Definition at line <u>129</u> of file <u>IMainGame.cpp</u>.

bool DsdlEngine::IMainGame::initSystems () [private]

initSystems. Create the SDL window and Renderer.

Returns:

bool.

Definition at line 169 of file IMainGame.cpp.

void DsdlEngine::IMainGame::mainLoop ()[private]

mainLoop The main game loop for the engine and game.

Definition at line <u>35</u> of file <u>IMainGame.cpp</u>.

virtual void DsdlEngine::IMainGame::onExit()[pure virtual]

onExit, pure virtual function for user custom logic. called when exiting the game, so user should implement any cleaup they want to do in here.

virtual void DsdlEngine::IMainGame::onlnit () [pure virtual]

onInit, pure virtual function for user custom logic should be used to setup window and fps as it is called before window is created.

void DsdlEngine::IMainGame::onSDLEvent (SDL_Event & evnt)

onSDLEvent, the games main Event listner

Parameters:

Ξ.		
	envt	as an SDL_Event

Definition at line 66 of file IMainGame.cpp.

void DsdlEngine::IMainGame::run ()

run, called in main file, runs the main game loop.

Definition at line <u>58</u> of file <u>IMainGame.cpp</u>.

void DsdlEngine::IMainGame::setFps (float fps)[inline]

setFps, Set the desired frame rate for the game.

Parameters:

fps	as a float value

Definition at line <u>58</u> of file <u>IMainGame.h</u>.

void DsdlEngine::IMainGame::setPaused ()[inline]

setPaused, Pauses the main game loop.

Definition at line 88 of file IMainGame.h.

void DsdlEngine::IMainGame::setRunning ()[inline]

setRunning, Starts the game loop running if paused.

Definition at line <u>93</u> of file <u>IMainGame.h</u>.

void DsdlEngine::IMainGame::setupWindow (int w, int h, std::string windowName, std::string path, int flag)

setupWindow, sets up the window defaults for Windows Platform.

Parameters:

w	as int width of the window.
h	as int height of the window.
windowName	as a std::string name of the window.
path	as std::string path to the windows root assets folder.
flag	as int SDL window creation flag

Definition at line <u>110</u> of file <u>IMainGame.cpp</u>.

void DsdlEngine::IMainGame::update()[private]

update, the main update function, called once every loop cycle updates any node that needs updating Definition at line 181 of file IMainGame.cpp.

Member Data Documentation

<u>AudioManager</u> DsdlEngine::IMainGame::m_audioManager[protected]

the main AudioManager

Definition at line 113 of file IMainGame.h.

bool DsdlEngine::IMainGame::m_blsPaused[protected]

bool variables for control

Definition at line 108 of file IMainGame.h.

bool DsdlEngine::IMainGame::m_blsRunning[protected]

Definition at line 108 of file IMainGame.h.

float DsdlEngine::IMainGame::m_fFps[private]

engines fps

Definition at line 118 of file IMainGame.h.

InputManager DsdlEngine::IMainGame::m_InputManager

Main games inputmanage object

Definition at line <u>101</u> of file <u>IMainGame.h</u>.

<u>IScene</u>* DsdIEngine::IMainGame::m_pCurrentRunning[protected]

current running scene

Definition at line <u>107</u> of file <u>IMainGame.h</u>.

SDL_Renderer* DsdlEngine::IMainGame::m_pGameRenderer[protected]

the engine renderer

Definition at line 111 of file IMainGame.h.

std::unique_ptr<<u>SceneManager</u>> DsdlEngine::IMainGame::m_pSceneManager[protected]

Main Scene Manager for the Engine

Definition at line 104 of file IMainGame.h.

Window DsdlEngine::IMainGame::m_Window[protected]

the main window variable

Definition at line <u>110</u> of file <u>IMainGame.h</u>.

int DsdlEngine::IMainGame::m_windowHeight[private]

window height variable

Definition at line 123 of file IMainGame.h.

int DsdlEngine::IMainGame::m_windowWidth[private]

window width variabel

Definition at line 122 of file IMainGame.h.

std::string DsdlEngine::IMainGame::mainAssetsPath[private]

asset path to windows assets folder

Definition at line 125 of file IMainGame.h.

unsigned int DsdlEngine::IMainGame::windowFlag[private]

windowFlag variable

Definition at line 121 of file IMainGame.h.

std::string DsdlEngine::IMainGame::windowtitle[private]

window title variable

Definition at line 124 of file IMainGame.h.

- <u>IMainGame.h</u>
- IMainGame.cpp

DsdlEngine::InputManager Class Reference

#include <InputManager.h>

Public Member Functions

- <u>InputManager</u> ()
- <u>~InputManager</u> ()
- void <u>update</u> ()
- void pressKey (unsigned int keyID)
- void releaseKey (unsigned int keyID)
- void <u>setMouseCoords</u> (float x, float y)
- bool <u>isKeyDown</u> (unsigned int keyID)
- bool <u>isKeyPressed</u> (unsigned int keyID)
- bool isKeyReleased (unsigned int KeyID)
- bool <u>isTouch</u> (unsigned int keyID)
- bool <u>isSwipe</u> (SDL_Event &evnt)
- bool <u>isSwipeUp</u> ()
- bool <u>isSwipeDown</u> ()
- bool <u>isSwipeLeft</u> (float x, float y)
- bool <u>isSwipeRight</u> (float x, float y)

Private Member Functions

• bool <u>wasKeyDown</u> (unsigned int keyID)

Private Attributes

- std::unordered_map< unsigned int, bool > <u>keyMap</u>
- std::unordered_map< unsigned int, bool > <u>_previousKeyMap</u>
- bool swipeup
- bool <u>swipedown</u>
- bool <u>swipeleft</u>
- bool swiperight
- bool <u>fingerDown</u>
- bool fingerUp

Detailed Description

InputManager Class handles all input in the game

Definition at line 15 of file InputManager.h.

Constructor & Destructor Documentation

DsdlEngine::InputManager::InputManager ()

Constructor

Definition at line 6 of file InputManager.cpp.

DsdlEngine::InputManager::~InputManager ()

Deconstructor

Definition at line 13 of file InputManager.cpp.

Member Function Documentation

bool DsdlEngine::InputManager::isKeyDown (unsigned int keyID)

isKeyDown, check if key is down.

Parameters:

keyID,ID	of key to be checked.

Returns:

bool.

Definition at line 39 of file InputManager.cpp.

bool DsdlEngine::InputManager::isKeyPressed (unsigned int keyID)

isKeyPressed, check if key was just pressed.

Parameters:

_		
	keyID,ID	of key to be checked.

Returns:

bool.

Definition at line 69 of file InputManager.cpp.

bool DsdlEngine::InputManager::isKeyReleased (unsigned int KeyID)

isKeyReleased, check if key was just realesed.

Parameters:

keyID,ID	of key to be checked.

Returns:

bool.

Definition at line 79 of file InputManager.cpp.

bool DsdlEngine::InputManager::isSwipe (SDL_Event & evnt)

isSwipe, check if it was a swipe event.

Parameters:

-	· diameterer	
	evnt,event	to be checked.

Returns:

bool.

Definition at line 100 of file InputManager.cpp.

bool DsdlEngine::InputManager::isSwipeDown ()

isSwipeDown, check if event was a swipe down.

Returns:

bool.

Definition at line <u>154</u> of file <u>InputManager.cpp</u>.

bool DsdlEngine::InputManager::isSwipeLeft (float x, float y)

isSwipeLeft, check if event was a swipe left.

Returns:

bool.

Definition at line <u>159</u> of file <u>InputManager.cpp</u>.

bool DsdlEngine::InputManager::isSwipeRight (float x, float y)

isSwipeRight, check if event was a swipe right.

Returns:

bool.

Definition at line <u>164</u> of file <u>InputManager.cpp</u>.

bool DsdlEngine::InputManager::isSwipeUp ()

isSwipeUP, check if event was a swipe up.

Returns:

bool.

Definition at line <u>149</u> of file <u>InputManager.cpp</u>.

bool DsdlEngine::InputManager::isTouch (unsigned int keyID)

isTouch, check if it was touch event.

Parameters:

K	eyID,ID	of key to be checked.

Returns:

bool.

Definition at line 90 of file InputManager.cpp.

void DsdlEngine::InputManager::pressKey (unsigned int keyID)

pressKey, add key pressed to the key map.

Parameters:

keyID,the	id of the key that was pressed.
	·

Definition at line <u>25</u> of file <u>InputManager.cpp</u>.

void DsdlEngine::InputManager::releaseKey (unsigned int keyID)

releaseKey, remove key pressed from the key map and add to previous map.

Parameters:

keyID,the	id of the key that was pressed.

Definition at line <u>30</u> of file <u>InputManager.cpp</u>.

void DsdlEngine::InputManager::setMouseCoords (float x, float y)

setMouseCoords, set the coordinates for the mouse.

Parameters:

x,float	value for mouse x location.
y,float	value for mouse y location.

Definition at line 34 of file InputManager.cpp.

void DsdlEngine::InputManager::update ()

update, loops through the key map.

Definition at line 18 of file InputManager.cpp.

bool DsdlEngine::InputManager::wasKeyDown (unsigned int keyID)[private]

wasKeyDown, Check if key was down.

Parameters:

keyID,ID	of key to be checked.

Returns:

bool.

Definition at line <u>54</u> of file <u>InputManager.cpp</u>.

Member Data Documentation

std::unordered map<unsigned int, bool> DsdlEngine::InputManager:: keyMap[private]

map to hold current keys

Definition at line 119 of file InputManager.h.

std::unordered_map<unsigned int, bool>

DsdlEngine::InputManager::_previousKeyMap[private]

map to hold previous keys

Definition at line 120 of file InputManager.h.

bool DsdlEngine::InputManager::fingerDown[private]

Definition at line 123 of file InputManager.h.

bool DsdlEngine::InputManager::fingerUp[private]

bools to control touch checking

Definition at line 123 of file InputManager.h.

bool DsdlEngine::InputManager::swipedown[private]

Definition at line 122 of file InputManager.h.

bool DsdlEngine::InputManager::swipeleft[private]

Definition at line 122 of file InputManager.h.

bool DsdlEngine::InputManager::swiperight[private]

bools to control swipe checking

Definition at line 122 of file InputManager.h.

bool DsdlEngine::InputManager::swipeup[private]

Definition at line <u>122</u> of file <u>InputManager.h</u>.

- InputManager.h
- InputManager.cpp

DsdlEngine::IScene Class Reference

#include <IScene.h>

Public Member Functions

- <u>IScene</u> ()
- virtual ~IScene ()
- virtual int <u>getNextSceneIndex</u> () const =0
- virtual int <u>getPreviousSceneIndex</u> () const =0
- virtual void <u>onEntryScene</u> ()=0
- virtual void onExitScene ()=0
- virtual void <u>updateScene</u> ()=0
- virtual void <u>destroyScene</u> ()=0
- int getSceneIndex () const
- <u>SceneState getSceneState</u> () const
- void setSceneRunning ()
- void <u>setParentGame</u> (<u>IMainGame</u> *game)
- virtual void <u>onInput</u> ()
- void <u>addLayerToScene</u> (<u>Layer</u> *layer)
- void <u>loadScene</u> (SDL_Renderer *r)
- void <u>drawScene</u> (SDL_Renderer *r)

Public Attributes

• std::vector< <u>Layer</u> * > <u>sceneLayers</u>

Protected Attributes

- SceneState m eCurrentState = SceneState::NONE
- <u>IMainGame</u> * <u>m game</u> = nullptr
- int m iSceneIndex = SCENE INDEX NO SCENE
- <u>InputManager m_inputManager</u>

Friends

- class <u>SceneManager</u> Friend Classes.
- class <u>InputManager</u>

Detailed Description

<u>IScene</u> is a inteface class to inherith from when creating a scene in the game.

Definition at line <u>35</u> of file <u>IScene.h.</u>

Constructor & Destructor Documentation

DsdlEngine::IScene::IScene ()[inline]

Constructor

Definition at line 40 of file IScene.h.

virtual DsdlEngine::IScene::~IScene ()[inline], [virtual]

Deconstructor

Definition at line 47 of file IScene.h.

Member Function Documentation

void DsdlEngine::IScene::addLayerToScene (Layer * layer)[inline]

Add a Layer to the current Scene.

Parameters:

layer, <u>Layer</u>	to add to the scene.
Definition at line 123 of	of file IScene.h.

virtual void DsdlEngine::IScene::destroyScene ()[pure virtual]

Pure virtual function. Destroy and cleanup when scene leaves scope.

void DsdlEngine::IScene::drawScene (SDL_Renderer * r)[inline]

Draw the current scenes layers to the window.

Parameters:

r,SDL_Renderer	to use when rendering.

Definition at line <u>141</u> of file <u>IScene.h</u>.

virtual int DsdlEngine::IScene::getNextSceneIndex () const [pure virtual]

Pure virtual function returns next scene.

Returns:

const int.

virtual int DsdlEngine::IScene::getPreviousSceneIndex () const [pure virtual]

Pure virtual function returns previous scene.

Returns:

const int.

int DsdlEngine::IScene::getSceneIndex () const[inline]

Gets the current scene's index.

Returns:

int.

Definition at line <u>93</u> of file <u>IScene.h</u>.

<u>SceneState</u> DsdlEngine::IScene::getSceneState () const [inline]

Get the current scenes state.

Returns:

SceneState int.

Definition at line 99 of file IScene.h.

void DsdlEngine::IScene::loadScene (SDL_Renderer * r)[inline]

Load the scene and its layers.

Parameters:

r,SDL_Renderer to use when loading

Definition at line 131 of file IScene.h.

virtual void DsdlEngine::IScene::onEntryScene ()[pure virtual]

Pure virtual function. Called when scene is loaded into focus.

virtual void DsdlEngine::IScene::onExitScene () [pure virtual]

Pure virtual function. Called when scene leaves focus.

void DsdlEngine::IScene::onInput ()[virtual]

Virtual function for scene specific input.

Definition at line 12 of file Scene.cpp.

void DsdlEngine::IScene::setParentGame (<u>IMainGame</u> * game)[inline]

Set the game that the scene belongs to. game. the <u>IMainGame</u> the scene belongs to.

Definition at line <u>110</u> of file <u>IScene.h</u>.

void DsdlEngine::IScene::setSceneRunning()[inline]

Set a scene running by setting the state.

Definition at line <u>104</u> of file <u>IScene.h</u>.

virtual void DsdlEngine::IScene::updateScene () [pure virtual]

Pure virtual function. Called when scene is in focus and updates all elemets in the scene.

Friends And Related Function Documentation

friend class InputManager[friend]

Definition at line <u>150</u> of file <u>IScene.h</u>.

friend class <u>SceneManager</u>[friend]

Friend Classes.

Definition at line <u>149</u> of file <u>IScene.h.</u>

Member Data Documentation

<u>SceneState</u> DsdlEngine::IScene::m_eCurrentState = <u>SceneState::NONE</u>[protected]

Scenes current state variabel

Definition at line 152 of file IScene.h.

IMainGame* DsdlEngine::IScene::m_game = nullptr[protected]

parent game.

Definition at line <u>154</u> of file <u>IScene.h</u>.

InputManager DsdlEngine::IScene::m_inputManager[protected]

scnenes input manager

Definition at line <u>156</u> of file <u>IScene.h</u>.

int DsdlEngine::IScene::m_iSceneIndex = SCENE_INDEX_NO_SCENE[protected]

scene index int

Definition at line 155 of file IScene.h.

std::vector<<u>Layer</u>*> DsdlEngine::lScene::sceneLayers

Vector to hold scenes game <u>Layer</u>

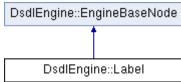
Definition at line 117 of file IScene.h.

- IScene.h
- Scene.cpp

DsdlEngine::Label Class Reference

#include <Label.h>

Inheritance diagram for DsdlEngine::Label:



Public Member Functions

- Label ()
- virtual <u>~Label</u> ()
- void <u>create</u> (<u>Vec2</u> pos, std::string text, int fontSize, SDL_Color color, std::string fontFilePath)
- void <u>setType</u> (<u>LableType</u> type)
- const int <u>getType</u> ()
- void <u>destroy</u> ()
- void <u>cleanup</u> ()

Protected Attributes

• <u>LableType</u> m_labelType

Detailed Description

<u>Label</u> class is the base for all labels in the engine it inherits for <u>EngineBaseNode</u>.

Definition at line 12 of file Label.h.

Constructor & Destructor Documentation

DsdlEngine::Label::Label()

Definition at line 11 of file Label.cpp.

DsdlEngine::Label::~Label()[virtual]

Deconstructor

Definition at line 16 of file Label.cpp.

Member Function Documentation

void DsdlEngine::Label::cleanup ()[virtual]

Cleanup the lable texture.

Reimplemented from DsdlEngine::EngineBaseNode.

Definition at line <u>39</u> of file <u>Label.cpp</u>.

void DsdlEngine::Label::create (<u>Vec2</u> pos, std::string text, int fontSize, SDL_Color color, std::string fontFilePath)

Create a basic <u>Label</u> pos, <u>Vec2</u> Position of the label. text, std::String label display text, fontSize, int the font size to use color, SDL_Color of the label. fontFilePath, std::string path to the font to use.

Definition at line <u>21</u> of file <u>Label.cpp</u>.

void DsdlEngine::Label::destroy()[virtual]

Destroy the label.

Reimplemented from <u>DsdlEngine::EngineBaseNode</u>.

Definition at line <u>34</u> of file <u>Label.cpp</u>.

const int DsdlEngine::Label::getType ()[inline]

Get the type of label it is.

Returns:

int label type,

Definition at line 44 of file Label.h.

void DsdlEngine::Label::setType (LableType type)[inline]

Set the type of label STATIC or DYNAMIC type, type of label.

Definition at line 38 of file Label.h.

Member Data Documentation

LableType DsdlEngine::Label::m_labelType[protected]

LabelType variable

Definition at line <u>57</u> of file <u>Label.h</u>.

- Label.h
- <u>Label.cpp</u>

DsdlEngine::Layer Class Reference

#include <Layer.h>
Inheritance diagram for DsdlEngine::Layer:

DsdlEngine::Layer

DsdlEngine::DsdlGui

Public Member Functions

- <u>Layer</u> ()
- virtual <u>~Layer</u> ()
- virtual void <u>destroy</u> ()
- void <u>loadNodes</u> (SDL_Renderer *r)
- void <u>drawNodes</u> (SDL_Renderer *r)
- void <u>addNodeToLayer</u> (<u>EngineBaseNode</u> *node)
- void removeNodeFromLayer (EngineBaseNode *node)

Public Attributes

• std::vector< <u>EngineBaseNode</u> * > <u>layerNodes</u>

Friends

• class <u>Gui</u>

Detailed Description

Definition at line 15 of file Layer.h.

Constructor & Destructor Documentation

DsdlEngine::Layer::Layer()

Constructor

Definition at line 13 of file Layer.cpp.

DsdlEngine::Layer::~Layer()[virtual]

Deconstructor

Definition at line 19 of file Layer.cpp.

Member Function Documentation

void DsdlEngine::Layer::addNodeToLayer (EngineBaseNode * node)

Add A node to the layer.

Parameters:

node, EngineBaseN	to add to the <u>Layer</u>
<u>ode</u>	

Definition at line <u>38</u> of file <u>Layer.cpp</u>.

void DsdlEngine::Layer::destroy ()[virtual]

Destroy the layer and all its contents

Reimplemented in <u>DsdlEngine::DsdlGui</u>.

Definition at line <u>24</u> of file <u>Layer.cpp</u>.

void DsdlEngine::Layer::drawNodes (SDL_Renderer * r)

Draw all nodes in the layer

Parameters:

r,SDL_Renderer	to be used when rendering	
----------------	---------------------------	--

Definition at line <u>57</u> of file <u>Layer.cpp</u>.

void DsdlEngine::Layer::loadNodes (SDL_Renderer * r)

Load all nodes in the layer

Parameters:

r,SDL_Renderer to be used when loading
--

Definition at line <u>50</u> of file <u>Layer.cpp</u>.

void DsdlEngine::Layer::removeNodeFromLayer (EngineBaseNode * node)

Remove a node from the layer.

Parameters:

node, Engine Base N	to be removed from the layer
<u>ode</u>	

Definition at line <u>44</u> of file <u>Layer.cpp</u>.

Friends And Related Function Documentation

friend class Gui[friend]

Definition at line <u>18</u> of file <u>Layer.h</u>.

Member Data Documentation

std::vector<<u>EngineBaseNode</u>*> DsdlEngine::Layer::layerNodes

vector to hold layer nodes

Definition at line <u>59</u> of file <u>Layer.h</u>.

- <u>Layer.h</u><u>Layer.cpp</u>

DsdlEngine::Music Class Reference

#include <AudioManager.h>

Public Member Functions

- void <u>play</u> (int loop=-1)
- void <u>audioPauseBG</u> ()
- void audioResumeBG ()
- void <u>audioStopBG</u> ()

Private Attributes

Mix_Music * m Music

Friends

class AudioManager

Detailed Description

Music class for interfacing with SDL Mix_Music.

Definition at line <u>39</u> of file <u>AudioManager.h</u>.

Member Function Documentation

void DsdlEngine::Music::audioPauseBG ()[inline]

Pause Music currently playeing.

Definition at line <u>55</u> of file <u>AudioManager.h</u>.

void DsdlEngine::Music::audioResumeBG ()[inline]

Resume Music that is currently paused.

Definition at line 60 of file AudioManager.h.

void DsdlEngine::Music::audioStopBG ()[inline]

Stop Music that is currently playing.

Definition at line 65 of file AudioManager.h.

void DsdlEngine::Music::play (int loop = -1)[inline]

Play Music.

Parameters:

loops	== -1 : loop forever, 0 : loop once, 1+ : loop that many times
-------	--

Definition at line <u>50</u> of file <u>AudioManager.h</u>.

Friends And Related Function Documentation

friend class AudioManager[friend]

Friend Class Audio Manager.

Definition at line 44 of file AudioManager.h.

Member Data Documentation

Mix_Music* DsdlEngine::Music::m_Music[private]

Private Mix_Music Variable

Definition at line <u>65</u> of file <u>AudioManager.h</u>.

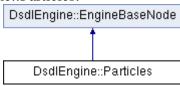
The documentation for this class was generated from the following file:

• AudioManager.h

DsdlEngine::Particles Class Reference

#include <Particles.h>

Inheritance diagram for DsdlEngine::Particles:



Public Member Functions

- Particles (int x, int y)
- <u>~Particles</u> ()
- bool <u>isDead</u> (<u>Particles</u> *p)

Static Public Member Functions

• static float <u>torad</u> (float <u>angle</u>)

Private Attributes

- int <u>life</u>
- float mPosX
- float mPosY
- float xvel
- float <u>yvel</u>
- float angle
- float <u>size</u>
- Uint32 <u>endtime</u>

Additional Inherited Members

Detailed Description

Definition at line 10 of file Particles.h.

Constructor & Destructor Documentation

DsdlEngine::Particles::Particles (int x, int y)

Definition at line 6 of file Particles.cpp.

DsdlEngine::Particles::~Particles ()

Definition at line 16 of file Particles.cpp.

Member Function Documentation

bool DsdlEngine::Particles::isDead (Particles * p)

Definition at line 19 of file Particles.cpp.

static float DsdlEngine::Particles::torad (float angle)[inline], [static]

Definition at line 18 of file Particles.h.

Member Data Documentation

float DsdlEngine::Particles::angle[private]

Definition at line 26 of file Particles.h.

Uint32 DsdlEngine::Particles::endtime[private]

Definition at line 27 of file Particles.h.

int DsdlEngine::Particles::life[private]

Definition at line 25 of file Particles.h.

float DsdlEngine::Particles::mPosX[private]

Definition at line 26 of file Particles.h.

float DsdlEngine::Particles::mPosY[private]

Definition at line 26 of file Particles.h.

float DsdlEngine::Particles::size[private]

Definition at line 26 of file Particles.h.

float DsdlEngine::Particles::xvel[private]

Definition at line 26 of file Particles.h.

float DsdlEngine::Particles::yvel[private]

- Particles.h
- Particles.cpp

DsdlEngine::ResourceTexture Class Reference

#include <ResourceTexture.h>

Public Member Functions

- ResourceTexture ()
- ~ResourceTexture ()
- bool loadTexture (std::string texturePath, SDL Renderer *r)
- bool <u>loadTTF</u> (std::string text, SDL_Color color, TTF_Font *myFont, SDL_Renderer *r)
- void <u>render</u> (<u>Vec2</u> p, <u>Vec2</u> s, SDL_Renderer *r, SDL_Rect *clip=NULL)
- void <u>setBlendMode</u> (SDL_BlendMode blending)
- void <u>setAlpha</u> (Uint8 alpha)
- void <u>destroy</u> ()

Private Attributes

- SDL_Texture * m_Texture
- std::map< std::string, SDL_Texture * > m TextureMap
- int m_iWidth
- int m iHeight

Detailed Description

<u>ResourceTexture</u> is responsible for loading and rendering all textures in the game. it is the base class for all textures.

Definition at line <u>15</u> of file <u>ResourceTexture.h</u>.

Constructor & Destructor Documentation

DsdlEngine::ResourceTexture::ResourceTexture ()

Constructor

Definition at line 24 of file ResourceTexture.cpp.

DsdlEngine::ResourceTexture::~ResourceTexture ()

Deconstructor

Definition at line 31 of file ResourceTexture.cpp.

Member Function Documentation

void DsdlEngine::ResourceTexture::destroy ()

Destroy the texture

Definition at line 129 of file ResourceTexture.cpp.

bool DsdlEngine::ResourceTexture::loadTexture (std::string texturePath, SDL_Renderer * r)

LoadTexture loads in sprite texture from the giving asset path.

Parameters:

texturePath,std::st	to the asset.
ring	
r,the	Renderer to use in loading

Returns:

bool

Definition at line <u>37</u> of file <u>ResourceTexture.cpp</u>.

bool DsdlEngine::ResourceTexture::loadTTF (std::string text, SDL_Color color, TTF_Font * myFont, SDL_Renderer * r)

LoadTTF loads in a texture created from a TTF font file.

Parameters:

text,text	to display onthe texture
color,the	SDL_COlor to use for the texture.
myFont,the	TTF_Font to use.
r,the	SDL_Renderer to use.

Returns:

bool.

Definition at line <u>80</u> of file <u>ResourceTexture.cpp</u>.

void DsdlEngine::ResourceTexture::render ($\underline{\text{Vec2}}$ p, $\underline{\text{Vec2}}$ s, SDL_Renderer * r, SDL_Rect * clip = NULL)

Render a texture to the window

Parameters:

<i>p</i> , <u><i>Vec</i>2</u>	postion to render too.
s, <u>Vec2</u>	size of texture to render.
r,SDL_Renderer	to use.
clip,the	Sprite texture clip frame to use.

Definition at line <u>107</u> of file <u>ResourceTexture.cpp</u>.

void DsdlEngine::ResourceTexture::setAlpha (Uint8 alpha)

Set the Alpha for a texture

Parameters:

alpha,UNit8	value of Alpha to use.

Definition at line <u>150</u> of file <u>ResourceTexture.cpp</u>.

void DsdlEngine::ResourceTexture::setBlendMode (SDL_BlendMode blending)

Set the blend mode of the texture.

Parameters:

	blending.	Blendmode to use.
_		

Definition at line <u>141</u> of file <u>ResourceTexture.cpp</u>.

Member Data Documentation

int DsdlEngine::ResourceTexture::m_iHeight[private]

widht and height of the SDL_Texture
Definition at line 75 of file ResourceTexture.h.

int DsdlEngine::ResourceTexture::m_iWidth[private]

Definition at line <u>75</u> of file <u>ResourceTexture.h.</u>

SDL_Texture* DsdlEngine::ResourceTexture::m_Texture[private]

The SDL_Texture to use when loading and rendering Definition at line 72 of file ResourceTexture.h.

std::map<std::string, SDL_Texture*> DsdlEngine::ResourceTexture::m_TextureMap[private]

std::Map to cache the textures

Definition at line <u>73</u> of file <u>ResourceTexture.h</u>.

- ResourceTexture.h
- ResourceTexture.cpp

DsdlEngine::SceneManager Class Reference

#include <SceneManager.h>

Public Member Functions

- <u>SceneManager</u> (<u>IMainGame</u> *game)
- ~SceneManager ()
- <u>IScene</u> * <u>moveNext</u> ()
- <u>IScene</u> * <u>movePrevious</u> ()
- void <u>setScene</u> (int nextScene)
- void addScene (IScene *newScene)
- void <u>destroy</u> ()
- <u>IScene</u> * <u>getCurrentScene</u> ()

Protected Attributes

- <u>IMainGame</u> * m_pGame
- std::vector< <u>IScene</u> *> <u>m pScenes</u>
- int m_iCurrentSceneIndex

Detailed Description

Scene Manager for handling all in game scenes, holds vector of all scenes, Definition at line 19 of file SceneManager.h.

Constructor & Destructor Documentation

DsdlEngine::SceneManager::SceneManager (IMainGame * game

Constructor.

Parameters:

game,the	IMainGame the manager belongs to

Definition at line <u>10</u> of file <u>SceneManager.cpp</u>.

DsdlEngine::SceneManager::~SceneManager()[inline]

Deconstructor.

Definition at line 31 of file SceneManager.h.

Member Function Documentation

void DsdlEngine::SceneManager::addScene (IScene * newScene)

Add a Scene to the Scene Manager.

Parameters:

newScene,the	IScene to add to the Manager.

Definition at line <u>39</u> of file <u>SceneManager.cpp</u>.

void DsdlEngine::SceneManager::destroy ()

Destroy the **SceneManager** and all of its Scenes

Definition at line <u>46</u> of file <u>SceneManager.cpp</u>.

<u>IScene</u> * DsdIEngine::SceneManager::getCurrentScene ()

Get the Current Scene been managed

Returns:

<u>IScene</u>, the current scene.

Definition at line <u>56</u> of file <u>SceneManager.cpp</u>.

IScene * DsdlEngine::SceneManager::moveNext ()

Move to Next scene in vector

Returns:

IScene, the scene to move to.

Definition at line 16 of file SceneManager.cpp.

<u>IScene</u> * DsdIEngine::SceneManager::movePrevious ()

Move to Previous scene in vector

Returns:

IScene, the scene to move to.

Definition at line <u>25</u> of file <u>SceneManager.cpp</u>.

void DsdlEngine::SceneManager::setScene (int nextScene)

Sets the current Scene

Parameters:

nextScene,the	current scene.
---------------	----------------

Definition at line <u>34</u> of file <u>SceneManager.cpp</u>.

Member Data Documentation

int DsdlEngine::SceneManager::m_iCurrentSceneIndex [protected]

index for the current Scene

Definition at line <u>74</u> of file <u>SceneManager.h</u>.

IMainGame* DsdlEngine::SceneManager::m_pGame[protected]

Main Game which scenemanager belongs too

Definition at line 70 of file SceneManager.h.

std::vector<|Scene*> DsdlEngine::SceneManager::m_pScenes [protected]

Vector to hold the game scenes

- SceneManager.h
- SceneManager.cpp

DsdlEngine::SFX Class Reference

#include <AudioManager.h>

Public Member Functions

• void <u>play</u> (int loop=0)

Private Attributes

Mix_Chunk * m_Chunk

Friends

class AudioManager

Detailed Description

Sound effect class for interfacing with SDL Mix_Chunk.

Definition at line 16 of file AudioManager.h.

Member Function Documentation

void DsdlEngine::SFX::play (int loop = 0)

Paly sound effect.

Parameters:

int loops == -1 : loop forever, 0 : loop once, 1+ : loop that many times

Definition at line 46 of file AudioManager.cpp.

Friends And Related Function Documentation

friend class AudioManager[friend]

Friend Class Audio Manager.

Definition at line 22 of file AudioManager.h.

Member Data Documentation

Mix_Chunk* DsdlEngine::SFX::m_Chunk[private]

Private Mix_Chunk Variable

Definition at line 33 of file AudioManager.h.

- <u>AudioManager.h</u><u>AudioManager.cpp</u>

DsdlEngine::Size Class Reference

#include <EngineMath.h>

Public Member Functions

- Size ()
- Size (float w, float h)
- <u>Size</u> (const <u>Size</u> &s)
- <u>~Size</u> ()

Public Attributes

- float w
- float <u>h</u>

Detailed Description

Size is a class for creating a 2 point size variable

Definition at line <u>55</u> of file <u>EngineMath.h.</u>

Constructor & Destructor Documentation

DsdlEngine::Size::Size ()

Constructor Defaults values to 0, 0

Definition at line <u>36</u> of file <u>EngineMath.cpp</u>.

DsdlEngine::Size::Size (float w, float h)

Constructor Set values on creation.

Parameters:

W	as a float argument
h	as a float argument

Definition at line <u>38</u> of file <u>EngineMath.cpp</u>.

DsdlEngine::Size::Size (const Size & s)

Constructor. Create a <u>Size</u> object with another <u>Size</u>

Parameters:

s as a Size argument

Definition at line <u>40</u> of file <u>EngineMath.cpp</u>.

DsdIEngine::Size::~Size()

Deconstructor

Definition at line 45 of file EngineMath.cpp.

Member Data Documentation

float DsdlEngine::Size::h_

float value for height

Definition at line <u>87</u> of file <u>EngineMath.h</u>.

float DsdlEngine::Size::w_

float value for width

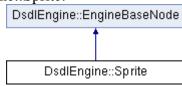
Definition at line <u>86</u> of file <u>EngineMath.h</u>.

- EngineMath.h
- EngineMath.cpp

DsdlEngine::Sprite Class Reference

#include <Sprite.h>

Inheritance diagram for DsdlEngine::Sprite:



Public Member Functions

- Sprite ()
- virtual <u>~Sprite</u> ()
- void <u>create</u> (<u>Vec2</u> spriteSize, <u>Vec2</u> position, std::string path)
- void <u>create</u> (<u>Vec2</u> spriteSize, <u>Vec2</u> position, std::string path, int numFrames)
- void <u>createWithPhysics</u> (b2World *world, <u>Vec2</u> spriteSize, <u>Vec2</u> position, std::string path, int numFrames, float den, float fri, bool FixedRotation)
- void <u>updateTexure</u> (<u>Vec2</u> spriteSize, <u>Vec2</u> position, std::string path, int numFrames)
- b2Body * getCollisionBody ()
- void <u>destroy</u> ()

Additional Inherited Members

Detailed Description

<u>Sprite</u> file for creating in game sprites. Inherits from <u>EngineBaseNode</u>

Definition at line 13 of file Sprite.h.

Constructor & Destructor Documentation

DsdlEngine::Sprite::Sprite ()

Constructor

Definition at line 11 of file Sprite.cpp.

DsdlEngine::Sprite::~Sprite()[virtual]

Deconstructor

Definition at line <u>21</u> of file <u>Sprite.cpp</u>.

Member Function Documentation

void DsdlEngine::Sprite::create (Vec2 spriteSize, Vec2 position, std::string path)

Create basic sprite with one frame.

Parameters:

spriteSize, <u>Vec2</u>	Size of the sprite.
position, <u>Vec2</u>	position of sprite.
path,std::string	path to sprite asset.

Definition at line <u>27</u> of file <u>Sprite.cpp</u>.

void DsdlEngine::Sprite::create (<u>Vec2</u> spriteSize, <u>Vec2</u> position, std::string path, int numFrames)

Create basic sprite with more then one frame.

Parameters:

spriteSize, <u>Vec2</u>	Size of the sprite.
position, <u>Vec2</u>	position of sprite.
path,std::string	path to sprite asset.
numFrames,int	number of frames.

Definition at line 43 of file Sprite.cpp.

void DsdlEngine::Sprite::createWithPhysics (b2World * world, <u>Vec2</u> spriteSize, <u>Vec2</u> position, std::string path, int numFrames, float den, float fri, bool FixedRotation)

Create basic sprite with one frame.

Parameters:

world,b2World	to add the sprite body to.
spriteSize, <u>Vec2</u>	Size of the sprite.
position, <u>Vec2</u>	position of sprite.
path,std::string	path to sprite asset.
numFrames,int	number of frames.
den,float	value for body dentisty.
fri,float	value for body friction.
fixedRotation,bool	value.

Definition at line <u>59</u> of file <u>Sprite.cpp</u>.

void DsdlEngine::Sprite::destroy () [virtual]

Destroy the sprite.

Reimplemented from DsdlEngine::EngineBaseNode.

Definition at line <u>79</u> of file <u>Sprite.cpp</u>.

b2Body* DsdlEngine::Sprite::getCollisionBody ()[inline]

Get the collision body of the sprite.

Returns:

b2Body, the sprites collision body Definition at line <u>68</u> of file <u>Sprite.h.</u>

void DsdlEngine::Sprite::updateTexure (<u>Vec2</u> spriteSize, <u>Vec2</u> position, std::string path, int numFrames)

Change the sprite texture of a sprite that is all ready loaded

Parameters:

spriteSize, <u>Vec2</u>	Size of the sprite.
-------------------------	---------------------

position, <u>Vec2</u>	position of sprite.
path,std::string	path to sprite asset.
numFrames,int	number of frames.

Definition at line <u>84</u> of file <u>Sprite.cpp</u>.

- Sprite.h Sprite.cpp

DsdlEngine::Vec2 Class Reference

#include <EngineMath.h>

Public Member Functions

- <u>Vec2</u> ()
- <u>Vec2</u> (float x, float y)
- <u>Vec2</u> (const <u>Vec2</u> &v)
- <u>~Vec2</u> ()

Public Attributes

- float x
- float <u>y</u>

Static Public Attributes

static const <u>Vec2</u> <u>ZERO</u>

Detailed Description

<u>Vec2</u> is a class for creating a 2 point position variable

Definition at line 11 of file EngineMath.h.

Constructor & Destructor Documentation

DsdIEngine::Vec2::Vec2 ()

Constructor Defaults values to 0, 0

Definition at line 11 of file EngineMath.cpp.

DsdlEngine::Vec2::Vec2 (float x, float y)

Constructor Set values on creation.

Parameters:

x	as a float argument
у	as a float argument

Definition at line 15 of file EngineMath.cpp.

DsdlEngine::Vec2::Vec2 (const Vec2 & v)

Constructor. Create a <u>Vec2</u> object with another <u>Vec2</u>

Parameters:

1,	as a Vac2 argument
V	as a <u>vecz</u> argument

Definition at line 18 of file EngineMath.cpp.

DsdlEngine::Vec2::~Vec2 ()

Deconstructor

Member Data Documentation

float DsdlEngine::Vec2::x_

float value for x position

Definition at line <u>46</u> of file <u>EngineMath.h</u>.

float DsdlEngine::Vec2::y_

float value for y position

Definition at line <u>47</u> of file <u>EngineMath.h</u>.

const Vec2 DsdlEngine::Vec2::ZERO[static]

const postition set to origin

Definition at line 44 of file EngineMath.h.

- EngineMath.h
- EngineMath.cpp

DsdlEngine::Window Class Reference

#include <Window.h>

Public Member Functions

- Window ()
- virtual ~Window ()
- int <u>createWindow</u> (std::string windowNname, int screenWidth, int screenHeight, unsigned int flag)
- void <u>swapBuffer</u> ()
- int getScreenWidth ()
- int getScreenHeight ()
- SDL_Renderer * <u>getRenderer</u> ()
- void <u>destroy</u> ()

Private Attributes

- SDL_Window * m_pSdlWindow
- SDL_Renderer * m_pSdlRenderer
- SDL_Surface * m_pScreenSurface
- int m screenHeight
- int m screenWidth

Detailed Description

<u>Window</u> class is the engines link to the SDL_Window. The window class is responsible for creating the window and renderer

Definition at line 16 of file Window.h.

Constructor & Destructor Documentation

DsdlEngine::Window::Window ()

Constructor

Definition at line 10 of file Window.cpp.

DsdlEngine::Window::~Window()[virtual]

Deconstructor

Definition at line <u>14</u> of file <u>Window.cpp</u>.

Member Function Documentation

int DsdlEngine::Window::createWindow (std::string windowNname, int screenWidth, int screenHeight, unsigned int flag)

Create the SDL_Window with the arguments passed in.

Parameters:

windowName,the	name of the window.
screenWidth,the	width of the window.
screenHeight,the	height of the window.
flag,SDL_Window	flag to use upon creation

Returns:

int, 0 upon success,

Definition at line 19 of file Window.cpp.

void DsdlEngine::Window::destroy ()

Destroy the window

Definition at line <u>74</u> of file <u>Window.cpp</u>.

SDL_Renderer* DsdlEngine::Window::getRenderer ()[inline]

Get the SDL_Renderer for the window.

Returns:

SDL Renderer.

Definition at line <u>60</u> of file <u>Window.h.</u>

int DsdlEngine::Window::getScreenHeight ()[inline]

Get the height of the window.

Returns:

int, height of the window,

Definition at line 54 of file Window.h.

int DsdlEngine::Window::getScreenWidth ()[inline]

Get the width of the window.

Returns:

int, width of the window.

Definition at line 48 of file Window.h.

void DsdlEngine::Window::swapBuffer ()

Swap the OpenGl window buffer.

Definition at line <u>69</u> of file <u>Window.cpp</u>.

Member Data Documentation

SDL_Surface* DsdlEngine::Window::m_pScreenSurface[private]

the SDL_Surface variable

Definition at line 70 of file Window.h.

SDL_Renderer* DsdlEngine::Window::m_pSdlRenderer[private]

the SDL_Renderer variable

Definition at line 69 of file Window.h.

SDL_Window* DsdlEngine::Window::m_pSdlWindow[private]

the SDL_Window variable

Definition at line <u>68</u> of file <u>Window.h</u>.

int DsdlEngine::Window::m_screenHeight[private]

Definition at line 72 of file Window.h.

int DsdlEngine::Window::m_screenWidth[private]

int variables for screenHeight and screenWidth

Definition at line <u>72</u> of file <u>Window.h</u>.

- Window.h
- Window.cpp

DsdlEngine::XmlLocalStorage Class Reference

#include <XmlLocalStorage.h>

Public Member Functions

- void <u>setIntegerForKey</u> (int value, const char *key)
- void setBoolForKey (bool value, const char *key)
- void setDoubleForKey (double value, const char *key)
- void <u>setFloatForKey</u> (float value, const char *key)
- void setStringForKey (std::string value, const char *key)
- int getIntegerForKey (const char *key)
- bool getBoolForKey (const char *key)
- double getDoubleForKey (const char *key)
- float getFloatForKey (const char *key)
- std::string getStringForKey (const char *key)
- void <u>deleteValueForKey</u> (const char *key)

Static Public Member Functions

• static <u>XmlLocalStorage</u> * <u>getInstance</u> ()

Protected Member Functions

- XmlLocalStorage ()
- virtual <u>~XmlLocalStorage</u> ()

Detailed Description

<u>XmlLocalStorage</u> is a class for reading and setting values in XML It is able to read and set all the base value types.

Definition at line 17 of file XmlLocalStorage.h.

Constructor & Destructor Documentation

DsdlEngine::XmlLocalStorage::XmlLocalStorage()[inline], [protected]

Constructor

Definition at line 108 of file XmlLocalStorage.h.

virtual DsdlEngine::XmlLocalStorage::~XmlLocalStorage()[inline],[protected],[virtual]

Deconstructor

Definition at line 113 of file XmlLocalStorage.h.

Member Function Documentation

void DsdlEngine::XmlLocalStorage::deleteValueForKey (const char * key)

Delete value for key. key, key to delete value for

Definition at line 211 of file XmlLocalStorage.cpp.

bool DsdlEngine::XmlLocalStorage::getBoolForKey (const char * key)

Get bool value by key, if the key doesn't exist, will return false.

Parameters:

key The key to get value.

Returns:

bool value by key.

Definition at line <u>55</u> of file <u>XmlLocalStorage.cpp</u>.

double DsdlEngine::XmlLocalStorage::getDoubleForKey (const char * key)

Get double value by key, if the key doesn't exist, will return 0.

Parameters:

7	TD1 1
kev	The key to get value.
I KC y	The key to get value.

Returns:

double value by key.

Definition at line <u>84</u> of file <u>XmlLocalStorage.cpp</u>.

float DsdlEngine::XmlLocalStorage::getFloatForKey (const char * key)

Get float value by key, if the key doesn't exist, will return 0.

Parameters:

key	The key to get value.	
-----	-----------------------	--

Returns:

float value by key.

Definition at line 112 of file XmlLocalStorage.cpp.

XmlLocalStorage * DsdlEngine::XmlLocalStorage::getInstance ()[static]

getInstance gets a static instance for the XmlLocalStorage class.

Returns:

XmlLocalStorage instance

Definition at line 17 of file XmlLocalStorage.cpp.

int DsdlEngine::XmlLocalStorage::getIntegerForKey (const char * key)

Get int value by key, if the key doesn't exist, will return 0.

Parameters:

key The key to get value.

Returns:

int value by key.

Definition at line <u>27</u> of file <u>XmlLocalStorage.cpp</u>.

std::string DsdlEngine::XmlLocalStorage::getStringForKey (const char * key)

Get string value by key, if the key doesn't exist, will return null.

Parameters:

key	The key to get value.

Returns:

string value by key.

Definition at line 118 of file XmlLocalStorage.cpp.

void DsdlEngine::XmlLocalStorage::setBoolForKey (bool value, const char * key)

Set a bool value for a key in xml.

Parameters:

key	The key to set.
value,bool	value to be saved

Definition at line 166 of file XmlLocalStorage.cpp.

void DsdlEngine::XmlLocalStorage::setDoubleForKey (double value, const char * key)

Set a double value for a key in xml.

Parameters:

key	The key to set.
value,double	value to be saved

Definition at line <u>176</u> of file <u>XmlLocalStorage.cpp</u>.

void DsdlEngine::XmlLocalStorage::setFloatForKey (float value, const char * key)

Set a float value for a key in xml.

Parameters:

key	The key to set.
value,float	value to be saved

Definition at line 198 of file XmlLocalStorage.cpp.

void DsdlEngine::XmlLocalStorage::setIntegerForKey (int value, const char * key)

Set a int value for a key in xml.

Parameters:

key	The key to set.
value,int	value to be saved

Definition at line <u>144</u> of file <u>XmlLocalStorage.cpp</u>.

void DsdlEngine::XmlLocalStorage::setStringForKey (std::string value, const char * key)

Set a string value for a key in xml.

Parameters:

key	The key to set.
value,string	value to be saved

Definition at line <u>203</u> of file <u>XmlLocalStorage.cpp</u>.

- XmlLocalStorage.h XmlLocalStorage.cpp

File Documentation

AudioManager.cpp File Reference

```
#include "AudioManager.h"
#include "FileIO.h"
```

Namespaces

• <u>DsdlEngine</u>

AudioManager.cpp

```
1 #include "AudioManager.h"
 2 #include "FileIO.h"
 7 namespace <a href="DsdlEngine">DsdlEngine</a> {
9
       //init audio manager
10
       void AudioManager::init() {
11
           //init audio
           if (Mix Init(MIX INIT MP3 | MIX INIT OGG) == -1) {
12
               SDL Log("Mix Init error: %s", Mix GetError());
13
14
15
           //open audio defaults
           if (Mix_OpenAudio(MIX_DEFAULT_FREQUENCY, MIX_DEFAULT_FORMAT, 2, 2048) == -1) {
16
17
               SDL Log("Mix OpenAudio error: %s " ,Mix GetError());
18
19
20
           m bisInitialized = true;
2.1
2.3
       //Clean Up Aduio
24
       void AudioManager::destroy() {
25
         if (m_bisInitialized)
26
               m bisInitialized = false;
27
28
           //Loop Through maps and free audio
29
           for (auto& it : m sfxAudioMap) {
30
               Mix FreeChunk(it.second);
31
32
33
           for (auto& it : m bgAudioMap) {
34
               Mix FreeMusic(it.second);
35
36
           //Clear maps
           m bgAudioMap.clear();
37
38
           m sfxAudioMap.clear();
39
40
           //Close and Quit Audio
           Mix CloseAudio();
41
42
           Mix Quit();
43
44
45
       //Play sound effect
       void SFX::play(int loop) {
46
           if (Mix PlayChannel(-1, m Chunk, loop) == -1) {
47
               if (Mix_PlayChannel(0, m_Chunk, loop) == -1) {
48
               SDL Log("Mix PlayChannel error: %s", Mix GetError());
49
50
51
52
53
54
       //Load Sound effect file
```

```
55
       //@parma : string audioPath ( path to file)
56
       SFX AudioManager::loadSFX(std::string audioPath) {
57
58
            std::string temp;
59
            temp = FileIO::getInstance()->getWritablePath() + audioPath;
60
61
62
           //Load SFX music (Mix Chunk)
63
            SFX sfx;
64
            Mix Chunk* sfxChunk = nullptr;
65
           //Check if allready cached
66
67
           auto it = m sfxAudioMap.find(temp);
68
69
           //Not cached so load and cahe it
70
           if (it == m sfxAudioMap.end()) {
71
                if ((sfxChunk = Mix LoadWAV(temp.c str())) == NULL) {
72
                    SDL_Log("Mix_LoadWAV: Failed to load Audio %s", Mix_GetError());
73
74
                sfx.m Chunk = sfxChunk;
75
                m sfxAudioMap[temp] = sfxChunk;
76
            //it is cached
77
78
            else {
79
               sfx.m Chunk = it->second;
80
81
           return sfx;
82
       }
83
84
85
       //Load Music file
86
        //@parma : string audioPath ( path to file)
87
       Music AudioManager::loadMusic(std::string audioPath) {
88
89
            std::string temp;
90
            temp = FileIO::getInstance()->getWritablePath() + audioPath;
91
92
93
           //Check if allready cached
94
           auto it = m bgAudioMap.find(temp);
95
96
            Music music;
97
            Mix Music* mix = nullptr;
98
            //Not cached so load and cahe it
99
            if (it == m bqAudioMap.end()) {
100
                if ((mix = Mix_LoadMUS(temp.c_str())) == NULL) {
101
                    SDL Log("Mix LoadMUS: Failed to load Audio %s", Mix GetError());
102
103
                music.m Music = mix;
104
               m bgAudioMap[temp] = mix;
105
106
            //it is cached
107
            else {
108
               music.m Music = it->second;
109
110
            return music;
111
112 }
113
```

AudioManager.h File Reference

#include "EngineDefines.h"

Classes

- class DsdlEngine::SFX
- class DsdlEngine::Music
- class <u>DsdlEngine</u>::AudioManager

Namespaces

DsdlEngine

AudioManager.h

```
1 #ifndef _AUDIOMANAGER_
2 #define _AUDIOMANAGER_
 4 #include "EngineDefines.h"
 6 Author: Derek O Brien
 7 File : AudioManager.h
8 Description: Engine Audio Manger handles loading, playing and stoping of audio files
 9 */
10 //Name Space Wrapper
11 namespace DsdlEngine {
12
16
       class SFX {
17
       public:
18
           friend class AudioManager;
2.2
2.3
28
           void play(int loop = 0);
     private:
29
33
          Mix Chunk* m Chunk;
34
35
39
       class Music {
40
       public:
           friend class AudioManager;
44
45
50
           void play(int loop = -1) { Mix PlayMusic(m Music, loop); };
51
55
           void audioPauseBG() { Mix PauseMusic(); };
56
60
            void audioResumeBG() { Mix ResumeMusic(); };
61
65
           void audioStopBG() { Mix_HaltMusic(); };
66
67
       private:
68
72
            Mix_Music* m Music;
73
74
75
       class AudioManager {
81
82
        public:
87
           AudioManager() { init(); };
88
93
            ~AudioManager() { destroy(); };
94
99
            void init();
100
104
           void destroy();
```

```
105
 111
112
            SFX loadSFX(std::string audioPath);
118
            Music loadMusic(std::string audioPath);
119
120
124
       private:
            std::map<std::string, Mix_Chunk*> m sfxAudioMap;
 125
 129
            std::map<std::string, Mix_Music*> m_bgAudioMap;
 130
 134
            bool m bisInitialized;
 135
         };
 136 }
 137
138 #endif
```

Button.cpp File Reference

#include "Button.h"
#include "IScene.h"

Namespaces

DsdlEngine

Button.cpp

```
1 #include "Button.h"
    2 #include "IScene.h"
    4 namespace DsdlEngine {
    5
           Button::Button() {
    7
                setEngineNodeType(NodeType::LABEL);
    8
                m eCurrentState = ButtonState::NORMAL;
    9
   10
   11
           Button::~Button() { destroy(); }
   12
   13
           //create button as label
           void Button::createTextButton(Vec2 pos, Size btnsize, std::string buttonText, std::string
   14
fontPath, SDL Color color, SDL Color bgColor) {
  15
                m size.y = btnsize.h ;
m size.x = buttonText.length() * btnsize.h ;
   16
   17
   1.8
               \frac{\text{m position.} \underline{x}}{\text{m position.} \underline{y}} = \text{pos.} \underline{x};
   19
   20
   21
   22
   2.3
               m labelText = buttonText;
               m textSize = btnsize.h ;
   24
              m textColor = color;
   2.5
   26
               setAssetPath(fontPath);
   2.7
   28
               m label = new Label();
   29
                m label->create(pos, m labelText, m textSize, m textColor, fontPath);
   30
   31
          //create button as sprite
   32
   33
           void Button::createSpriteButton(Vec2 spriteSize, Vec2 position, std::string imagePath,
std::string name) {
             \underline{\text{m size.x}} = \text{spriteSize.x};
               \overline{\text{m size.y}} = \text{spriteSize.y};
   35
   36
   37
               setAssetPath(imagePath);
   38
   39
               m buttonName = name;
   40
               \frac{\text{m position.x}}{\text{m position.y}} = \text{position.x};
\frac{\text{m position.y}}{\text{position.y}} = \text{position.y};
   41
   42
   43
                setPosition(position);
   44
   4.5
               setEngineNodeType(NodeType::SPRITE);
   46
   47
               m numFrames = 1;
   48
                m spriteBtn = new Sprite();
                m spriteBtn->create(m size, m position, imagePath, 1);
   49
   50
   51
   52
           //Set State to Hovering
           void Button::onMouseEnters() {
```

```
54
            m eCurrentState = ButtonState::HOVERING;
 55
            SDL Log("Mouse over button");
 56
 57
       //Set State Back to Normal
 58
 59
        void Button::onMouseLeaves() {
 60
            m eCurrentState = ButtonState::NORMAL;
 61
 62
 63
        //Set State to Pressd, Preform Action
        void Button::onClicked() {
 64
 65
            m eCurrentState = ButtonState::PRESSED;
 66
            SDL Log("Pressed button");
 67
 68
 69
        //Check for mouse input on a button
 70
        void Button::checkInput(SDL Event& e) {
 71
 72
            //check if mouse over button
            if (e.type == SDL_MOUSEBUTTONDOWN || e.type == SDL MOUSEBUTTONUP) {
 73
 74
                int x, y;
 75
                SDL GetMouseState(&x, &y);
 76
 77
                //Check if mouse inside button area
                bool inside = true;
 78
 79
 80
                //Check if mouse inside button
 81
                if (x < m position.x ) {</pre>
 82
                    inside = false;
 83
 84
                else if (x > m position.x + m size.y) {
                   inside = false;
 85
 86
 87
                else if (y < m position.y ) {</pre>
                    inside = false;
 88
 89
 90
                else if (y > m position.y + m size.x) {
                    inside = false;
 91
 92
 93
 94
                //If mouse outside buttton
 95
                if (!inside) {
 96
                    onMouseLeaves();
 97
 98
                else {//If mouse is inside button check mouse input type
99
                   switch (e.type) {
100
                    case SDL MOUSEMOTION:
101
                        onMouseEnters();
102
                        break;
103
                    case SDL MOUSEBUTTONUP:
104
                        onMouseEnters();
105
                        break;
                    case SDL_MOUSEBUTTONDOWN:
106
107
                        onClicked();
108
                        break;
109
                        //Touch down
110
                    case SDL FINGERDOWN:
                        onClicked();
111
112
                        break;
113
                    case SDL_FINGERMOTION:
114
                        onMouseEnters();
115
                        break;
116
                    case SDL FINGERUP:
117
                        onMouseEnters();
118
                        break;
119
                    default:
120
                        break:
121
122
                }
123
            }
124
```

```
125 }
126
127 void <u>Button::destroy()</u> {
128
129 }
130 }
```

Button.h File Reference

```
#include "EngineBaseNode.h"
#include "Label.h"
#include "Sprite.h"
```

Classes

• class <u>DsdlEngine::Button</u>

Namespaces

• <u>DsdlEngine</u>

Button.h

```
1 #ifndef _BUTTON_
2 #define BUTTON
    4 #include "EngineBaseNode.h"
    6 #include "Label.h"
    7 #include "Sprite.h"
   13 namespace DsdlEngine {
   19
           class <u>Button</u> : public <u>EngineBaseNode</u> {
   20
         public:
   21
   25
               Button();
   26
               virtual ~Button();
   30
   31
   35
               void destroy();
   36
46 void \underline{\text{createTextButton}}(\underline{\text{Vec2}} \text{ pos, } \underline{\text{Size}} \text{ size, std::string buttonText, std::string fontPath, SDL Color textColor, SDL Color bgColor);}
   55
                void createSpriteButton(Vec2 spriteSize, Vec2 position, std::string imagePath,
std::string name);
   56
   62
                void checkInput(SDL Event& e);
   63
   64
   70
                std::string getButtonName() { return m buttonName; }
   71
                ButtonState m eCurrentState;
   77
           private:
   78
   83
                void onMouseEnters();
   84
   89
                void onMouseLeaves();
   90
   95
                void onClicked();
   96
   97
  101
                Label * m label;
  102
  106
                Sprite* m_spriteBtn;
  107
  111
                std::string m buttonName;
  112
  113
           };
  114 }
  115
```

116 #endif //!_BUTTON_

CollisionShane.cpp File Reference

#include "CollisionShape.h"

Namespaces

• DsdlEngine

CollisionShane.cpp

```
1 #include "CollisionShape.h"
 4 namespace DsdlEngine {
 6
       CollisionShape::CollisionShape() {
 7
         //Empty
 8
9
10
     CollisionShape::~CollisionShape() {
11
         //Empty
12
13
14
15
     void CollisionShape::init(b2World* world,
16
17
          Vec2 position,
         Vec2 dimensions,
18
19
          float density,
         float friction,
20
         bool fixedRotation) {
21
23
         m dimensions = dimensions;
24
25
          // Make the body
          b2BodyDef bodyDef;
26
27
         bodyDef.type = b2_dynamicBody;
28
         bodyDef.position.\overline{S}et(position.\underline{x}, position.\underline{y});
29
          bodyDef.fixedRotation = fixedRotation;
30
          m body = world->CreateBody(&bodyDef);
31
32
33
          // Create the box
         b2PolygonShape boxShape;
34
         boxShape.SetAsBox(dimensions.x / 2.0f, (dimensions.y - dimensions.x ) / 2.0f);
35
36
37
         b2FixtureDef fixtureDef;
         fixtureDef.shape = &boxShape;
         fixtureDef.density = density;
39
40
          fixtureDef.friction = friction;
          m fixtures[0] = m body->CreateFixture(&fixtureDef);
41
42
43
44
45
       void CollisionShape::destroy(b2World* world) {
46
       if (m body) {
47
               world->DestroyBody(m body);
48
               m body = nullptr;
49
50
      }
51 }
```

CollisionShape.h File Reference

#include "EngineDefines.h"

Classes

• class <u>DsdlEngine</u>::CollisionShape

Namespaces

• <u>DsdlEngine</u>

CollisionShape.h

```
1 #ifndef COLLISIONSHAPE
 2 #define __COLLISIONSHAPE
 4 #include "EngineDefines.h"
8 namespace DsdlEngine {
12
     class <u>CollisionShape</u>
13
     public:
14
18
          CollisionShape();
19
          ~CollisionShape();
23
24
33
         void init(b2World* world,
34
35
               Vec2 position,
Vec2 dimensions,
              float density,
36
              float friction,
37
38
               bool fixedRotation);
39
44
         void destroy(b2World* world);
45
50
          b2Body* getBody() const { return m body; }
51
          b2Fixture* getFixture(int index) const { return m fixtures[index]; }
57
58
63
          const Vec2 getDimensions() const { return m dimensions; }
64
    protected:
65
           b2Body* m_body = nullptr;
69
70
          b2Fixture* m fixtures[1];
74
75
79
           Vec2 m dimensions;
80
81
82 }
83 #endif // ! COLLISIONSHAPE
```

DsdlEngine.cpp File Reference

```
#include "EngineDefines.h"
#include "DsdlEngine.h"
```

Namespaces

• DsdlEngine

Functions

• int <u>DsdlEngine::init</u> ()

DsdlEngine.cpp

```
2 #include "EngineDefines.h"
3 #include "DsdlEngine.h"
 5 namespace DsdlEngine{
      int init(){
      //Initialize SDL
SDL Init(SDL INIT AUDIO | SDL INIT EVENTS | SDL INIT TIMER | SDL INIT VIDEO);
 8
 9
10
          SDL Log("Log Print SDL Finised Init!\n");
11
12
13
          //Initialize TTF
14
          TTF_Init();
15
           SDL GL SetAttribute(SDL GL ACCELERATED VISUAL, 1);
16
17
18
          return 0;
19
20 }
```

DsdlEngine.h File Reference

```
#include "AudioManager.h"
#include "Button.h"
#include "CollisionShape.h"
#include "EngineBaseNode.h"
#include "EngineDefines.h"
#include "EngineMaster.h"
#include "EngineMath.h"
#include "EngineError.h"
#include "FileIO.h"
#include "Gui.h"
#include "IMainGame.h"
#include "InputManager.h"
#include "IScene.h"
#include "Label.h"
#include "Layer.h"
#include "ResourceTexture.h"
#include "SceneManager.h"
#include "Sprite.h"
#include "Timing.h"
#include "XmlLocalStorage.h"
#include "Window.h"
```

Namespaces

DsdlEngine

Functions

• int DsdlEngine::init ()

DsdlEngine.h

```
1 #ifndef DSDLENGINE
 3 #include "AudioManager.h"
 4 #include "Button.h"
 5 #include "CollisionShape.h"
6 #include "EngineBaseNode.h"
 7 #include "EngineDefines.h"
 8 #include "EngineMaster.h"
9 #include "EngineMath.h"
10 #include "EngineError.h"
11 #include "FileIO.h"
12 #include "Gui.h"
13 #include "IMainGame.h"
14 #include "InputManager.h"
15 #include "IScene.h"
16 #include "Label.h"
17 #include "Layer.h"
18 #include "ResourceTexture.h"
19 #include "SceneManager.h"
20 #include "Sprite.h"
21
22 #include "Timing.h"
23 #include "XmlLocalStorage.h"
25 #include "Window.h"
```

```
29 namespace DsdlEngine {
33    extern int init();
34 }
35
36 #endif // !_DSDLENGINE_
37
```

EngineBaseNode.cpp File Reference

#include "EngineBaseNode.h"
#include "FileIO.h"

Namespaces

DsdlEngine

EngineBaseNode.cpp

```
2 #include "EngineBaseNode.h"
3 #include "FileIO.h"
 7 namespace DsdlEngine {
8
 9
       //Constructor
10
       EngineBaseNode::EngineBaseNode() {
11
12
           m engineTexture = NULL;
13
           setEngineNodeType (NodeType::BASENODE);
14
           m frame = 0;
1.5
          m numFrames = 1;
16
           m opacity = 255;
17
           m objectBoundingBox = new SDL Rect();
18
19
           updateTextureInfo = false;
2.0
       }
21
22
       //Deconstructor
23
       EngineBaseNode::~EngineBaseNode() {
24
           destroy();
25
26
2.7
28
29
       //Render Node by type
30
       void EngineBaseNode::render(SDL Renderer* r) {
31
           if (nodeType == NodeType::SPRITE) {
32
               m currentFrame = &m gSpriteClips[m frame / m numFrames];
33
               m engineTexture->setAlpha(m opacity);
34
35
               //Draw Bounding Box
               //SDL SetRenderDrawColor(r, 0, 255, 255, 255);
36
37
               //SDL RenderDrawRect(r, m objectBoundingBox);
38
39
               //render texture
40
               m engineTexture->render(m position, m size, r, m currentFrame);
41
               ++m frame;
42
               if (m frame / m numFrames) = m numFrames) {
43
44
                   m frame = 0;
4.5
46
47
           else if (\underline{nodeType} == \underline{NodeType::LABEL}) {
               m engineTexture->render(m position, m size, r);
48
49
           else if (nodeType == NodeType::BUTTON) {
50
51
               m engineTexture->render(m position, m size, r);
52
53
           else if (nodeType == NodeType::PARTICLE) {
54
55
56
57
58
```

```
59
 60
        //Load Node as engine texture
 61
        bool EngineBaseNode::load (SDL Renderer * r) {
62
63
            m engineTexture = new ResourceTexture();
 64
            m objectBoundingBox = new SDL Rect();
65
 66
            if (nodeType == NodeType::SPRITE || nodeType == NodeType::BUTTON) {
67
                if (!m engineTexture->loadTexture(m assetPath, r))
 68
                    SDL Log("Faild to load sprite");
69
70
                else {
 71
 72
                    int temp = 0;
73
                    for (int i = 0; i < m numFrames; i++) {</pre>
 74
 75
                        m gSpriteClips[i].x = temp;
                         m gSpriteClips[i].y = 0;
 76
                         m gSpriteClips[i].w = m size.x ;
77
 78
                        m gSpriteClips[i].h = m size.y;
 79
80
                         temp += m size.x ;
81
82
                        m \text{ objectBoundingBox} \rightarrow x = m \text{ position.x};
83
                        m objectBoundingBox->y = m position.y;
84
                        m objectBoundingBox->w = m size.x ;
85
                        m objectBoundingBox->h = m size.y;
86
87
88
                return true;
89
90
            else if (nodeType == NodeType::LABEL) {
91
92
                if (!TTF WasInit()) {
93
94
                    TTF Init();
95
96
97
                std::string temp;
98
99
                temp = FileIO::getInstance()->getWritablePath() + m assetPath;
100
101
                //Using getWritablePath
102
                FileIO::getInstance()->getWritablePath() + m assetPath;
103
104
                //Check if font in chache
105
                auto it = m FontMap.find(temp);
106
107
                // if not load and create texture
108
                if (it == m FontMap.end()) {
109
110
                    //open font
                    m font = TTF_OpenFont(temp.c_str(), m textSize);
111
112
                    if (m font == NULL) {
113
                         SDL_Log("TTF_OpenFont Error : %s",TTF_GetError());
114
115
116
                    m engineTexture->loadTTF(m labelText, m textColor, m font, r);
117
118
                    m FontMap[temp] = m font;
119
120
                else {//create texture
121
                    m font = it->second;
122
                    m engineTexture->loadTTF(m labelText, m textColor, m font, r);
123
124
125
                return true;
126
127
            else
128
                return false;
129
```

```
130
131
        void EngineBaseNode::setBoundingBox(Vec2 pos, Vec2 size) {
132
         m objectBoundingBox = new SDL Rect();
            m objectBoundingBox->x = pos.x;
133
134
            m objectBoundingBox->y = pos.y ;
135
            m objectBoundingBox->w = size.x;
136
            m objectBoundingBox->h = size.y;
137
138
139
        void EngineBaseNode::updateLabelText(std::string text) {
140
            m labelText = text;
141
142
143
        void EngineBaseNode::setOpacity(int op) {
144
145
            if (op > 255 \mid \mid op < 0) {
146
                SDL Log("Invalid opacity value passed in.");
147
                \underline{m} opacity = 255;
148
149
            else {
150
                m opacity = op;
151
       }
152
153
154
        void EngineBaseNode::destroy() {
155
          m engineTexture->destroy();
156
            m objectBoundingBox = nullptr;
157
            m currentFrame = nullptr;
158
            m frame = 0;
159
            m numFrames = 0;
160
            m opacity = 0;
            m CollisionShape = nullptr;
161
162
            m \text{ size} = Vec2(0, 0);
163
164
165
        void EngineBaseNode::cleanup() {
166
            m engineTexture->destroy();
167
168
169
170
        void EngineBaseNode::renderCollisionShape(SDL Renderer* r, CollisionShape* shape) {
171
172
            SDL Rect collisionbox;
173
            collisionbox.x = shape->getBody()->GetPosition().x;
            collisionbox.y = shape->getBody()->GetPosition().y;
collisionbox.w = shape->getDimensions().x;
174
175
176
            collisionbox.h = shape->getDimensions().y;
177
178
            SDL_SetRenderDrawColor(r, 0, 255, 0, 255);
179
            SDL RenderDrawRect(r, &collisionbox);
180
181
182 }
```

EngineBaseNode.h File Reference

```
#include "EngineDefines.h"
#include "ResourceTexture.h"
#include "CollisionShape.h"
```

Classes

class DsdlEngine::EngineBaseNode

Namespaces

DsdlEngine

EngineBaseNode.h

```
1 #ifndef _ENGINEBASENODE_
  2 #define ENGINEBASENODE
 4 #include "EngineDefines.h"
 5 #include "ResourceTexture.h"
 6 #include "CollisionShape.h"
 14 namespace DsdlEngine {
     class EngineBaseNode {
 18
        public:
 19
 23
            EngineBaseNode();
 24
            virtual ~EngineBaseNode();
 28
 33
            virtual void destroy();
 34
 38
            virtual void cleanup();
 39
 45
            bool load (SDL Renderer* r);
 46
            void render (SDL_Renderer* r);
 51
 52
 53
 59
            void renderCollisionShape (SDL Renderer* r, CollisionShape* shape);
 60
 65
            void setPosition(const Vec2& pos) { m position.x = pos.x , m position.y = pos.y ; };
 66
            void setPositionX(int x) { m position.x = x; }
 72
            void \underline{\text{setPositionY}}(\text{int y}) \ \{ \underline{\text{m position}}.\underline{\text{y}} = y; \ \}
 78
 83
            const Vec2 getPosition() const { return m position; }
 84
 89
            void setSize(Size si) { m size.x = si.w; m size.y = si.h; }
 90
95
            void setWidth(int w) { m size.x = w; };
 96
101
            void setHeight(int h) { m size.y = h; };
102
107
            const Vec2 getContentSize() const { return m size; }
108
109
            //Set Anchor Point
110
             //TODO
111
            //Rotate Node
112
113
             //TODO
114
```

```
119
             void scaleNode(float scale) { m size.x = m size.x * scale; m size.y = m size.y *
scale; }
 120
 125
             void scaleWidth(float scale) { m size.x = m size.x * scale; }
 126
  131
             void scaleHeight(float scale) { m size.y = m size.y * scale; }
  132
  133
  138
             void <u>setAssetPath</u>(std::string path) { <u>m_assetPath</u> = path; }
  139
  144
             std::string getAssetsPath() { return m assetPath; }
  145
  150
             NodeType getNodeType() { return nodeType; }
  151
  156
             void <u>setEngineNodeType</u>(NodeType type) { nodeType = type; }
  157
  162
             void setOpacity(int opacity);
  163
  168
             ResourceTexture* getEngineTexture() { return m engineTexture; }
  169
  174
             void updateLabelText (std::string text);
  175
             SDL_Rect* getBoundingBox() { return m objectBoundingBox; }
  180
  181
  187
             void <u>setBoundingBox(Vec2</u> pos, <u>Vec2</u> size);
  188
  189
  194
              void setUpdateTextureTrue(bool value) { updateTextureInfo = value; }
  195
  200
             bool isTextureChanged() { return updateTextureInfo; }
  201
  202
         protected:
  203
  204
              //EngineBaseNode* m node;
                                                                       /**< EngineBaseNode node*/
  205
              std::string m assetPath;
  206
              NodeType nodeType = NodeType::BASENODE;
              ResourceTexture* m engineTexture;
  208
  209
              SDL Rect* m objectBoundingBox;
  211
              //nodes position Vec2
  212
              Vec2 m position;
  213
              Vec2 m size;
  215
              int m numFrames, m frame, m opacity;
  217
              bool updateTextureInfo;
  219
             SDL_Rect m gSpriteClips[14];
  220
             SDL Rect* m currentFrame;
              // For labels
  223
  224
             TTF Font* m font;
  225
             std::map<std::string, TTF Font*> m FontMap;
  227
              std::string m labelText;
  228
             int m textSize;
  229
             SDL Color m textColor;
  232
              //Node Collision shape
              CollisionShape* m CollisionShape;
  233
  234
  235 }
  236
 237 #endif // ! ENGINEBASENODE
```

EngineDefines.h File Reference

```
#include <SDL.h>
#include "../dependencies/SDL2/SDL_image/SDL_image.h"
#include "../dependencies/SDL2/SDL_ttf/SDL_ttf.h"
#include "../dependencies/SDL2/SDL_mixer/SDL_mixer.h"
#include <string>
#include <iostream>
#include <memory>
#include <vector>
#include <map>
#include <unordered_map>
#include <functional>
#include "EngineError.h"
#include "EngineMath.h"
#include <Box2D\Box2D.h>
```

Namespaces

DsdlEngine

Macros

- #define <u>USING NS DSDL</u> using namespace DsdlEngine
- #define <u>NS_DSDL_START</u> namespace DsdlEngine{
- #define <u>NS_DSDL_END</u> }
- #define <u>DEFAULT_ROOT_N</u>AME "DefaultRoot"
- #define XML FILE "Default.xml"
- #define TOTAL PARTICLES 30
- #define METRESTOPIXELS 30 Box2D scaling defines.
- #define <u>PIXELSTOMETRES</u> 1/30.0f
- #define <u>RADTODEG</u> (-180/3.1415926536f)
- #define **DEGTORAD** -0.0174532925199432957f
- #define GRAVITYSCALE 9.0f

Typedefs

- typedef SDL TimerID DsdlEngine::CallBackTimer
- typedef SDL TimerCallback DsdlEngine::CallBack

Enumerations

- enum <u>DsdlEngine::NodeType</u> { <u>DsdlEngine::NodeType::BASENODE</u>, <u>DsdlEngine::NodeType::SPRITE</u>, DsdlEngine::NodeType::LABEL, DsdlEngine::NodeType::BUTTON, DsdlEngine::NodeType::PARTICLE }
- enum <u>DsdlEngine::ButtonState</u> { <u>DsdlEngine::ButtonState::NORMAL</u>, <u>DsdlEngine::ButtonState::PRESSED</u>, DsdlEngine::ButtonState::HOVERING }
- enum <u>DsdlEngine::ButtonType</u> { <u>DsdlEngine::ButtonType::LABEL_BTN</u>, <u>DsdlEngine::ButtonType::SPRITE_BTN</u> }
- enum <u>DsdlEngine::LableType</u> { <u>DsdlEngine::LableType::LABEL_STATIC</u>,
 DsdlEngine::LableType::LABEL_DYNAMIC }

Macro Definition Documentation

#define DEFAULT_ROOT_NAME "DefaultRoot"

Definition at line <u>77</u> of file <u>EngineDefines.h</u>.

#define DEGTORAD -0.0174532925199432957f

Definition at line 88 of file EngineDefines.h.

#define GRAVITYSCALE 9.0f

Definition at line 90 of file EngineDefines.h.

#define METRESTOPIXELS 30

Box2D scaling defines.

Definition at line <u>85</u> of file <u>EngineDefines.h</u>.

#define NS_DSDL_END }

Definition at line <u>32</u> of file <u>EngineDefines.h</u>.

#define NS_DSDL_START namespace DsdlEngine{

Definition at line <u>31</u> of file <u>EngineDefines.h</u>.

#define PIXELSTOMETRES 1/30.0f

Definition at line <u>86</u> of file <u>EngineDefines.h</u>.

#define RADTODEG (-180/3.1415926536f)

Definition at line 87 of file EngineDefines.h.

#define TOTAL_PARTICLES 30

Definition at line 81 of file EngineDefines.h.

#define USING_NS_DSDL using namespace DsdlEngine

Definition at line $\underline{30}$ of file $\underline{EngineDefines.h}$.

#define XML_FILE "Default.xml"

Definition at line 78 of file EngineDefines.h.

EngineDefines.h

```
2 #ifndef ENGINEDEFINES
 3 #define _ENGINEDEFINES_
 6 #include <SDL.h>
7 #include "../dependencies/SDL2/SDL image/SDL image.h"
8 #include "../dependencies/SDL2/SDL_ttf/SDL_ttf.h"
9 #include "../dependencies/SDL2/SDL mixer/SDL mixer.h"
10
11 #include <string>
12 #include <iostream>
13 #include <memory>
14 #include <vector>
15 #include <map>
16 #include <unordered map>
17
18 #include <functional> //for std::function (CALLBACK FUNCTION)
19
20
21 #include "EngineError.h"
22 #include "EngineMath.h"
23 #include "EngineMaster.h"
24
25
26 #include <Box2D\Box2D.h>
28
29 //Set Macro Defines for Namespace
30 #define USING NS DSDL using namespace DsdlEngine 31 #define NS_DSDL_START namespace DsdlEngine{
32 #define NS DSDL END
33
34 namespace DsdlEngine {
38 enum class NodeType {
39
         BASENODE,
40
           SPRITE,
           LABEL,
41
42
           BUTTON,
43
           PARTICLE
44
45
49
       enum class ButtonState {
       NORMAL,
50
51
           PRESSED,
52
           HOVERING
53
       };
54
       enum class ButtonType {
58
59
          LABEL BTN,
60
           SPRITE BTN
61
      };
62
66
       enum class LableType {
67
           LABEL STATIC,
           LABEL DYNAMIC
68
69
70
       typedef SDL TimerID CallBackTimer;
74
       typedef SDL TimerCallback CallBack;
```

```
76
77 #define DEFAULT ROOT NAME "DefaultRoot"
78 #define XML_FILE "Default.xml"
79
80
81 #define TOTAL_PARTICLES 30
82
84
85 #define METRESTOPIXELS 30
86 #define PIXELSTOMETRES 1/30.0f
87 #define RADTODEG (-180/3.1415926536f)
88 #define DEGTORAD -0.0174532925199432957f
89
90 #define GRAVITYSCALE 9.0f
91
92 }
93
94 #endif //!_ENGINEDEFINES_
```

EngineError.cpp File Reference

#include "EngineError.h"

Namespaces

• <u>DsdlEngine</u>

EngineError.cpp

```
1 #include "EngineError.h"
2
3 namespace DsdlEngine{
4
5
6 }
```

EngineError.h File Reference

#include "EngineDefines.h"

Namespaces

<u>DsdlEngine</u>

Macros

- #define <u>DEBUG_DSDL</u> 1
- #define <u>DEBUG_MSG</u>(x) (std::cout << (x) << std::endl)

Macro Definition Documentation

```
#define DEBUG_DSDL 1
```

Definition at line 9 of file EngineError.h.

```
#define DEBUG_MSG( x) (std::cout << (x) <<std::endl)
```

Definition at line 12 of file EngineError.h.

EngineError.h

```
2 #ifndef _ENGINEERROR_
3 #define ENGINEERROR
 5 #include "EngineDefines.h"
 7 namespace <a href="DsdlEngine">DsdlEngine</a>{
       #define DEBUG DSDL 1
 9
#if defined DEBUG_DSDL
      #if (DEBUG_DSDL == 1)
11
12
                 #define DEBUG_MSG(x) (std::cout << (x) <<std::endl)</pre>
13 #C-
14 #de
#endif
               #define DEBUG_MSG(x)
16 #else
17
      #define DEBUG MSG(x)
18
       #endif
19
20 }
21
22
23 #endif
```

EngineMaster.cpp File Reference

#include "EngineMaster.h"

Namespaces

• <u>DsdlEngine</u>

Variables

• static EngineMaster * <u>DsdlEngine::Instance</u> = nullptr

EngineMaster.cpp

```
1 #include "EngineMaster.h"
2
3 namespace DsdlEngine {
4
5    //Create As Singleton
6    static EngineMaster* Instance = nullptr;
7    EngineMaster* EngineMaster::getInstance() {
8         if (!Instance) {
9             Instance = new (std::nothrow) EngineMaster();
10         }
11            return Instance;
12    }
13
14 }
```

EngineMaster.h File Reference

#include "EngineDefines.h"

Classes

• class <u>DsdlEngine::EngineMaster</u>

Namespaces

• <u>DsdlEngine</u>

EngineMaster.h

```
1 #ifndef _ENGINEMASTER_
2 #define _ENGINEMASTER_
 4 #include "<a href="EngineDefines.h"</a>"
10 namespace DsdlEngine
14 class EngineMaster{
15 public:
static EngineMaster* getInstance();
      virtual ~EngineMaster(){};
33
     private:
34
35
36
       };
37 }
39
40 #endif // ! ENGINEMASTER
```

EngineMath.cpp File Reference

#include "EngineMath.h"
#include "XmlLocalStorage.h"

Namespaces

DsdlEngine

EngineMath.cpp

```
1 #include "EngineMath.h"
  2 #include "XmlLocalStorage.h"
 7 namespace DsdlEngine(
 9
10
         //Defaults to position of (0 , 0)
11
         Vec2::Vec2() : x (0), y (0){
12
13
         //Set position to values passed in (x , y)
14
15
         \underline{\text{Vec2}::\text{Vec2}}(\text{float } x, \text{ float } y) : \underline{x}(x), \underline{y}(y) \{
16
17
18
         Vec2::Vec2(const Vec2& v) {
        this->\underline{x} = v.\underline{x};
this->\underline{y} = v.\underline{y};
19
20
21
22
23
        Vec2::~Vec2(){}
24
25
26
27
              SDL Window Coordintes
28
             origin is top left corner
29
         const \underline{\text{Vec2}} \underline{\text{Vec2}::ZERO} (0, 0);
30
31 }
32
33 namespace DsdlEngine{
35
36
         Size::Size() : w_(0), h_(0){}
37
38
         Size::Size (float w, float h) : \underline{w} (w), \underline{h} (h) {}
39
40
         Size::Size(const Size& s) {
           this->h = s.h;
this-><u>w</u> = s.w;
41
42
43
44
45
         Size::~Size(){}
46
47 }
```

EngineMath.h File Reference

Classes

- class <u>DsdlEngine::Vec2</u>
- class <u>DsdlEngine::Size</u>

Namespaces

• <u>DsdlEngine</u>

EngineMath.h

```
1 #ifndef _ENGINEMATH_
 2 #define _ENGINEMATH_
3 7 namespace DsdlEngine{
11 class Vec2{
12
    public:
13
14
19
         Vec2();
20
27
         Vec2(float x, float y);
28
34
          Vec2 (const Vec2 & v);
35
39
          <u>~Vec2</u>();
40
44
          static const Vec2 ZERO;
45
         float <u>x</u>;
46
47
          float y ;
50
51
55
     class Size{
     public:
56
57
62
          <u>Size</u>();
63
70
          Size(float w, float h);
71
72
78
          Size(const Size& s);
79
83
          ~<u>Size</u>();
84
85
          float w;
86
87
          float h;
88
      };
89
90
91 }
92 #endif
```

FilelO.cpp File Reference

#include "FileIO.h"
#include <fstream>

Namespaces

DsdlEngine

Variables

• static FileIO * DsdlEngine::Instance = nullptr

FileIO.cpp

```
1 #include "FileIO.h"
   2 #include <fstream>
   8 namespace <a href="DsdlEngine">DsdlEngine</a>{
  10
         //Create As Singleton
  11
        static FileIO* Instance = nullptr;
  12
         FileIO* FileIO::getInstance(){
  13
           if (!Instance) {
  14
  1.5
                 Instance = new (std::nothrow) FileIO();
  16
  17
             return Instance;
  18
  19
  20
  21
         std::string FileIO::getSuitableFOpen(const std::string& filenameUtf8) const{
  22
  23
             return filenameUtf8;
  2.4
  25
  26
  27
             Get Path to file
  28
  29
  30
             if defs here for different platfoms as windows needs to find assets in root folder which
i have created
  31
             but android needs to find assets in the jni/assets folder. android is allready set up
to go look in this folder
             there for all that was needed was the name and in the windows platfom i add on path to
the assets folder so it just has to look for name of file
  33
  34
              this will need to be done to each asset type loding function eq. audio, fonts, images
  35
  37
         std::string FileIO::getWritablePath() {
  38
         //For Windows
  39 #ifdef WIN32
  40
             m_path;
  41 #endif
  42
            //For Android
  44 #ifdef __ANDROID_
  m path = "";
  47
            return m path;
  48
  49
  50
  51
  52 //Loads complete file into memory
```

```
53
          bool FileIO::loadDocument(const char* filepath, char** doc contents) {
  54
   55
              //Open file
  56
              SDL RWops *file = SDL RWFromFile(filepath, "rb");
              if (file != nullptr) {
  57
   58
  59
                  //Get length of file
   60
                  size t file length = SDL RWseek(file, 0, SEEK END);
                  (*doc_contents) = new char[file_length + 1];
SDL RWseek(file, 0, SEEK SET);
  61
  62
  63
  64
                  //Read File into buffer
   65
                  int n blocks = SDL RWread(file, (*doc contents), 1, file length);
  66
  67
                  //Close file
  68
                  SDL RWclose(file);
   69
  70
                  //add null terminator to end of file
  71
                  (*doc contents)[file length] = '\0';
   72
                  return true;
  73
  74
              return false;
   75
   76
  77
  78
          //Write contents of buffer to file and save.
   79
          bool FileIO::writeDocument(const char* filepath, const char** doc contents) {
  80
              SDL RWops *file = SDL RWFromFile(filepath, "w");
  81
  82
              if (file != nullptr) {
  83
                  //Length of data to write
                  size_t len = SDL_strlen(*doc_contents);
  84
  8.5
  86
                  //Write the data
                  SDL RWwrite(file, *doc contents, 1, len);
  87
  88
  89
                  //close file
  90
                  SDL RWclose(file);
  91
  92
                  return true;
  93
              }
  94
              return false;
   95
  96
   97
          //Parse Xml for Element for Key and return Element node if found
         XMLElement* FileIO::getXMLNodeForKey(const char*pKey, XMLElement** rootNode,
  98
XMLDocument** doc) {
  99
  100
              XMLElement* curNode = nullptr;
 101
  102
              char* contents = NULL;
  103
  104
              std::string path = getWritablePath() + "Default.xml";
  105
  106
              //Check the key
  107
              if (!pKey) {
  108
                  return nullptr;
  109
              }
  110
  111
              //Load Xml document into contents char
  112
              if (FileIO::getInstance()->loadDocument(path.c str(), &contents) != true) {
  113
                  SDL Log("can not read xml file using SDL rwops");
  114
  115
  116
              //SDL Log(contents);
  117
  118
              XMLDocument* xmlDoc = new XMLDocument;
  119
              *doc = xmlDoc;
  120
  121
              if (xmlDoc->Parse(contents) == XML SUCCESS) {
 122
                 //SDL Log("Doc Parsed");
```

```
123
                // get root node
124
                *rootNode = xmlDoc->RootElement();
125
126
                if (nullptr == *rootNode) {
                     SDL Log("read root node error ");
127
128
129
130
                // find the node
                curNode = (*rootNode) ->FirstChildElement();
131
132
                while (curNode != nullptr)
133
                     const char* nodeName = curNode->Value();
134
135
                     if (!strcmp(nodeName, pKey)) {
136
                         break;
137
138
                     curNode = curNode->NextSiblingElement();
139
                }
140
141
            else {
                SDL Log("Could not load doc: ");
142
143
144
145
            delete[] contents;
146
147
            return curNode;
148
149
150
151
152
            Set Value for key in xml file
153
            @parma key = name of node to be written to file
154
            @parma vale = vale of node to be saved
155
156
        void FileIO::setValueForKey(const char* value, const char* key) {
157
158
            XMLElement* rootNode;
            XMLDocument* doc;
159
160
            XMLElement* node;
            XMLPrinter printer;
161
162
            std::string path;
163
164
165
            if (!key || !value) {
166
                return;
167
168
169
            path = getWritablePath() + "Default.xml";
170
171
            //Check if node exists allready
172
            node = getXMLNodeForKey(key, &rootNode, &doc);
173
174
            //if node allready exists change value
175
            if (node) {
176
                if (node->FirstChild()) {
177
                    node->FirstChild()->SetValue(value);
178
179
                else {
                    XMLText* content = doc->NewText(value);
180
181
                    node->LinkEndChild(content);
182
183
            }//Create new node and set value
184
            else {
185
                if (rootNode) {
                    XMLElement* temp = doc->NewElement(key);
186
                     rootNode->LinkEndChild(temp);
187
                     XMLText* content = doc->NewText(value);
temp->LinkEndChild(content);
188
189
190
191
192
193
            // attach printer to the document you want to convert in to a std::string
```

```
194
             doc->Accept(&printer);
  195
  196
             // Create a std::string and copy your document data in to the string
 197
             const char* buffer = printer.CStr();
 198
  199
             //Write back to file and save file
  200
             if (FileIO::getInstance()->writeDocument(path.c_str(), &buffer)) {
  201
                 SDL Log("Key: %s:: Value: %s:: saved", key, value);
  202
  203
             delete doc;
  204
  205
  206
         //Create XML File
  207
         bool FileIO::createXMLFile() {
  208
             bool bRet = false;
  209
  210
             XMLPrinter printer;
  211
  212
             XMLDocument *doc = new XMLDocument();
             if (nullptr == doc) {
  213
  214
                 return false;
  215
 216
  217
             XMLDeclaration *pDeclaration = doc->NewDeclaration(nullptr);
 218
             if (nullptr == pDeclaration) {
  219
                 return false;
  220
  221
  222
             doc->LinkEndChild(pDeclaration);
  223
             XMLElement *pRootEle = doc->NewElement(DEFAULT ROOT NAME);
  224
             if (nullptr == pRootEle) {
 225
                 return false;
  226
  227
 228
             doc->LinkEndChild(pRootEle);
 229
 230
             std::string path;
 231
             path = getWritablePath() + "Default.xml";
 2.32
 233
 234
             bRet = XML SUCCESS ==
 235
doc->SaveFile(FileIO::getInstance()->getSuitableFOpen(path).c str());
 236
  237
             if (doc) delete doc;
 238
 239
             return bRet;
 240
 241
242 }
```

FileIO.h File Reference

```
#include "EngineDefines.h"
#include <sys/stat.h>
#include "../dependencies/tinyxml/tinyxml2.h"
```

Classes

• class <u>DsdlEngine::FileIO</u>

Namespaces

DsdlEngine

FileIO.h

```
1 #ifndef _FILEIO_
   2 #define FILEIO
   4 #include "EngineDefines.h"
   5 #include <sys/stat.h>
   6 #include "../dependencies/tinyxml/tinyxml2.h"
  11 namespace <a href="DsdlEngine">DsdlEngine</a>{
  12
         using namespace tinyxml2;
  14
  15
         using namespace std;
  16
  21
         class FileIO{
  22
         public:
  23
  28
             static FileIO* getInstance();
  29
  35
              std::string getSuitableFOpen(const std::string& filenameUtf8) const;
  36
  41
              std::string getWritablePath();
  42
              void setAssetsPath( std::string assetsPath) { m path = assetsPath; }
  47
  48
  49
  54
              std::string getFileToOpen() { return m fileName; }
  55
  60
              void setFileToOpen(std::string file) { m fileName = file; }
  61
              bool loadDocument(const char* filepath, char** doc contents);
  68
  69
  76
              bool writeDocument(const char* filepath, const char** doc contents);
  8.5
              XMLElement* getXMLNodeForKey(const char*pKey, XMLElement** rootNode, XMLDocument**
doc);
  86
  92
              void setValueForKey(const char* value, const char* key);
  93
  98
             bool createXMLFile();
  99
 100
         protected:
 104
             FileIO(){};
 105
 109
              virtual <u>~FileIO(){};</u>
 110
         private:
 111
 112
 113
              std::string m path;
 114
              std::string m fileName;
 116
          };
 117 }
```

119 #endif // !_FILEIO_

Gui.cpp File Reference

```
#include "Gui.h"
#include "Button.h"
#include "Window.h"
#include "Label.h"
```

Namespaces

• <u>DsdlEngine</u>

Gui.cpp

```
1 #include "Gui.h"
   2 #include "Button.h"
   3 #include "Window.h"
   4 #include "Label.h"
   6 /*
         File: Gui
   8
          Author: Derek O Brien
   9
          Description: GUi Layer templtate for creating an a UI Layer. Inherits from layer
  10 */
  11 namespace DsdlEngine{
  13
          //Constructor
  14
          DsdlGui::DsdlGui() {
           //Empty
  15
  16
  17
         //Deconstructor
  18
  19
          DsdlGui::~DsdlGui() {
  20
              destroy();
  21
  2.2
  23
         //Add Button to GUI
  24
         void DsdlGui::addButton(ButtonType type, std::string name, Vec2 pos, Vec2 size, std::string
path, SDL Color color, SDL Color bgColor, const char* text) {
  26
              \underline{m} \underline{btn} = \underline{new} \underline{Button}();
  27
              //Create as Text Button
  28
              if (type == ButtonType::LABEL BTN) {
  29
                  m btn->createTextButton(pos, Size(size.x_, size.y_), text, path, color, bgColor);
                  m btn->setPosition(pos);
  31
  33
              //Create as Sprite Button
  34
              if (type == ButtonType::SPRITE BTN) {
  35
                  m btn->createSpriteButton(size , pos, path, name);
  36
                  m btn->setEngineNodeType(NodeType::SPRITE);
  37
                  m btn->setPosition(pos);
  38
  39
              //Add button to gui elements array
  40
  41
              GUIElements.push back(m btn);
  42
              layerNodes.push back(m btn);
  43
  44
  45
          //Add Label to Gui Layer
  46
          void <u>DsdlGui::addLabel(LableType</u> type, <u>Vec2</u> pos, std::string text, int fontSize, SDL_Color
color, std::string fontFilePath) {
  47
              m label = new Label();
  48
  49
              m label->setType(type);
  50
              m label->create (pos, text, fontSize, color, fontFilePath);
  51
```

```
52
             layerNodes.push back(m label);
  53
  54
  55
  56
         //Add predefined label to the gui layer
  57
         void DsdlGui::addPreDefineLabel(Label* label, LableType type) {
             label->setType (type);
  58
  59
              layerNodes.push back(label);
  60
  61
  62
         //Set Gui Layer Position
  63
         void DsdlGui::setGUIPos() {
  64
  65
  66
  67
          //Event Manager for input on gui Buttons
  68
         void DsdlGui::onSDLEvent(SDL Event& e) {
             //Loop and Check each button for input
  69
  70
              for (size t i = 0; i < GUIElements.size(); i++){</pre>
  71
                  GUIElements.at(i) -> checkInput(e);
  72
  73
  74
  75
         //Destroy
         void DsdlGui::destroy() {
  76
  77
  78
             if (layerNodes.size() > 0) {
  79
  80
                  for (size t i = 0; i < layerNodes.size(); i++) {</pre>
                      layerNodes.erase(std::remove(layerNodes.begin(), layerNodes.end(),
  81
layerNodes[i]), layerNodes.end());
  82
  83
                      layerNodes[i]->destroy();
  84
  85
  86
              // layerNodes.clear();
  87
              // GUIElements.clear();
  88
  89
  90 }
  91
  92
```

Gui.h File Reference

```
#include "EngineDefines.h"
#include "Layer.h"
#include "IScene.h"
```

Classes

• class <u>DsdlEngine::DsdlGui</u>

Namespaces

• <u>DsdlEngine</u>

Gui.h

```
1 #ifndef _GUI_
2 #define GUI
   4 #include "EngineDefines.h"
   5 #include "Layer.h"
   6 #include "IScene.h"
  11 namespace <a href="DsdlEngine">DsdlEngine</a>{
  12
   13
          //Forward Decalare Classes
          class Button;
  14
  15
         class Label;
  16
   20
          class <u>DsdlGui</u> : public <u>Layer</u>{
         public:
   21
   25
              DsdlGui();
   26
   30
             virtual ~DsdlGui();
  31
              void <u>addButton</u> (<u>ButtonType</u> type, std::string name, <u>Vec2</u> pos, <u>Vec2</u> size, std::string path,
   4.3
SDL Color color, SDL Color bgColor, const char* text = NULL);
              void addLabel(LableType type, Vec2 pos, std::string text, int fontSize, SDL_Color color,
   54
std::string fontFilePath);
   5.5
              void addPreDefineLabel(Label* label, LableType type);
   61
   62
   66
              void setGUIPos();
   67
   72
              void onSDLEvent (SDL Event& e);
   77
              void destroy();
   78
   79
               std::vector<Button*> GUIElements;
   85
               Button* getButton() { return m btn; }
   86
          protected:
   87
   88
               //Variables
               Label* m label;
   89
   90
               Button* m btn;
   91
           };
   92 }
   93
  94 #endif
```

IMainGame.cpp File Reference

```
#include "IMainGame.h"
#include "SceneManager.h"
#include "IScene.h"
```

Namespaces

DsdlEngine

Functions

• template<typename T, typename... Args> std::unique_ptr< T > <u>DsdlEngine::make_unique</u> (Args &&...args)

IMainGame.cpp

```
1 #include "IMainGame.h"
 4 #include "SceneManager.h"
 5 #include "IScene.h"
8 namespace DsdlEngine {
10
      *Added template version of make unique as Ndk did not support it in its version of STL
11
12
      *Error was make unique not part of std::
13
       *After research this was the easiest solution to solve error
      *Ndk-build now builds apk as of 26/01/2016
14
16
      template<typename T, typename ...Args>
17
       std::unique ptr<T> make unique(Args&& ...args) {
18
           return std::unique ptr<T>(new T(std::forward<Args>(args)...));
19
20
       //Constructor
21
22
       IMainGame::IMainGame() {
           m pSceneManager = DsdlEngine::make_unique<SceneManager>(this);
2.3
24
25
26
       IMainGame::~IMainGame() {
27
           //Empty
28
30
31
           Main Game Loop
32
33
34
35
       void IMainGame::mainLoop() {
        if (!init()) return;
36
37
          FpsLimiter fpsLimit;
           fpsLimit.setMaxFPS(m fFps);
39
40
           setRunning();
41
           while (m bIsRunning) {
42
              fpsLimit.begin();
43
44
              m pCurrentRunning->onInput();
45
46
               update();
47
               draw();
48
49
               while (m bIsPaused == true) {
```

```
50
                     m pCurrentRunning->onInput();
  51
                 }
  52
  53
                 m fFps = fpsLimit.end();
  54
             }
  55
  56
  57
         //Call Main Update loop
  58
         void <u>IMainGame::run</u>() {
  59
             mainLoop();
  60
  61
  62
  63
  64
             Main Inputmanager control
  65
  66
         void IMainGame::onSDLEvent(SDL Event& evnt) {
             m InputManager.update();
  67
  68
             //Will keep looping until there are no more events to process
  69
             if (evnt.key.repeat == 0)
  70
  71
  72
  73
                 switch (evnt.type) {
  74
                 case SDL QUIT:
  75
                     exitGame();
  76
                     break:
                 case SDL MOUSEMOTION:
  77
  78
                     m InputManager.setMouseCoords((float)evnt.motion.x, (float)evnt.motion.y);
  79
                     break;
  80
                 case SDL KEYDOWN:
  81
                     m InputManager.pressKey(evnt.key.keysym.sym);
  82
                     break;
  83
                 case SDL KEYUP:
  84
                     m InputManager.releaseKey (evnt.key.keysym.sym);
  85
                     break;
                 case SDL_MOUSEBUTTONDOWN:
  86
  87
                     m InputManager.pressKey(evnt.button.button);
  88
                     break:
  89
                 case SDL MOUSEBUTTONUP:
  90
                     m InputManager.releaseKey(evnt.button.button);
  91
                     break;
  92
                 case SDL FINGERDOWN:
  93
                     m InputManager.isSwipe (evnt);
  94
                     break;
                 case SDL FINGERMOTION:
  95
  96
                     m InputManager.isSwipe(evnt);
  97
                     break;
  98
                 case SDL FINGERUP:
  99
                     m InputManager.releaseKey(evnt.tfinger.fingerId);
 100
                     break;
  101
                 default:
 102
                     break:
 103
 104
             }
 105
         }
 106
 107
 108
             Get users window information
 109
 110
         void IMainGame::setupWindow(int w, int h, std::string windowName, std::string path, int
flag) {
 111
             m windowWidth = w;
 112
             m windowHeight = h;
             windowtitle = windowName;
 113
 114
             windowFlag = flag;
 115
 116
             mainAssetsPath = path;
             //Set windowss asset path
 117
 118 #ifdef
             WIN32
             FileIO::getInstance()->setAssetsPath(mainAssetsPath);
 119
```

```
120 #endif // ! WIN32
 121
 122
 123
 124
 125
 126
 127
             Init all Engine elements
 128
 129
         bool IMainGame::init() {
 130
             //Init Engine
 131
             DsdlEngine::init();
 132
             //Init audio Manager
 133
             m audioManager.init();
 134
 135
             //call game's on init method
 136
             onInit();
 137
 138
             //If window creation fails exit
 139
             if (!initSystems()) {
                 SDL Log("InitSystems Failed : Window not created");
 140
 141
                 return false;
 142
             }
 143
 144
             //Add all Scenes
 145
             addScenes();
 146
 147
             //Load up First Scene
 148
             m pCurrentRunning = m pSceneManager->getCurrentScene();
             m pCurrentRunning->setSceneRunning();
 149
 150
             m pCurrentRunning->onEntryScene();
 151
 152
             //Load all scene Children nodes for first scene on init of game
 153
             for (size_t i = 0; i < m pCurrentRunning->sceneLayers.size(); i++) {
 154
                 m pCurrentRunning->loadScene (m pGameRenderer);
 155
 156
 157
             //for running scene render each node that is in the child vector
 158
             for (size t i = 0; i < m pCurrentRunning->sceneLayers.size(); i++) {
 159
                 m pCurrentRunning->drawScene(m pGameRenderer);
 160
 161
 162
             return true;
 163
         }
 164
 165
 166
             InitSystem
 167
             Create window and get window render
 168
 169
         bool IMainGame::initSystems() {
 170
             m Window.createWindow(windowtitle, m windowWidth, m windowHeight, windowFlag);
             m pGameRenderer = m Window.getRenderer();
 171
 172
             return true;
 173
 174
 175
 176
 177
             Call current scenes update
 178
             Handel switching between scenes
 179
 180
 181
         void IMainGame::update() {
 182
             if (m pCurrentRunning) {
                 switch (m pCurrentRunning->getSceneState()) {
 183
                 case SceneState::RUNNING:
 184
 185
                     m pCurrentRunning->updateScene();
 186
                     break;
 187
                 case SceneState::CHANGE NEXT:
                     m_pCurrentRunning->onExitScene();
 188
 189
                     m pCurrentRunning = m pSceneManager->moveNext();
                     if (m pCurrentRunning) {
190
```

```
191
                         m pCurrentRunning->setSceneRunning();
192
                         m pCurrentRunning->onEntryScene();
193
                         //Load all scene Children nodes for next scene
194
                         for (size t i = 0; i < m pCurrentRunning->sceneLayers.size(); i++) {
195
                             m pCurrentRunning->loadScene(m pGameRenderer);
196
197
198
                     break;
199
                 case <u>SceneState::CHANGE PREVIOUS:</u>
200
                     m pCurrentRunning->onExitScene();
201
                     m pCurrentRunning = m pSceneManager->movePrevious();
202
                     if (m pCurrentRunning) {
                         m pCurrentRunning->setSceneRunning();
203
204
                         m pCurrentRunning->onEntryScene();
205
                         //Load all scene Children nodes for previous scene
                         for (size t i = 0; i < m pCurrentRunning->sceneLayers.size(); i++) {
206
207
                             m pCurrentRunning->loadScene(m pGameRenderer);
208
209
                     }
210
                    break;
211
                 case <u>SceneState::EXIT_APP</u>:
212
                     exitGame();
213
                     break;
214
                 default:
215
                     break;
216
217
            }
218
            else {
219
                 exitGame();
220
            }
221
222
223
224
225
            Render all Scene nodes to screen
226
        */
227
228
229
        void IMainGame::draw() {
230
            if (m pCurrentRunning && m pCurrentRunning->getSceneState() == SceneState::RUNNING) {
231
                 SDL_RenderClear(m pGameRenderer);
232
233
                 //for running scene render each node that is in the child vector
2.34
                 for (size_t i = 0; i < m pCurrentRunning->sceneLayers.size(); i++) {
235
                     m pCurrentRunning->drawScene (m pGameRenderer);
236
237
                 SDL RenderPresent (m pGameRenderer);
238
239
240
241
242
2.43
244
            Exit Game
245
246
         void IMainGame::exitGame() {
247
            m pCurrentRunning->onExitScene();
248
            if (m pSceneManager) {
                m pSceneManager ->destroy();
m pSceneManager.reset();
249
250
251
252
            m bIsRunning = false;
253
254 }
```

IMainGame.h File Reference

```
#include "EngineDefines.h"
#include "DsdlEngine.h"
#include "Window.h"
#include "InputManager.h"
#include "Timing.h"
#include "AudioManager.h"
#include "ResourceTexture.h"
#include "Layer.h"
```

Classes

• class <u>DsdlEngine::IMainGame</u>

Namespaces

DsdlEngine

IMainGame.h

```
1 #ifndef MAINGAME
 2 #define MAINGAME
 4 #include "EngineDefines.h"
 5 #include "DsdlEngine.h"
 6 #include "Window.h"
 7 #include "InputManager.h"
 8 #include "Timing.h"
 9 #include "AudioManager.h"
10 #include "ResourceTexture.h"
11 #include "Layer.h"
16 namespace DsdlEngine{
17
19
       class SceneManager;
20
      class IScene;
21
26
       class <u>IMainGame</u>{
27
2.8
     public:
32
           IMainGame();
33
           virtual ~IMainGame();
37
38
42
           void run();
43
           void setupWindow(int w, int h, std::string windowName, std::string path, int flag);
52
53
58
           void setFps(float fps) { m fFps = fps; }
59
64
           virtual void onInit() = 0;
65
71
           virtual void addScenes() = 0;
72
77
           virtual void onExit() = 0;
78
83
           void onSDLEvent(SDL Event& evnt);
84
88
           void setPaused() { m bIsPaused = true; }
89
93
           void setRunning() { m bIsPaused = false; m bIsRunning = true; }
94
99
            bool <u>checkPaused() { return m bIsPaused; }</u>
100
```

```
101
            InputManager m InputManager;
102
       protected:
103
          //Scene Manager
            std::unique ptr<SceneManager> m pSceneManager;
104
106
            //Current Scene
            IScene* m pCurrentRunning;
bool m bIsRunning, m bIsPaused;
107
108
110
            Window m Window;
            SDL_Renderer* m_pGameRenderer;
111
113
            AudioManager m audioManager;
115
       private:
116
117
            //Game frame rate
118
            float m fFps;
120
            //Game Windows details
            unsigned int windowFlag;
121
122
            int m windowWidth;
123
            int m windowHeight;
124
            std::string windowtitle;
125
            std::string mainAssetsPath;
            const float getFps() const { return m fFps; }
132
133
137
            void mainLoop();
138
            void <u>update();</u>
143
144
149
            void draw();
150
155
            bool <u>init();</u>
156
161
            bool initSystems();
162
166
            void exitGame();
167
168
        };
169 }
170
171 #endif //! MAINGAME
```

InputManager.cpp File Reference

#include "InputManager.h"

Namespaces

DsdlEngine

InputManager.cpp

```
1 #include "InputManager.h"
 3 namespace DsdlEngine{
 5
 6
       InputManager::InputManager(){
        swipedown = false;
 7
 8
           swipeup = false;
           swipeleft = false;
9
10
           swiperight = false;
11
12
13
       InputManager::~InputManager() {
14
          //Empty
15
16
17
18
       void InputManager::update() {
        // Loop through _keyMap using a for each loop, and copy it over to _previousKeyMap
for (auto& it : keyMap) {
19
20
               previousKeyMap[it.first] = it.second;
21
22
23
24
25
       void <u>InputManager::pressKey(unsigned int keyID)</u> {
26
         // if keyID doesn't already exist in keyMap, it will get added
27
           keyMap[keyID] = true;
28
29
       void <u>InputManager::releaseKey</u>(unsigned int keyID) {
30
            keyMap[keyID] = false;
31
32
33
34
       void InputManager::setMouseCoords(float x, float y) {
35
36
37
39
       bool InputManager::isKeyDown(unsigned int keyID) {
40
          // We dont want to use the associative array approach here
           // because we don't want to create a key if it doesnt exist.
41
42
           // So we do it manually
           auto it = keyMap.find(keyID);
if (it != keyMap.end()) {
43
44
               // Found the key
45
46
               return it->second;
47
48
           else {
49
               // Didn't find the key
50
               return false;
51
52
53
54
       bool InputManager::wasKeyDown(unsigned int keyID) {
55
          // We dont want to use the associative array approach here
56
           // because we don't want to create a key if it doesnt exist.
           // So we do it manually
```

```
auto it = __previousKeyMap.find(keyID);
if (it != __previousKeyMap.end()) {
 58
 59
                // Found the key
 60
 61
                return it->second;
 62
 63
            else {
 64
                // Didn't find the key
 65
                return false;
 66
            }
 67
 68
        bool InputManager::isKeyPressed(unsigned int keyID) {
 69
 70
            // Check if it is pressed this frame, and wasn't pressed last frame
 71
             if (isKeyDown(keyID) == true && wasKeyDown(keyID) == false) {
 72
                pressKey(keyID);
 7.3
                return true;
 74
 75
            return false;
 76
 77
 78
 79
        bool InputManager::isKeyReleased(unsigned int keyID) {
 80
            // Check if it is pressed this frame, and wasn't pressed last frame
 81
             if (isKeyDown(keyID) == false && wasKeyDown(keyID) == true) {
 82
                releaseKey(keyID);
 83
                 return true;
 84
 85
            return false;
 86
 87
        }
 88
 89
 90
        bool InputManager::isTouch(unsigned int keyID) {
 91
 92
             if (keyID == SDL FINGERDOWN) {
 93
                return true;
 94
 95
            return false;
 96
 97
 98
 99
100
        bool InputManager::isSwipe(SDL Event& evnt) {
101
102
             float startX, startY, endX, endY;
103
104
            if (evnt.type = SDL FINGERMOTION) {
105
106
                 startX = ((float)evnt.tfinger.x);
                startY = ((float)evnt.tfinger.y);
107
                fingerDown = true;
108
109
                 if (evnt.type = SDL_FINGERUP) {
110
111
                     endX = ((float)evnt.tfinger.x);
112
                     endY = ((float)evnt.tfinger.y);
113
                     fingerUp = true;
114
115
            }
116
117
118
            if (fingerDown && fingerUp) {
119
120
                 if (startX < endX) {
121
                     //swipe down
                     SDL Log("SWIPE DOWN---Start : %f ---- End : %f", startX, endX);
122
123
                     swipedown = true;
124
125
                else if (startX > endX) {
126
                     //swipe up
127
                     SDL Log("SWIPE UP---Start : %f ---- End : %f", startX, endX);
128
                     swipeup = true;
```

```
129
 130
                 else if (startY > endY) {
 131
                    //swipe left
                     swipeleft = true;
 132
                     SDL Log("SWIPE LEFT---Start : %f ---- End : %f", startY, endY);
 133
 134
 135
                 else if (startY < endY) {
 136
                     //swipe right
                     swiperight = true;
 137
 138
                     SDL Log("SWIPE RIGHT---Start: %f ---- End: %f", starty, endy);
 139
 140
                 else if (startX == endX) {
 141
                     SDL Log("Only tough happened not swipe");
 142
 143
 144
            return true;
 145
 146
 147
 148
 149
         bool <u>InputManager::isSwipeUp()</u> {
 150
 151
             return swipeup;
 152
 153
 154
         bool InputManager::isSwipeDown() {
 155
 156
             return swipedown;
 157
 158
 159
         bool InputManager::isSwipeLeft(float x, float y) {
 160
 161
            return swipeleft;
 162
 163
 164
         bool InputManager::isSwipeRight(float x, float y) {
 165
 166
            return swiperight;
 167
 168
169 }
```

InputManager.h File Reference

#include "EngineDefines.h"

Classes

• class <u>DsdlEngine::InputManager</u>

Namespaces

• <u>DsdlEngine</u>

InputManager.h

```
1 #ifndef INPUTMANAGER
  2 #define INPUTMANAGER
  4 #include "EngineDefines.h"
11 namespace DsdlEngine{
15
       class InputManager{
       public:
 16
 20
            InputManager();
 21
 25
            ~InputManager();
 26
 30
            void <u>update();</u>
 31
            void pressKey(unsigned int keyID);
 36
 37
 42
            void releaseKey(unsigned int keyID);
 43
 49
            void setMouseCoords(float x, float y);
 50
 56
            bool isKeyDown(unsigned int keyID);
 63
             bool <u>isKeyPressed</u>(unsigned int keyID);
 64
            bool isKeyReleased(unsigned int KeyID);
 70
 71
 77
            bool <u>isTouch</u>(unsigned int keyID);
 78
 84
            bool isSwipe(SDL Event& evnt);
 85
            bool isSwipeUp();
 91
 96
            bool isSwipeDown();
97
102
             bool isSwipeLeft(float x, float y);
103
108
             bool isSwipeRight(float x, float y);
109
110
        private:
111
117
            bool wasKeyDown(unsigned int keyID);
118
            std::unordered_map<unsigned int, bool> keyMap;
std::unordered_map<unsigned int, bool> previousKeyMap;
119
120
           bool swipeup, swipedown, swipeleft, swiperight;
122
123
            bool <u>fingerDown</u>, <u>fingerUp</u>;
124
        };
125 }
126 #endif
```

IScene.h File Reference

```
#include "EngineDefines.h"
#include "EngineBaseNode.h"
#include "Sprite.h"
#include "InputManager.h"
#include "Layer.h"
```

Classes

• class <u>DsdlEngine::IScene</u>

Namespaces

<u>DsdlEngine</u>

Macros

• #define <u>SCENE INDEX NO SCENE</u> -1

Enumerations

 enum <u>DsdlEngine::SceneState</u> { <u>DsdlEngine::SceneState::NONE</u>, <u>DsdlEngine::SceneState::RUNNING</u>, <u>DsdlEngine::SceneState::EXIT_APP</u>, <u>DsdlEngine::SceneState::CHANGE_NEXT</u>, <u>DsdlEngine::SceneState::CHANGE_PREVIOUS</u> }

Macro Definition Documentation

#define SCENE_INDEX_NO_SCENE -1

Definition at line 4 of file IScene.h.

IScene.h

```
1 #ifndef _ISCENE_
2 #define _ISCENE_
 4 #define SCENE INDEX NO SCENE -1
 5 #include "EngineDefines.h"
6 #include "EngineBaseNode.h"
7 #include "Sprite.h"
8 #include "InputManager.h"
9 #include "Layer.h"
10
15 namespace DsdlEngine {
16
17
        //forward declartion of class
       class IMainGame;
18
19
24
       enum class SceneState {
        NONE,
25
26
            RUNNING,
            EXIT APP,
27
28
            CHANGE NEXT,
            CHANGE PREVIOUS
29
30
```

```
31
 35
        class <u>IScene</u> {
 36
        public:
 40
            IScene() {
 41
                //Empty
 42
 4.3
 47
            virtual ~IScene() {
 48
                //Empty
 49
 50
 5.5
            virtual int getNextSceneIndex() const = 0;
 56
 61
            virtual int getPreviousSceneIndex() const = 0;
 62
 63
            // Called when a screen enters and exits focus
 64
 69
            virtual void onEntryScene() = 0;
 70
 75
            virtual void onExitScene() = 0;
 76
 81
            virtual void updateScene() = 0;
 82
 87
            virtual void destroyScene() = 0;
 88
 93
            int getSceneIndex() const { return m iSceneIndex; }
 94
99
            SceneState getSceneState() const { return m eCurrentState; }
100
104
            void setSceneRunning() { m eCurrentState = SceneState::RUNNING; }
105
            void <u>setParentGame(IMainGame</u>* game) { m_game = game; }
110
111
115
            virtual void onInput();
116
117
            std::vector<Layer*> sceneLayers;
            void addLayerToScene (Layer* layer) {
123
124
                sceneLayers.push back(layer);
125
126
131
            void <u>loadScene</u>(SDL_Renderer* r) {
132
                for (size t i = 0; i < sceneLayers.size(); i++) {
133
                    sceneLayers.at(i)->loadNodes(r);
134
135
            }
136
141
            void drawScene(SDL Renderer* r) {
142
                for (size_t i = 0; i < sceneLayers.size(); i++) {
                     sceneLayers.at(i)->drawNodes(r);
143
144
            }
145
146
147
        protected:
149
           friend class SceneManager;
150
            friend class InputManager;
151
152
            SceneState m eCurrentState = SceneState::NONE;
            IMainGame * m game = nullptr;
154
155
            int m iSceneIndex = SCENE INDEX NO SCENE;
156
            InputManager m inputManager;
158
159 }
160
161
162 #endif //! ISCENE
```

Label.cpp File Reference

#include "Label.h"

Namespaces

• DsdlEngine

Label.cpp

```
1 #include "Label.h"
   2 /*
    3
         Base Label Class
   4
          author: Derek O Brien
          Description: label class for creating labels. inherits for EngineBaseNode
   5
   6 */
   8 namespace DsdlEngine{
  10
         //Constructor
  11 <u>Label::Label</u>(){
            setEngineNodeType (NodeType::LABEL);
  12
  13
  14
  //Deconstructor
Label::~Label() {
  17
            destroy();
  18
  19
  20
        //Create Label
        void Label::create(Vec2 pos, std::string text, int txtsize, SDL Color color, std::string
  21
fontFilePath) {
  23
             m labelText = text;
            m_textSize = txtsize;
m_textColor = color;
  24
  25
  26
             setAssetPath(fontFilePath);
  27
  28
             \frac{\text{m position.x}}{\text{m position.y}} = \frac{\text{pos.x}}{\text{pos.y}};
  29
        }
  31
  32
  33 //Destroy Label
  34 void Label::destroy(){
              EngineBaseNode::destroy();
  35
  36
  37
  38
         //Cleanup Label
  39
          void Label::cleanup() {
            EngineBaseNode::cleanup();
  40
  41
  42 }
```

Label.h File Reference

#include "EngineBaseNode.h"

Classes

• class <u>DsdlEngine::Label</u>

Namespaces

• DsdlEngine

Label.h

```
1 #ifndef _LABEL_
   2 #define _LABEL_
   4 #include "EngineBaseNode.h"
  8 namespace DsdlEngine{
  12 class <u>Label</u>: public <u>EngineBaseNode</u>{
       public:
  13
            /***
* COnstructor
  14
  15
  16
            Label();
  17
  18
  22
           virtual ~Label();
  23
           void create(Vec2 pos, std::string text, int fontSize, SDL_Color color, std::string
  32
fontFilePath);
  33
  38
           void setType(LableType type) { m labelType = type; };
  39
  44
           const int getType() { return (int)m labelType; }
  45
  49
           void destroy();
  50
           void cleanup();
  54
  55
        protected:
  56
  57
            LableType m labelType;
  59
  60 }
 61 #endif //!_LABEL_
```

Layer.cpp File Reference

#include "Layer.h"
#include "Sprite.h"

Namespaces

DsdlEngine

Layer.cpp

```
1 #include "Layer.h"
   2 #include "Sprite.h"
   4 /*
   5
         Base Layer Class
          author: Derek O Brien
   7
         Description: Layer base class for all layers in game.
   8 */
   9
  10 namespace DsdlEngine {
  11
          //Constructor
  12
         Layer::Layer() {
  13
              //Emptty
  14
  15
              layerNodes.reserve(20);
  16
  17
  18
         //Deconstructor
  19
         Layer ::~Layer() {
  20
             destroy();
  21
  22
  23
         //Destroy layer nodes and cleanup
  2.4
         void Layer::destroy() {
             for (size t i = 0; i < layerNodes.size(); i++) {</pre>
                  layerNodes.erase(std::remove(layerNodes.begin(), layerNodes.end(),
layerNodes[i]), layerNodes.end());
  28
                 layerNodes[i]->destroy();
  29
  30
                  //layerNodes.shrink_to_fit();
                  //delete(layerNodes[i]);
  32
  33
             layerNodes.clear();
  34
  35
  36
  37
          //Add Engine node to layer for loading and rendering
         void Layer::addNodeToLayer(EngineBaseNode* node) {
  38
  39
             layerNodes.push_back(node);
  40
  41
  42
  43
         //Remove Node from sceen Vector
  44
          void Layer::removeNodeFromLayer(EngineBaseNode* node) {
  45
             layerNodes.erase(std::remove(layerNodes.begin(), layerNodes.end(), node),
layerNodes.end());
  46
             node->destroy();
  47
  48
  49
         //Load all nodes added to layer
  50
          void Layer::loadNodes(SDL Renderer* r) {
             for (size t i = 0; i < layerNodes.size(); i++) {
  51
  52
                  layerNodes.at(i) ->load(r);
```

```
54
55
       //Render all nodes added to layer
56
       void Layer::drawNodes(SDL Renderer* r) {
57
           for (size t i = 0; i < layerNodes.size(); i++) {</pre>
58
59
60
               if (layerNodes.at(i)->getNodeType() == NodeType::SPRITE) {
61
                   //Reload texture if the texture has been changed
62
63
                   if (layerNodes.at(i) ->isTextureChanged() == true) {
64
                       layerNodes.at(i)->cleanup();
65
                        layerNodes.at(i)->load(r);
66
                       layerNodes.at(i) ->setUpdateTextureTrue(false);
67
68
                   layerNodes.at(i)->render(r);
69
               }
70
71
72
               Reload labels for update each tick for changes to take effect
73
               old label destroyed first to realease its memory so no extra memory been taking up
74
75
               if (layerNodes.at(i)->getNodeType() == NodeType::LABEL) {
76
                   layerNodes.at(i)->cleanup();
77
                   layerNodes.at(i) ->load(r);
78
                   layerNodes.at(i)->render(r);
79
80
           }
81
82
83 }
```

Layer.h File Reference

#include "EngineBaseNode.h"
#include "ResourceTexture.h"

Classes

• class <u>DsdlEngine::Layer</u>

Namespaces

• <u>DsdlEngine</u>

Layer.h

```
1 #ifndef _LAYER_
2 #define _LAYER_
4 #include "EngineBaseNode.h"
5 #include "ResourceTexture.h"
11 namespace DsdlEngine {
12 /***
13 * Base layer class for the engine
15 class Layer {
     public:
16
      //Add Gui Class As friend class
17
18
          friend class Gui;
19
23
          Layer();
24
28
          virtual ~Layer();
29
          virtual void destroy();
33
39
          void loadNodes(SDL Renderer* r);
40
45
           void drawNodes (SDL_Renderer* r);
46
51
           void addNodeToLayer(EngineBaseNode* node);
52
           void removeNodeFromLayer(EngineBaseNode* node);
58
59
           std::vector<EngineBaseNode*> layerNodes;
      private:
60
61
62
       };
63 }
64 #endif //! LAYER
```

Particles.cpp File Reference

#include "Particles.h"

Namespaces

• <u>DsdlEngine</u>

Particles.cpp

```
2 #include "Particles.h"
4 namespace DsdlEngine {
9
10
13
14
    }
15
16
   Particles::~Particles() {}
17
18
19
    bool Particles::isDead(Particles *p)
20
      if ((p->life < 0) || (p->size < 1))
21
22
         return true;
23
      return false;
24
25
26
27 }
```

Particles.h File Reference

#include "DsdlEngine.h"

Classes

• class <u>DsdlEngine</u>::Particles

Namespaces

• <u>DsdlEngine</u>

Macros

• #define MAL PARTICLE LIFE 500

Macro Definition Documentation

#define MAL PARTICLE LIFE 500

Definition at line 7 of file Particles.h.

Particles.h

```
1 #ifndef __Particles_
2 #define __Particles_
3 #include "DsdlEngine.h"
 5 namespace <a href="DsdlEngine">DsdlEngine</a> {
 7 #define MAL_PARTICLE_LIFE 500
 8
10
       class Particles : public EngineBaseNode{
11 public:
           Particles(int x, int y);
12
             ~Particles();
13
14
15
            bool <u>isDead(Particles</u> *p);
16
17
18
            static inline float torad(float angle) {
19
                  return (angle * M_PI) / 180;
20
21
22
      private:
23
24
25
           int life;
26
           float <u>mPosX</u>, <u>mPosY</u>, <u>xvel</u>, <u>yvel</u>, <u>angle</u>, <u>size</u>;
27
             Uint32 endtime;
28
29
30 }
31 #endif // !__Particles_
```

ResourceTexture.cpp File Reference

```
#include "ResourceTexture.h"
#include "EngineError.h"
#include "FileIO.h"
```

Namespaces

DsdlEngine

ResourceTexture.cpp

```
2 /*
 3
       Engine Texture
       Handles all the loading for different textures in the engine
 5
       Handles main call to render texture
 6
       Link between ( Engine / Game ) and SDL Textures
 8 */
9
10
11 #include "ResourceTexture.h"
12 #include "EngineError.h"
13 #include "FileIO.h"
14
15 /*
16
      FIle : ResourcTexture
17
       Author: Derek O Brien
18
       Description: Load & Render Image and TTf media into Sdl Texture
19 */
20
21 namespace DsdlEngine{
22
23
       //Constructor
24
       ResourceTexture::ResourceTexture(){
           m Texture = NULL;
m iHeight = 0;
25
26
          m iWidth = 0;
27
28
      }
29
      //Deconstructor
31
      ResourceTexture::~ResourceTexture(){
32
           destroy();
33
34
35
36
       //Load Sprite from file
37
       bool ResourceTexture::loadTexture(std::string texturePath, SDL Renderer* r){
38
39
           std::string temp = FileIO::getInstance()->getWritablePath() + texturePath;
40
           //Store in map for loading
41
          auto it = m TextureMap.find(temp);
42
           SDL_Texture* newTexture = NULL;
43
44
          SDL Surface* loadedSurface = NULL;
45
46
           if (it == m TextureMap.end()){
47
               //Load image at specified path
48
               loadedSurface = IMG Load(temp.c str());
49
50
51
               if (loadedSurface == NULL)
                   SDL_Log("SDL_image Error : %s ", IMG_GetError());
52
53
               else{
54
                   //Create texture from surface pixels
```

```
55
                       newTexture = SDL CreateTextureFromSurface(r, loadedSurface);
  56
                       if (newTexture == NULL) {
   57
                           SDL Log("SDL CreateTextureFromSurface Error : %s ",IMG GetError());
  58
  59
                       else{
  60
                           //Get image dimensions
  61
                           m iWidth = loadedSurface->w;
                           m iHeight = loadedSurface->h;
  62
  63
  64
                       //Get rid of old loaded surface
  65
                       SDL FreeSurface(loadedSurface);
  66
   67
                    //Add to map
  68
                   m TextureMap[temp] = newTexture;
   69
  70
              else{
   71
                  newTexture = it->second;
  72
  7.3
              //Return success
              m Texture = newTexture;
return m Texture != NULL;
   74
   75
  76
   77
   78
          //Load ttf to sdl texture
  79
  80
          bool ResourceTexture::loadTTF(std::string text, SDL Color color, TTF Font* myfont,
SDL Renderer* r) {
  81
              this->destroy();
  83
               //Create font as surface
  84
              SDL Surface* textSurface = TTF RenderText Blended(myfont, text.c str(), color);
  85
  86
              if (textSurface == NULL) {
  87
                   SDL Log("TTF RenderText Blended Error: %s", TTF GetError());
  88
              else{//COnvert Surface to the Texture
   89
  90
                   m Texture = SDL CreateTextureFromSurface(r, textSurface);
  91
                   if (m Texture == NULL) {
                       SDL Log("TTF RenderText Blended Error : %s" , TTF GetError());
  92
  93
  94
                   else{
                       m iWidth = textSurface->w;
  95
  96
                       m iHeight = textSurface->h;
  97
  98
                   //Free surface as no longer needed
  99
                   SDL FreeSurface(textSurface);
  100
  101
               //Return it
  1.02
              return m Texture != NULL;
  103
          }
  104
  105
          //Basic render
  106
  107
          void ResourceTexture::render(Vec2 p, Vec2 s, SDL Renderer* r, SDL Rect* clip){
               //Set rendering space and render to screen
  108
  109
               SDL Rect renderQuad;
  110
              if (s.\underline{x} != NULL && s.\underline{y} != NULL) {//For Sprites
  111
                   renderQuad = { p.\underline{x}, p.\underline{y}, s.\underline{x}, s.\underline{y} };
  112
  113
               else { //For TTf Labels
  114
                  renderQuad = { p.x_, p.y_, m_iWidth, m_iHeight };
  115
  116
  117
              //Set clip rendering dimensions
  118
              if (clip != NULL) {
                  renderQuad.w = clip->w;
renderQuad.h = clip->h;
  119
  120
  121
  122
  123
               //Render to screen
 124
              SDL RenderCopy(r, m Texture, clip, &renderQuad);
```

```
125
126
127
        //Clean up
128
129
        void ResourceTexture::destroy(){
             if (m Texture != NULL) {
    SDL_DestroyTexture(m Texture);
130
131
                 m Texture = NULL;
m iWidth = 0;
m iHeight = 0;
132
133
134
135
136
        }
137
138
139
            Set Texture Blend Mode
140
        void ResourceTexture::setBlendMode(SDL BlendMode blend) {
141
142
             SDL_SetTextureBlendMode(m Texture, blend);
143
144
145
146
             Set Alpha value of texture for transperence
147
148
             @parma alpha value of texture alpha 0 to 255
149
150
        void ResourceTexture::setAlpha(Uint8 alpha){
151
             SDL SetTextureAlphaMod(m Texture, alpha);
152
153
154 }
```

ResourceTexture.h File Reference

#include "EngineDefines.h"

Classes

• class <u>DsdlEngine::ResourceTexture</u>

Namespaces

• <u>DsdlEngine</u>

ResourceTexture.h

```
1 #ifndef RESOURCETEXTURE
 2 #define RESOURCETEXTURE
4 #include "EngineDefines.h"
9 namespace <a href="DsdlEngine">DsdlEngine</a>{
10
15
      class ResourceTexture{
16
     public:
20
          ResourceTexture();
21
25
          ~ResourceTexture ();
26
33
          bool loadTexture(std::string texturePath, SDL Renderer* r);
34
43
          bool loadTTF(std::string text, SDL Color color, TTF Font* myFont, SDL Renderer* r);
44
52
          void render(Vec2 p, Vec2 s, SDL Renderer* r, SDL Rect* clip = NULL);
53
          void setBlendMode(SDL BlendMode blending);
58
59
64
          void setAlpha(Uint8 alpha);
65
          void destroy();
69
70
     private:
71
72
           SDL Texture* m Texture;
           std::map<std::string, SDL Texture*> m TextureMap;
73
75
           int m_iWidth, m_iHeight;
76
       };
77 }
79 #endif // ! RESOURCETEXTURE
```

Scene.cpp File Reference

#include "IScene.h"

Namespaces

• DsdlEngine

Scene.cpp

```
1 #include "IScene.h"
   2 /*
   3
         Base Scene Class
   4
         author: @Derek O Brien
         Description: Interface for base scene in game.
   5
   6 */
   8
  9 namespace <a href="DsdlEngine">DsdlEngine</a> {
  11
  12
         void IScene::onInput() {
  13
            SDL Event evnt;
             m inputManager.update();
  14
  15
  16
            while (SDL_PollEvent(&evnt)) {
  17
  18
                 if (evnt.key.repeat == 0) {
  19
  20
  21
                      switch (evnt.type) {
                      case SDL QUIT:
  22
  23
                          exit(1);
  24
                         break;
  25
                      case SDL MOUSEMOTION:
  26
                          m inputManager.setMouseCoords(evnt.motion.x, evnt.motion.y);
  27
                         break;
                     case SDL KEYDOWN:
  28
  29
                          m inputManager.pressKey(evnt.key.keysym.sym);
  30
                          break;
  31
                      case SDL KEYUP:
  32
                          m inputManager.releaseKey (evnt.key.keysym.sym);
  33
                      case SDL_MOUSEBUTTONDOWN:
  34
  35
                          m inputManager.pressKey(evnt.button.button);
  36
                          break;
  37
                      case SDL MOUSEBUTTONUP:
                          m inputManager.releaseKey(evnt.button.button);
  39
                          break;
  40
                          //Touch down
  41
                      case SDL FINGERDOWN:
  42
                          m inputManager.pressKey(evnt.button.button);
  43
                          break:
                      case SDL_FINGERMOTION:
  44
                          m inputManager.setMouseCoords((float)evnt.motion.x,
  45
(float)evnt.motion.y);
  46
  47
                      case SDL_FINGERUP:
  48
                          m inputManager.releaseKey(evnt.button.button);
  49
                          break;
  50
                      default:
                          break;
  52
  53
                 }
             }
  54
  55
  56 }
```

SceneManager.cpp File Reference

#include "SceneManager.h"
#include "IScene.h"

Namespaces

DsdlEngine

SceneManager.cpp

```
1 #include "SceneManager.h"
 2 #include "IScene.h"
 7 namespace DsdlEngine{
 8
       //Constructor sets links scene manager and ImainGame
9
10
       SceneManager::SceneManager(IMainGame* game) :
11
           m pGame(game){
12
           //Empty
13
14
15
       //Move to next scene
       IScene* SceneManager::moveNext(){
16
17
           IScene* currentScene = getCurrentScene();
18
           if (currentScene->getNextSceneIndex() != SCENE INDEX NO SCENE) {
19
               m iCurrentSceneIndex = currentScene->getNextSceneIndex();
20
2.1
           return getCurrentScene();
22
23
24
       //Move to Previous Scene
       IScene* SceneManager::movePrevious(){
25
26
           IScene* currentScene = getCurrentScene();
27
           if (currentScene->getPreviousSceneIndex() != SCENE INDEX NO SCENE){
2.8
               m iCurrentSceneIndex = currentScene->getPreviousSceneIndex();
30
           return getCurrentScene();
31
32
33
       //Set Current scene
34
       void SceneManager::setScene(int nextScene) {
3.5
           m iCurrentSceneIndex = nextScene;
36
37
38
       //Add a scene to game
39
       void <u>SceneManager::addScene(IScene</u>* newScene) {
40
           newScene->m iSceneIndex = m pScenes.size();
           m pScenes.push back(newScene);
41
42
           newScene->setParentGame(m pGame);
43
44
45
       //Clean up scenes
       void SceneManager::destroy(){
46
47
           for (size t i = 0; i < m pScenes.size(); i++) {
48
               m pScenes[i]->destroyScene();
49
               //delete m pScenes[i];
50
51
           m pScenes.clear();
52
           m iCurrentSceneIndex = SCENE INDEX NO SCENE;
53
54
55
       //Return the current scene
56
       IScene* SceneManager::getCurrentScene() {
57
           if (m iCurrentSceneIndex == SCENE INDEX NO SCENE) {
58
               return nullptr;
59
```

```
60    return m pScenes[m iCurrentSceneIndex];
61  }
62    63 }
```

SceneManager.h File Reference

#include "EngineDefines.h"

Classes

• class <u>DsdlEngine::SceneManager</u>

Namespaces

• <u>DsdlEngine</u>

SceneManager.h

```
1 #ifndef _SCREENMANAGER_
 2 #define _SCREENMANAGER_
 4 #include "EngineDefines.h"
10 namespace DsdlEngine{
12
      //Forward declearation of classes
13 class IMainGame;
14
     class IScene;
15
19
     class <u>SceneManager</u>{
20 public:
21
26
          SceneManager(IMainGame* game);
27
31
          ~SceneManager() { destroy(); };
32
37
          IScene* moveNext();
38
43
         IScene* movePrevious();
44
49
         void setScene(int nextScene);
50
55
          void addScene(IScene* newScene);
56
60
          void destroy();
61
          IScene* getCurrentScene();
66
67
    protected:
68
69
70
          IMainGame* m pGame;
72
          std::vector<IScene*> m pScenes;
74
          int m_iCurrentSceneIndex;
75
76 }
77 #endif //!_SCREENMANAGER_
```

Sprite.cpp File Reference

#include "Sprite.h"

Namespaces

DsdlEngine

Sprite.cpp

```
1 #include "Sprite.h"
 2
 3 /*
        File: Sprite.h
 4
        Author: Derek O Brien
 6
        Description: Sprite file for creating in game sprites. Inherits from engine base node
 7 */
 8 namespace DsdlEngine{
10
         //Constructor
        Sprite::Sprite(){
11
12
             setEngineNodeType(NodeType::SPRITE);
13
             m frame = 0;
14
            m numFrames = 1;
15
             m opacity = 255;
             m objectBoundingBox = new SDL_Rect();
16
17
             m engineTexture = new ResourceTexture();
        }
18
19
20
        //DeConstructor
21
        Sprite::~Sprite() {
22
             destroy();
2.3
24
25
26
        //Create basic sprite with one frame
27
        void Sprite::create(Vec2 spriteSize, Vec2 position, std::string path){
28
             setAssetPath (path);
29
             m_size.x_ = spriteSize.x_;
m size.y = spriteSize.y;
30
31
32
             \frac{\text{m position.x}}{\text{m position.y}} = \text{position.x};
\frac{\text{m position.y}}{\text{position.y}} = \text{position.y};
33
34
35
             setPosition(position);
36
37
            m numFrames = 1;
             setBoundingBox(position, spriteSize);
39
40
41
        //{\tt Create} basic sprite with more than one frame
42
43
        void Sprite::create(Vec2 spriteSize, Vec2 position, std::string path, int nf){
44
             setAssetPath(path);
45
             \frac{\text{m size.x}}{\text{m size.y}} = \text{spriteSize.x};
\frac{\text{m size.y}}{\text{spriteSize.y}};
46
47
48
49
            m position.x = position.x;
            m position.y = position.y;
50
51
             setPosition (position);
52
53
54
             setBoundingBox(position, spriteSize);
             \underline{m} \ \underline{numFrames} = \underline{nf};
5.5
56
57
```

```
//Create basic sprite with more than one frame and physics body attached
   59
             void <a href="Sprite::createWithPhysics">Sprite::createWithPhysics</a> (b2World* world, <a href="Vec2">Vec2</a> spriteSize, <a href="Vec2">Vec2</a> position, std::string
path, int numFrames, float den, float fri, bool FixedRotation) {
                  setAssetPath(path);
   60
    61
                  \frac{\text{m size.x}}{\text{m size.y}} = \text{spriteSize.x};\frac{\text{m size.y}}{\text{spriteSize.y}};
    62
   63
    64
                   \frac{\text{m position.} \underline{x}}{\text{m position.} \underline{y}} = \text{position.} \underline{x};
    65
    66
    67
    68
                   setPosition (position);
    69
    70
                   setBoundingBox (position, spriteSize);
    71
                   m_numFrames = numFrames;
    72
    73
                   //Create new collision shape
    74
                   m CollisionShape = new CollisionShape();
                   m CollisionShape->init(world, position, spriteSize, den, fri, FixedRotation);
    75
    76
             }
    77
    78
              //Destroy Sprite
    79
             void Sprite::destroy(){
    80
                  EngineBaseNode::destroy();
    81
    82
    83
             //Change sprite texture after sprite is loaded
    84
              void Sprite::updateTexure(Vec2 spriteSize, Vec2 position, std::string path, int numFrames)
    85
    86
                   setUpdateTextureTrue(true);
    87
    88
                  setAssetPath(path);
    89
                   \frac{\text{m size.}\underline{x}}{\text{m size.}\underline{y}} = \text{spriteSize.}\underline{x};
    90
    91
    92
                   \frac{\text{m position.} \underline{x}}{\text{m position.} \underline{y}} = \frac{\text{position.} \underline{x}}{\text{position.} \underline{y}};
    93
    94
    95
                   setPosition(position);
    96
    97
    98
                   setBoundingBox (position, spriteSize);
   99
                   m numFrames = numFrames;
   100
   101
  102 }
```

Sprite.h File Reference

#include "EngineBaseNode.h"

Classes

• class <u>DsdlEngine::Sprite</u>

Namespaces

• <u>DsdlEngine</u>

Sprite.h

```
1 #ifndef _SPRITE_
    2 #define _SPRITE_
    4 #include "EngineBaseNode.h"
   8 namespace DsdlEngine{
          class Sprite : public EngineBaseNode{
   13
   14
           public:
   18
                Sprite();
   19
   23
               virtual ~Sprite();
   24
   31
                void create(Vec2 spriteSize, Vec2 position, std::string path);
                void <u>create(Vec2</u> spriteSize, <u>Vec2</u> position, std::string path, int numFrames);
   40
   41
void <a href="mailto:createWithPhysics">createWithPhysics</a> (b2World* world, <a href="mailto:vec2">Vec2</a> spriteSize, <a href="mailto:vec2">Vec2</a> position, std::string path, int numFrames, float den, float fri, bool FixedRotation);
   54
   62
                void updateTexure(Vec2 spriteSize, Vec2 position, std::string path, int numFrames);
   63
   68
                b2Body* getCollisionBody() { return m CollisionShape->getBody(); }
   69
   73
                void destroy();
   74
   75
            private:
   76
   77
            };
   78 }
   80 #endif //!_SPRITE
```

Timing.cpp File Reference

#include "EngineDefines.h"
#include "Timing.h"

Namespaces

DsdlEngine

Timing.cpp

```
1 #include "EngineDefines.h"
 2 #include "Timing.h"
 4 /*
 5
      File: timimg.cpp
       Author: Derek O Brien
 7 */
8 namespace DsdlEngine{
10
       FpsLimiter::FpsLimiter() {
11
          //Empty
12
      FpsLimiter::~FpsLimiter() {
13
         //Empty
14
15
16
       //Initilaze frame rate
17
18
      void FpsLimiter::init(float maxFPS) {
19
        setMaxFPS(maxFPS);
20
21
22
      //Set Max Frame Rate
23
      void FpsLimiter::setMaxFPS(float maxFPS) {
2.4
          m fMaxFPS = maxFPS;
25
26
27
      //Get start timer
      void FpsLimiter::begin() {
2.8
29
           m iStartTicks = SDL GetTicks();
30
31
32
      //End timer
33
      float FpsLimiter::end() {
34
         calculateFPS();
35
36
         float frameTicks = (float)(SDL GetTicks() - m iStartTicks);
          //Limit the FPS to the max FPS
37
38
           if (1000.0f / m fMaxFPS > frameTicks) {
               SDL Delay((Uint32)(1000.0f / m fMaxFPS - frameTicks));
40
41
          return m fFps;
42
43
44
       void FpsLimiter::calculateFPS() {
45
46
         //The number of frames to average
          static const int NUM SAMPLES = 10;
47
48
          //Stores all the frametimes for each frame that we will average
49
          static float frameTimes[NUM SAMPLES];
50
          //The current frame we are on
          static int currentFrame = 0;
51
52
          //the ticks of the previous frame
53
          static Uint32 prevTicks = SDL GetTicks();
54
55
       //Ticks for the current frame
```

```
56
          Uint32 currentTicks = SDL GetTicks();
57
58
           //Calculate the number of ticks (ms) for this frame
           m fFrameTime = (float) (currentTicks - prevTicks);
59
           frameTimes[currentFrame % NUM SAMPLES] = m fFrameTime;
60
61
62
          //current ticks is now previous ticks
63
          prevTicks = currentTicks;
64
65
           //The number of frames to average
66
          int count;
67
68
          currentFrame++;
           if (currentFrame < NUM SAMPLES) {</pre>
69
70
              count = currentFrame;
71
72
           else {
73
              count = NUM_SAMPLES;
74
75
76
           //{\tt Average} all the frame times
77
           float frameTimeAverage = 0;
78
           for (int i = 0; i < count; i++) {
79
              frameTimeAverage += frameTimes[i];
80
81
          frameTimeAverage /= count;
82
83
           //Calculate FPS
84
           if (frameTimeAverage > 0) {
              m fFps = 1000.0f / frameTimeAverage;
85
86
87
           else {
             m fFps = 120.0f;  //MAX ALLOWED
88
89
90
91 }
```

Timing.h File Reference

Classes

• class <u>DsdlEngine::FpsLimiter</u>

Namespaces

• <u>DsdlEngine</u>

Timing.h

```
1 #ifndef TIMING
2 #define TIMING
12 class <a href="#">FpsLimiter</a> {
13 public:
17
       FpsLimiter();
18
            ~FpsLimiter();
 22
23
28     void init(float maxFPS);
 29
 34
            void setMaxFPS(float maxFPS);
 35
 39
            void begin();
 40
45
            float end();
46
47 private:
51 void
       void <u>calculateFPS();</u>
 52
53
54
55
       float m fFps, m fMaxFPS, m fFrameTime;
unsigned int m iStartTicks;
 56 }
 57
58 #endif
```

Window.cpp File Reference

```
#include "Window.h"
#include "EngineError.h"
```

Namespaces

DsdlEngine

Window.cpp

```
1 #include "Window.h"
    2 #include "EngineError.h"
    3 /*
         File : Window.h
   5
         Author: Derek O Brien
          Description: set up and create Window and render for sdl window
   7 */
   8 namespace DsdlEngine{
  10
          Window::Window() {
  11
            //Empty
  12
  13
          Window::~Window() {
  14
            destroy();
  1.5
  16
  17
  18
         //Create Sdl Window
          int Window::createWindow(std::string windowName, int screenWidth, int screenHeight,
  19
unsigned int flag) {
  20
  21
              m screenHeight = screenHeight;
  2.2
             m screenWidth = screenWidth;
  2.3
  24
              //Screen dimensions
              SDL Rect gScreenRect = { 0, 0, 320, 240 };
  2.5
              SDL DisplayMode displayMode;
  2.6
              if (SDL_GetCurrentDisplayMode(0, &displayMode) == 0)
  2.7
  28
                  gScreenRect.w = displayMode.w;
  29
  30
                  gScreenRect.h = displayMode.h;
  31
  32
   33
              //Load Window for windows using size passed in
  34 #ifdef WIN32
             SDL Log("Windows Created for Windows Platform");
              m pSdlWindow = SDL CreateWindow(windowName.c str(), 0, 0, screenWidth, screenHeight,
  36
flag);
  37
              m pSdlRenderer = SDL CreateRenderer(m pSdlWindow, -1, SDL RENDERER TARGETTEXTURE |
SDL RENDERER ACCELERATED | SDL RENDERER PRESENTVSYNC);
             SDL SetRenderDrawColor(m pSdlRenderer, 0, 0, 0, 120);
  38
  39 #endif
  40
  //Load Window for Android using device screen Size
42 #ifdef __ANDROID_
43 SDL_Log("Windows Created for Android Platform");
              m pSdlWindow = SDL CreateWindow(windowName.c str(), SDL WINDOWPOS UNDEFINED,
SDL WINDOWPOS UNDEFINED, gScreenRect.w, gScreenRect.h, SDL WINDOW ALLOW HIGHDPI);
              m pSdlRenderer = SDL CreateRenderer(m pSdlWindow, -1, SDL RENDERER TARGETTEXTURE |
SDL RENDERER ACCELERATED | SDL RENDERER PRESENTVSYNC);
              SDL_SetRenderDrawColor(m_pSdlRenderer, 0, 0, 0, 120);
  46
  47 #endif
  48
  49
              if (m pSdlWindow == nullptr) {
```

```
51
               SDL_Log("Window could not be created! SDL Error: %s\n", SDL_GetError());
52
               SDL Quit();
           }
53
54
55
56
           //Initialize PNG loading
           int imgFlags = IMG_INIT_PNG;
57
           if (!(IMG Init(imgFlags) & imgFlags)){
58
               SDL_Log("SDL_image could not initialize! SDL_image Error %s\n", IMG_GetError());
59
60
               SDL Quit();
61
62
63
           SDL Log("Window Created");
64
65
           return 0;
66
       }
67
68
       //Swap Window Buffer
69
       void Window::swapBuffer(){
70
           SDL GL SwapWindow(m pSdlWindow);
71
72
73
       //Destroy Window and Renderer
74
       void Window::destroy() {
75
           SDL DestroyRenderer(m pSdlRenderer);
           SDL DestroyWindow(m pSdlWindow);
m pScreenSurface = nullptr;
76
77
78
           SDL Quit();
79
80
81 }
```

Window.h File Reference

#include "EngineDefines.h"

Classes

• class <u>DsdlEngine::Window</u>

Namespaces

• <u>DsdlEngine</u>

Window.h

```
1 #ifndef WINDOW
    2 #define _WINDOW_
   4 #include "EngineDefines.h"
  10 //Wrap Frame Work Code in Namespace
  11 namespace DsdlEngine{
  12
      class Window{
public:
  16
  17
  21
             Window();
  22
  26
             virtual ~Window();
  27
  2.8
  37
              int createWindow(std::string windowNname, int screenWidth, int screenHeight, unsigned
int flag);
  42
              void <u>swapBuffer();</u>
  43
  48
              int getScreenWidth() { return m screenWidth; }
  49
  54
              int getScreenHeight() { return m screenHeight; }
  55
   60
              SDL Renderer* getRenderer() { return m pSdlRenderer; }
  61
  65
              void destroy();
  66 private:
  67
  68
              SDL Window* m pSdlWindow;
              SDL_Renderer* m pSdlRenderer;
  69
              SDL Surface* m pScreenSurface;
int m screenHeight, m screenWidth;
  70
  72
  74
  75 }
  76 #endif //! WINDOW
```

XmlLocalStorage.cpp File Reference

#include "XmlLocalStorage.h"
#include "FileIO.h"

Namespaces

DsdlEngine

Variables

• static XmlLocalStorage * <u>DsdlEngine::Instance</u> = nullptr

XmlLocalStorage.cpp

```
1 #include "XmlLocalStorage.h"
 2 #include "FileIO.h"
 3
 4 /*
       File: XmlLocalStorage
       Author: Derek O Brien
 6
       Description: For loading and saving values to XML file
 8 */
10 using namespace tinyxml2;
11 using namespace std;
13 namespace DsdlEngine {
14
15
       //Create As Singleton static instance
16
       static XmlLocalStorage* Instance = nullptr;
      XmlLocalStorage* XmlLocalStorage::getInstance() {
17
18
          if (!Instance) {
19
               Instance = new (std::nothrow) XmlLocalStorage();
2.0
21
           return Instance;
22
23
24
25
26
       //get integet value for key passed in
27
       int XmlLocalStorage::getIntegerForKey(const char* key) {
29
           const char* value = nullptr;
30
           XMLElement* rootNode;
           XMLDocument* doc;
31
32
          XMLElement* node;
33
34
           //Get node from xml file
35
          node = FileIO::getInstance()->getXMLNodeForKey(key, &rootNode, &doc);
36
37
           //Get the value from the node
38
           if (node && node->FirstChild()) {
39
               value = (const char*) (node->FirstChild()->Value());
40
41
42
           //Convert value to type needed
4.3
           int temp = 0;
44
           if (value) {
45
               temp = SDL atoi(value);
46
47
48
           if (doc) delete doc;
49
50
           return temp;
51
```

```
52
 53
 54
        //Get bool Value for key
 55
        bool XmlLocalStorage::getBoolForKey(const char* key) {
 56
            const char* value = nullptr;
XMLElement* rootNode;
 57
 58
 59
            XMLDocument* doc;
            XMLElement* node;
 60
 61
 62
            //Get node from xml file
 63
            node = FileIO::getInstance()->getXMLNodeForKey(key, &rootNode, &doc);
 64
 65
            //Get the value from the node
            if (node && node->FirstChild()) {
 66
                value = (const char*) (node->FirstChild()->Value());
 67
 68
 69
 70
            //Convert value to type needed
 71
 72
            bool temp = true;
 73
            if (value) {
 74
                temp = (!strcmp(value, "true"));
 75
 76
 77
            if (doc) delete doc;
 78
 79
            return temp;
 80
 81
 82
 83
        //Get Double Value for key passed in
 84
        double XmlLocalStorage::getDoubleForKey(const char* key) {
 85
            const char* value = nullptr;
 86
 87
            XMLElement* rootNode;
            XMLDocument* doc;
 88
 89
            XMLElement* node;
 90
 91
            //Get node from xml file
 92
            node = FileIO::getInstance()->getXMLNodeForKey(key, &rootNode, &doc);
 93
            //Get the value from the node
 95
            if (node && node->FirstChild()) {
 96
                value = (const char*) (node->FirstChild()->Value());
 97
 98
 99
            //Convert value to type needed
100
            double temp = 0.0;
101
102
            if (value) {
103
                temp = SDL atof(value);
104
105
106
            if (doc) delete doc;
107
108
            return temp;
109
        };
110
        //Get float value for key passed in
111
112
        float XmlLocalStorage::getFloatForKey(const char* key) {
113
            float temp = (float)getDoubleForKey(key);
114
            return temp;
115
        };
116
117
        //Get String value for key passed in
118
        std::string XmlLocalStorage::getStringForKey(const char* key) {
119
            const char* value = nullptr;
XMLElement* rootNode;
120
121
          XMLDocument* doc;
122
```

```
123
            XMLElement* node;
124
125
            //Get node from xml file
126
            node = FileIO::getInstance()->getXMLNodeForKey(key, &rootNode, &doc);
127
128
            //Get the value from the node
129
            if (node && node->FirstChild()) {
130
                value = (const char*) (node->FirstChild()->Value());
131
132
133
            //Convert value to type needed
            string temp = "No Value Found";
134
135
136
            if (value) {
                temp = string(value);
137
138
139
140
            return temp;
141
142
143
        //Set a string value for the key
144
        void XmlLocalStorage::setIntegerForKey(int value, const char* key) {
            // check key
145
146
            if (!key) {
147
                return;
148
149
150
            // format the value as char for saving
151
            char tmp[50];
            memset(tmp, 0, 50);
152
153 #ifdef __WIN32
154
            \overline{\text{sprintf_s}}(\text{tmp, "%d", value});
155 #endif
156
157 #ifdef ANDROID
158
            sprintf(tmp, "%d", value);
159 #endif
160
            //Save the Value and key
161
162
            FileIO::getInstance()->setValueForKey(tmp, key);
163
164
165
        //Set bool value for key passed in
        void XmlLocalStorage::setBoolForKey(bool value, const char* key) {
166
167
            if (value == true) {
                setStringForKey("true", key);
168
169
170
            else {
171
                setStringForKey("false", key);
172
173
        }
174
175
        //Set double value for key passed in
176
        void XmlLocalStorage::setDoubleForKey(double value, const char* key) {
177
            // check key
            if (!key) {
178
179
                return;
            }
180
181
182
            // format the value as char for saving
memset(tmp, 0, 50);
185 #ifdef __WIN32
            sprintf s(tmp, "%f", value);
187 #endif
188
189 #ifdef ANDROID
190
            sprintf(tmp, "%f", value);
191 #endif
192
193
        //Save the value and key
```

```
194
            FileIO::getInstance()->setValueForKey(tmp, key);
195
        }
196
197
        //Set float value for key
        void XmlLocalStorage::setFloatForKey(float value, const char* key) {
198
199
            setDoubleForKey(value, key);
200
201
202
        //Set String value for key
203
        void XmlLocalStorage::setStringForKey(std::string value, const char* key) {
204
            if (!key) return;
205
206
            FileIO::getInstance() ->setValueForKey(value.c str(), key);
2.07
208
209
210
        //Delete value fo node
211
        void XmlLocalStorage::deleteValueForKey(const char* key) {
212
213
            XMLElement* rootNode;
            XMLDocument* doc;
214
215
            XMLElement* node;
            XMLPrinter printer;
216
217
            // check the params
218
219
            if (!key) {
220
                return:
221
222
223
            // find the node
224
            node = FileIO::getInstance()->getXMLNodeForKey(key, &rootNode, &doc);
225
226
            // if node not exist, don't need to delete
            if (!node) {
227
228
                return;
229
230
231
            if (doc) {
232
                //Delete Node
233
                doc->DeleteNode(node);
234
                std::string path;
                path = FileIO::getInstance()->getWritablePath() + "Default.xml";
235
236
                // attach printer to the document you want to convert in to a std::string
237
238
                doc->Accept(&printer);
239
240
                // Create a std::string and copy your document data in to the string
241
                const char* buffer = printer.CStr();
242
243
                //Write back to file and save file
244
                if (FileIO::getInstance()->writeDocument(path.c str(), &buffer)) {
245
                    SDL Log("Key : %s :: deleted", key);
2.46
247
                delete doc;
248
            }
249
250 }
```

XmlLocalStorage.h File Reference

```
#include "DsdlEngine.h"
#include "EngineDefines.h"
#include "../dependencies/tinyxml/tinyxml2.h"
```

Classes

• class <u>DsdlEngine::XmlLocalStorage</u>

Namespaces

DsdlEngine

XmlLocalStorage.h

```
1 #ifndef _XMLLOCALSTORAGE_
  2 #define XMLLOCALSTORAGE
 4 #include "DsdlEngine.h"
 5 #include "EngineDefines.h"
 6 #include "../dependencies/tinyxml/tinyxml2.h"
11 namespace <a href="DsdlEngine">DsdlEngine</a> {
12
 17
        class XmlLocalStorage {
18
       public:
19
           static XmlLocalStorage* getInstance();
 24
 25
            void setIntegerForKey(int value, const char* key);
 32
 38
            void setBoolForKey(bool value, const char* key);
 39
 45
            void <u>setDoubleForKey</u>(double value, const char* key);
 46
            void setFloatForKey(float value, const char* key);
 52
 53
 59
            void setStringForKey(std::string value, const char* key);
 60
 61
            int getIntegerForKey(const char* key);
 67
 68
 74
            bool getBoolForKey(const char* key);
 75
 81
            double getDoubleForKey(const char* key);
 88
            float getFloatForKey(const char* key);
 89
 95
            std::string getStringForKey(const char* key);
 96
101
            void deleteValueForKey(const char* key);
102
103
        protected:
104
108
            XmlLocalStorage() {};
109
113
            virtual ~XmlLocalStorage() {};
114
115
        private:
116
117
        };
119 #endif // !_XMLLOCALSTORAGE_
```

Index

_keyMap	addNodeToLayer
DsdlEngine::InputManager, 46	DsdlEngine::Layer, 54
_previousKeyMap	addPreDefineLabel
DsdlEngine::InputManager, 46	DsdlEngine::DsdlGui, 20
~AudioManager	addScene
DsdlEngine::AudioManager, 11	DsdlEngine::SceneManager, 65
~Button	addScenes
DsdlEngine::Button, 14	DsdlEngine::IMainGame, 39
~CollisionShape	angle
DsdlEngine::CollisionShape, 16	DsdlEngine::Particles, 60
~DsdlGui	AudioManager
DsdlEngine::DsdlGui, 19	DsdlEngine::AudioManager, 11
~EngineBaseNode	DsdlEngine::Music, 58
DsdlEngine::EngineBaseNode, 23	DsdlEngine::SFX, 68
~EngineMaster	AudioManager.cpp, 84
DsdlEngine::EngineMaster, 30	AudioManager.h, 86
~FileIO	audioPauseBG
DsdlEngine::FileIO, 31	DsdlEngine::Music, 57
~FpsLimiter	audioResumeBG
DsdlEngine::FpsLimiter, 35	DsdlEngine::Music, 57
~IMainGame	audioStopBG
DsdlEngine::IMainGame, 39	DsdlEngine::Music, 57
~InputManager	BASENODE
DsdlEngine::InputManager, 44	DsdlEngine, 9
~IScene	begin
DsdlEngine::IScene, 49	DsdlEngine::FpsLimiter, 35
~Label	Button
DsdlEngine::Label, 52	DsdlEngine::Button, 13
_	BUTTON
~Layer DsdlEngine::Layer, 54	DsdlEngine, 9
~Particles	_
	Button.cpp, 88
DsdlEngine::Particles, 59 ~ResourceTexture	Button.h, 91 ButtonState
DsdlEngine::ResourceTexture, 62	DsdlEngine, 8
~SceneManager	ButtonType
DsdlEngine::SceneManager, 65	DsdlEngine, 8
~Size	calculateFPS
DsdlEngine::Size, 70	DsdlEngine::FpsLimiter, 36
~Sprite	CallBack
DsdlEngine::Sprite, 72	DsdlEngine, 8
~Vec2	CallBackTimer
DsdlEngine::Vec2, 75	DsdlEngine, 8
~Window	CHANGE_NEXT
DsdlEngine::Window, 77	DsdlEngine, 9
~XmlLocalStorage	CHANGE_PREVIOUS
DsdlEngine::XmlLocalStorage, 80	DsdlEngine, 9
addButton	checkInput
DsdlEngine::DsdlGui, 20	DsdlEngine::Button, 14
addLabel	checkPaused
DsdlEngine::DsdlGui, 20	DsdlEngine::IMainGame, 39
addLayerToScene	cleanup
DsdlEngine::IScene, 49	DsdlEngine::EngineBaseNode, 23

DsdlEngine::Label, 52	CHANGE_PREVIOUS, 9
CollisionShane.cpp, 93	EXIT_APP, 9
CollisionShape	HOVERING, 8
DsdlEngine::CollisionShape, 16	init, 9
CollisionShape.h, 94	Instance, 9, 10
create	LABEL, 9
DsdlEngine::Label, 53	LABEL_BTN, 8
DsdlEngine::Sprite, 72, 73	LABEL_DYNAMIC, 9
createSpriteButton	LABEL_STATIC, 9
DsdlEngine::Button, 14	LableType, 8
createTextButton	make_unique, 9
DsdlEngine::Button, 14	NodeType, 9
createWindow	NONE, 9
DsdlEngine::Window, 77	NORMAL, 8
createWithPhysics	PARTICLE, 9
DsdlEngine::Sprite, 73	PRESSED, 8
createXMLFile	RUNNING, 9
DsdlEngine::FileIO, 32	SceneState, 9
DEBUG_DSDL	SPRITE, 9
EngineError.h, 108	SPRITE_BTN, 8
DEBUG MSG	DsdlEngine.cpp, 95
EngineError.h, 108	DsdlEngine.h, 96
DEFAULT_ROOT_NAME	DsdlEngine::AudioManager, 11
EngineDefines.h, 104	~AudioManager, 11
DEGTORAD	AudioManager, 11
EngineDefines.h, 104	destroy, 11
deleteValueForKey	init, 12
DsdlEngine::XmlLocalStorage, 81	loadMusic, 12
destroy	loadSFX, 12
DsdlEngine::AudioManager, 11	m_bgAudioMap, 12
DsdlEngine::Button, 14	m_bisInitialized, 12
DsdlEngine::CollisionShape, 16	m_sfxAudioMap, 12
DsdlEngine::DsdlGui, 20	DsdlEngine::Button, 13
DsdlEngine::EngineBaseNode, 23	~Button, 14
DsdlEngine::Label, 53	Button, 13
DsdlEngine::Layer, 55	checkInput, 14
DsdlEngine::ResourceTexture, 62	createSpriteButton, 14
DsdlEngine::SceneManager, 66	createTextButton, 14
DsdlEngine::Sprite, 73	destroy, 14
DsdlEngine::Window, 78	getButtonName, 14
destroyScene	m_buttonName, 15
DsdlEngine::IScene, 49	m_eCurrentState, 15
draw	m_label, 15
DsdlEngine::IMainGame, 39	m_spriteBtn, 15
drawNodes	onClicked, 15
DsdlEngine::Layer, 55	onMouseEnters, 15
drawScene	onMouseLeaves, 15
DsdlEngine::IScene, 49	DsdlEngine::CollisionShape, 16
DsdlEngine, 7	~CollisionShape, 16
BASENODE, 9	CollisionShape, 16
BUTTON, 9	destroy, 16
ButtonState, 8	getBody, 17
ButtonType, 8	getDimensions, 17
CallBack, 8	getFixture, 17
CallBackTimer, 8	init, 17
CHANGE_NEXT, 9	m_body, 17

m_dimensions, 17	setPositionX, 26
m_fixtures, 17	setPositionY, 26
DsdlEngine::DsdlGui, 19	setSize, 26
~DsdlGui, 19	setUpdateTextureTrue, 27
addButton, 20	setWidth, 27
addLabel, 20	updateLabelText, 27
addPreDefineLabel, 20	updateTextureInfo, 29
destroy, 20	DsdlEngine::EngineMaster, 30
DsdlGui, 19	~EngineMaster, 30
getButton, 20	EngineMaster, 30
GUIElements, 21	getInstance, 30
m_btn, 21	DsdlEngine::FileIO, 31
m_label, 21	~FileIO, 31
onSDLEvent, 21	createXMLFile, 32
setGUIPos, 21	FileIO, 31
DsdlEngine::EngineBaseNode, 22	getFileToOpen, 32
~EngineBaseNode, 23	getInstance, 32
cleanup, 23	getSuitableFOpen, 32
destroy, 23	getWritablePath, 32
EngineBaseNode, 23	getXMLNodeForKey, 32
getAssetsPath, 23	loadDocument, 33
getBoundingBox, 24	m_fileName, 33
getContentSize, 24	m_path, 34
getEngineTexture, 24	setAssetsPath, 33
getNodeType, 24	setFileToOpen, 33
getPosition, 24	setValueForKey, 33
isTextureChanged, 24	writeDocument, 33
load, 24	DsdlEngine::FpsLimiter, 35
m_assetPath, 27	~FpsLimiter, 35
m_CollisionShape, 27	begin, 35
m_currentFrame, 27	calculateFPS, 36
m_engineTexture, 27	end, 36
m_font, 27	FpsLimiter, 35
m_FontMap, 27	init, 36
m_frame, 28	m_fFps, 36
m_gSpriteClips, 28	m_fFrameTime, 36
m_labelText, 28	m_fMaxFPS, 36
m_numFrames, 28	m_iStartTicks, 36
m_objectBoundingBox, 28	setMaxFPS, 36
m_opacity, 28	DsdlEngine::IMainGame, 38
m_position, 28	~IMainGame, 39
m_size, 28	addScenes, 39
m_textColor, 28	checkPaused, 39
m_textSize, 28	draw, 39
nodeType, 29	exitGame, 39
render, 25	getFps, 39
renderCollisionShape, 25	IMainGame, 39
scaleHeight, 25	init, 40
scaleNode, 25	initSystems, 40
scaleWidth, 25	m_audioManager, 41
setAssetPath, 25	m_bIsPaused, 41
setBoundingBox, 25	m_blsRunning, 41
setBoundingBox, 23 setEngineNodeType, 26	m_fFps, 41
setHeight, 26	m_InputManager, 41
setOpacity, 26	m_pCurrentRunning, 41
setPosition, 26	m_pGameRenderer, 42

m_pSceneManager, 42	m_inputManager, 51
m_Window, 42	m_iSceneIndex, 51
m_windowHeight, 42	onEntryScene, 50
m_windowWidth, 42	onExitScene, 50
mainAssetsPath, 42	onInput, 50
mainLoop, 40	sceneLayers, 51
onExit, 40	SceneManager, 50
onInit, 40	setParentGame, 50
onSDLEvent, 40	setSceneRunning, 50
run, 40	updateScene, 50
setFps, 40	DsdlEngine::Label, 52
set Paused, 40	~Label, 52
setRunning, 41	cleanup, 52
setupWindow, 41	create, 53
update, 41	destroy, 53
windowFlag, 42	getType, 53
<u> </u>	
windowtitle, 42	Label, 52
DsdlEngine::InputManager, 43	m_labelType, 53
_keyMap, 46	setType, 53
_previousKeyMap, 46	DsdlEngine::Layer, 54
~InputManager, 44	~Layer, 54
fingerDown, 46	addNodeToLayer, 54
fingerUp, 46	destroy, 55
InputManager, 43	drawNodes, 55
isKeyDown, 44	Gui, 55
isKeyPressed, 44	Layer, 54
isKeyReleased, 44	layerNodes, 55
isSwipe, 44	loadNodes, 55
isSwipeDown, 44	removeNodeFromLayer, 55
isSwipeLeft, 45	DsdlEngine::Music, 57
isSwipeRight, 45	AudioManager, 58
isSwipeUp, 45	audioPauseBG, 57
isTouch, 45	audioResumeBG, 57
pressKey, 45	audioStopBG, 57
releaseKey, 45	m_Music, 58
setMouseCoords, 45	play, 57
swipedown, 46	DsdlEngine::Particles, 59
swipeleft, 46	~Particles, 59
swiperight, 47	angle, 60
swipeup, 47	endtime, 60
update, 46	isDead, 60
wasKeyDown, 46	life, 60
DsdlEngine::IScene, 48	mPosX, 60
~IScene, 49	mPosY, 60
addLayerToScene, 49	Particles, 59
destroyScene, 49	size, 60
drawScene, 49	torad, 60
getNextSceneIndex, 49	xvel, 60
getPreviousSceneIndex, 49	yvel, 60
getSceneIndex, 49	DsdlEngine::ResourceTexture, 62
getSceneState, 49	~ResourceTexture, 62
InputManager, 50	
	destroy, 62
IScene, 48	loadTexture, 63
loadScene, 50	loadTTF, 63
m_eCurrentState, 51	m_iHeight, 64
m. game, 51	m iWidth, 64

m_Texture, 64	~XmlLocalStorage, 80
m_TextureMap, 64	deleteValueForKey, 81
render, 63	getBoolForKey, 81
ResourceTexture, 62	getDoubleForKey, 81
setAlpha, 63	getFloatForKey, 81
setBlendMode, 63	getInstance, 81
DsdlEngine::SceneManager, 65	getIntegerForKey, 81
~SceneManager, 65	getStringForKey, 82
addScene, 65	setBoolForKey, 82
destroy, 66	setDoubleForKey, 82
getCurrentScene, 66	setFloatForKey, 82
m_iCurrentSceneIndex, 66	setIntegerForKey, 82
m_pGame, 66	setStringForKey, 82
m_pScenes, 66	XmlLocalStorage, 80
moveNext, 66	DsdlGui
movePrevious, 66	DsdlEngine::DsdlGui, 19
SceneManager, 65	end
setScene, 66	DsdlEngine::FpsLimiter, 36
DsdlEngine::SFX, 68	endtime
AudioManager, 68	DsdlEngine::Particles, 60
m_Chunk, 68	EngineBaseNode
play, 68	DsdlEngine::EngineBaseNode, 23
DsdlEngine::Size, 70	EngineBaseNode.cpp, 98
~Size, 70	EngineBaseNode.h, 101
h_, 71	EngineDefines.h, 103, 105
Size, 70	DEFAULT_ROOT_NAME, 104
W_, 71	DEGTORAD, 104
DsdlEngine::Sprite, 72	GRAVITYSCALE, 104
~Sprite, 72	METRESTOPIXELS, 104
create, 72, 73	NS_DSDL_END, 104
createWithPhysics, 73	NS_DSDL_START, 104
destroy, 73	PIXELSTOMETRES, 104
getCollisionBody, 73	RADTODEG, 104
Sprite, 72	TOTAL_PARTICLES, 104
updateTexure, 73	USING_NS_DSDL, 104
DsdlEngine::Vec2, 75	XML_FILE, 105
~Vec2, 75	EngineError.cpp, 107
Vec2, 75	EngineError.h, 108
x_, 76	DEBUG_DSDL, 108
y_, 76	DEBUG_MSG, 108
ZERO, 76	EngineMaster
DsdlEngine::Window, 77	DsdlEngine::EngineMaster, 30
~Window, 77	EngineMaster.cpp, 109
createWindow, 77	EngineMaster.h, 110
destroy, 78	EngineMath.cpp, 111
getRenderer, 78	EngineMath.h, 112
getScreenHeight, 78	EXIT_APP
getScreenWidth, 78	DsdlEngine, 9
m_pScreenSurface, 78	exitGame
m_pSdlRenderer, 78	DsdlEngine::IMainGame, 39
m_pSdlWindow, 79	FileIO
m_screenHeight, 79	DsdlEngine::FileIO, 31
m_screenWidth, 79	FileIO.cpp, 113
swapBuffer, 78	FileIO.h, 117
Window, 77	fingerDown
DsdlEngine::XmlLocalStorage, 80	DsdlEngine::InputManager, 46

fingerUp	getScreenHeight
DsdlEngine::InputManager, 46	DsdlEngine::Window, 78
FpsLimiter	getScreenWidth
DsdlEngine::FpsLimiter, 35	DsdlEngine::Window, 78
getAssetsPath	getStringForKey
DsdlEngine::EngineBaseNode, 23	DsdlEngine::XmlLocalStorage, 82
getBody	getSuitableFOpen
DsdlEngine::CollisionShape, 17	DsdlEngine::FileIO, 32
getBoolForKey	getType
DsdlEngine::XmlLocalStorage, 81	DsdlEngine::Label, 53
getBoundingBox	getWritablePath
DsdlEngine::EngineBaseNode, 24	DsdlEngine::FileIO, 32
getButton	getXMLNodeForKey
DsdlEngine::DsdlGui, 20	DsdlEngine::FileIO, 32
getButtonName	GRAVITYSCALE
DsdlEngine::Button, 14	EngineDefines.h, 104
getCollisionBody	Gui
DsdlEngine::Sprite, 73	DsdlEngine::Layer, 55
getContentSize	Gui.cpp, 119
DsdlEngine::EngineBaseNode, 24	Gui.h, 121
getCurrentScene	GUIElements
DsdlEngine::SceneManager, 66	DsdlEngine::DsdlGui, 21
getDimensions	h
DsdlEngine::CollisionShape, 17	DsdlEngine::Size, 71
getDoubleForKey	HOVERING
DsdlEngine::XmlLocalStorage, 81	DsdlEngine, 8
getEngineTexture	IMainGame
DsdlEngine::EngineBaseNode, 24	DsdlEngine::IMainGame, 39
getFileToOpen	IMainGame.cpp, 122
DsdlEngine::FileIO, 32	IMainGame.h, 126
getFixture	init
DsdlEngine::CollisionShape, 17	DsdlEngine, 9
getFloatForKey	DsdlEngine::AudioManager, 12
DsdlEngine::XmlLocalStorage, 81	DsdlEngine::CollisionShape, 17
getFps	DsdlEngine::FpsLimiter, 36
DsdlEngine::IMainGame, 39	DsdlEngine::IMainGame, 40
getInstance	initSystems
DsdlEngine::EngineMaster, 30	DsdlEngine::IMainGame, 40
DsdlEngine::FileIO, 32	InputManager
DsdlEngine::XmlLocalStorage, 81	DsdlEngine::InputManager, 43
getIntegerForKey	DsdlEngine::IScene, 50
DsdlEngine::XmlLocalStorage, 81	InputManager.cpp, 128
getNextSceneIndex	InputManager.h, 131
DsdlEngine::IScene, 49	Instance
getNodeType	DsdlEngine, 9, 10
DsdlEngine::EngineBaseNode, 24	IScene
getPosition	DsdlEngine::IScene, 48
DsdlEngine::EngineBaseNode, 24	IScene.h, 132
getPreviousSceneIndex	SCENE_INDEX_NO_SCENE, 132
DsdlEngine::IScene, 49	isDead
getRenderer	DsdlEngine::Particles, 60
DsdlEngine::Window, 78	isKeyDown
getSceneIndex	DsdlEngine::InputManager, 44
DsdlEngine::IScene, 49	isKeyPressed
getSceneState	DsdlEngine::InputManager, 44
DsdlEngine::IScene, 49	isKeyReleased
	10110 / 110100000

DsdlEngine::IMainGame, 41 DsdlEngine::InputManager, 44 isSwipe m bgAudioMap DsdlEngine::InputManager, 44 DsdlEngine::AudioManager, 12 isSwipeDown m bisInitialized DsdlEngine::InputManager, 44 DsdlEngine::AudioManager, 12 m bIsPaused isSwipeLeft DsdlEngine::InputManager, 45 DsdlEngine::IMainGame, 41 isSwipeRight m bIsRunning DsdlEngine::InputManager, 45 DsdlEngine::IMainGame, 41 isSwipeUp m body DsdlEngine::InputManager, 45 DsdlEngine::CollisionShape, 17 isTextureChanged m_btn DsdlEngine::EngineBaseNode, 24 DsdlEngine::DsdlGui, 21 m_buttonName DsdlEngine::InputManager, 45 DsdlEngine::Button, 15 Label m Chunk DsdlEngine::Label, 52 DsdlEngine::SFX, 68 **LABEL** m_CollisionShape DsdlEngine, 9 DsdlEngine::EngineBaseNode, 27 Label.cpp, 134 m_currentFrame Label.h, 135 DsdlEngine::EngineBaseNode, 27 LABEL BTN m dimensions DsdlEngine::CollisionShape, 17 DsdlEngine, 8 LABEL_DYNAMIC m_eCurrentState DsdlEngine, 9 DsdlEngine::Button, 15 LABEL_STATIC DsdlEngine::IScene, 51 DsdlEngine, 9 m_engineTexture LableType DsdlEngine::EngineBaseNode, 27 DsdlEngine, 8 m_fFps Layer DsdlEngine::FpsLimiter, 36 DsdlEngine::Layer, 54 DsdlEngine::IMainGame, 41 m_fFrameTime Layer.cpp, 136 Layer.h, 138 DsdlEngine::FpsLimiter, 36 layerNodes m_fileName DsdlEngine::Layer, 55 DsdlEngine::FileIO, 33 life m fixtures DsdlEngine::CollisionShape, 17 DsdlEngine::Particles, 60 load m fMaxFPS DsdlEngine::EngineBaseNode, 24 DsdlEngine::FpsLimiter, 36 m_font loadDocument DsdlEngine::FileIO, 33 DsdlEngine::EngineBaseNode, 27 loadMusic m_FontMap DsdlEngine::EngineBaseNode, 27 DsdlEngine::AudioManager, 12 loadNodes m frame DsdlEngine::Layer, 55 DsdlEngine::EngineBaseNode, 28 loadScene m game DsdlEngine::IScene, 50 DsdlEngine::IScene, 51 m_gSpriteClips loadSFX DsdlEngine::AudioManager, 12 DsdlEngine::EngineBaseNode, 28 loadTexture m iCurrentSceneIndex DsdlEngine::ResourceTexture, 63 DsdlEngine::SceneManager, 66 loadTTF m iHeight DsdlEngine::ResourceTexture, 63 DsdlEngine::ResourceTexture, 64 m assetPath m_inputManager DsdlEngine::IScene, 51 DsdlEngine::EngineBaseNode, 27 m_audioManager m_InputManager

DsdlEngine::IMainGame, 41 m_Texture m iSceneIndex DsdlEngine::ResourceTexture, 64 DsdlEngine::IScene, 51 m TextureMap m iStartTicks DsdlEngine::ResourceTexture, 64 DsdlEngine::FpsLimiter, 36 m Window DsdlEngine::IMainGame, 42 m iWidth DsdlEngine::ResourceTexture, 64 m windowHeight m label DsdlEngine::IMainGame, 42 DsdlEngine::Button, 15 m_windowWidth DsdlEngine::DsdlGui, 21 DsdlEngine::IMainGame, 42 m labelText mainAssetsPath DsdlEngine::EngineBaseNode, 28 DsdlEngine::IMainGame, 42 m labelType mainLoop DsdlEngine::Label, 53 DsdlEngine::IMainGame, 40 m_Music make_unique DsdlEngine::Music, 58 DsdlEngine, 9 m numFrames MAL PARTICLE LIFE DsdlEngine::EngineBaseNode, 28 Particles.h, 140 m objectBoundingBox **METRESTOPIXELS** DsdlEngine::EngineBaseNode, 28 EngineDefines.h, 104 moveNext m opacity DsdlEngine::EngineBaseNode, 28 DsdlEngine::SceneManager, 66 movePrevious m_path DsdlEngine::FileIO, 34 DsdlEngine::SceneManager, 66 m pCurrentRunning mPosX DsdlEngine::IMainGame, 41 DsdlEngine::Particles, 60 m pGame mPosY DsdlEngine::SceneManager, 66 DsdlEngine::Particles, 60 m_pGameRenderer nodeType DsdlEngine::IMainGame, 42 DsdlEngine::EngineBaseNode, 29 m_position NodeType DsdlEngine, 9 DsdlEngine::EngineBaseNode, 28 m_pSceneManager **NONE** DsdlEngine::IMainGame, 42 DsdlEngine, 9 m pScenes **NORMAL** DsdlEngine::SceneManager, 66 DsdlEngine, 8 m pScreenSurface NS DSDL END DsdlEngine::Window, 78 EngineDefines.h, 104 m pSdlRenderer NS DSDL START DsdlEngine::Window, 78 EngineDefines.h, 104 m_pSdlWindow onClicked DsdlEngine::Window, 79 DsdlEngine::Button, 15 m_screenHeight onEntryScene DsdlEngine::Window, 79 DsdlEngine::IScene, 50 m_screenWidth DsdlEngine::Window, 79 DsdlEngine::IMainGame, 40 m sfxAudioMap onExitScene DsdlEngine::AudioManager, 12 DsdlEngine::IScene, 50 m size onInit DsdlEngine::EngineBaseNode, 28 DsdlEngine::IMainGame, 40 m spriteBtn onInput DsdlEngine::Button, 15 DsdlEngine::IScene, 50 m textColor onMouseEnters DsdlEngine::EngineBaseNode, 28 DsdlEngine::Button, 15 m textSize onMouseLeaves DsdlEngine::EngineBaseNode, 28 DsdlEngine::Button, 15

onSDLEvent setAlpha DsdlEngine::DsdlGui, 21 DsdlEngine::ResourceTexture, 63 setAssetPath DsdlEngine::IMainGame, 40 **PARTICLE** DsdlEngine::EngineBaseNode, 25 DsdlEngine, 9 setAssetsPath DsdlEngine::FileIO, 33 **Particles** DsdlEngine::Particles, 59 setBlendMode Particles.cpp, 139 DsdlEngine::ResourceTexture, 63 Particles.h, 140 setBoolForKey MAL_PARTICLE_LIFE, 140 DsdlEngine::XmlLocalStorage, 82 **PIXELSTOMETRES** setBoundingBox EngineDefines.h, 104 DsdlEngine::EngineBaseNode, 25 setDoubleForKey play DsdlEngine::Music, 57 DsdlEngine::XmlLocalStorage, 82 DsdlEngine::SFX, 68 setEngineNodeType **PRESSED** DsdlEngine::EngineBaseNode, 26 DsdlEngine, 8 setFileToOpen pressKey DsdlEngine::FileIO, 33 DsdlEngine::InputManager, 45 setFloatForKey **RADTODEG** DsdlEngine::XmlLocalStorage, 82 EngineDefines.h, 104 releaseKey DsdlEngine::IMainGame, 40 DsdlEngine::InputManager, 45 setGUIPos removeNodeFromLayer DsdlEngine::DsdlGui, 21 DsdlEngine::Layer, 55 setHeight render DsdlEngine::EngineBaseNode, 26 DsdlEngine::EngineBaseNode, 25 setIntegerForKey DsdlEngine::ResourceTexture, 63 DsdlEngine::XmlLocalStorage, 82 renderCollisionShape setMaxFPS DsdlEngine::EngineBaseNode, 25 DsdlEngine::FpsLimiter, 36 ResourceTexture setMouseCoords DsdlEngine::ResourceTexture, 62 DsdlEngine::InputManager, 45 ResourceTexture.cpp, 141 setOpacity ResourceTexture.h, 144 DsdlEngine::EngineBaseNode, 26 setParentGame run DsdlEngine::IScene, 50 DsdlEngine::IMainGame, 40 **RUNNING** setPaused DsdlEngine::IMainGame, 40 DsdlEngine, 9 scaleHeight setPosition DsdlEngine::EngineBaseNode, 25 DsdlEngine::EngineBaseNode, 26 setPositionX scaleNode DsdlEngine::EngineBaseNode, 25 DsdlEngine::EngineBaseNode, 26 scaleWidth setPositionY DsdlEngine::EngineBaseNode, 25 DsdlEngine::EngineBaseNode, 26 Scene.cpp, 145 setRunning SCENE INDEX NO SCENE DsdlEngine::IMainGame, 41 IScene.h, 132 setScene sceneLavers DsdlEngine::SceneManager, 66 DsdlEngine::IScene, 51 setSceneRunning SceneManager DsdlEngine::IScene, 50 DsdlEngine::IScene, 50 setSize DsdlEngine::SceneManager, 65 DsdlEngine::EngineBaseNode, 26 SceneManager.cpp, 147 setStringForKey SceneManager.h, 149 DsdlEngine::XmlLocalStorage, 82 SceneState setType DsdlEngine, 9 DsdlEngine::Label, 53

set Up date Texture TrueupdateScene DsdlEngine::EngineBaseNode, 27 DsdlEngine::IScene, 50 setupWindow updateTextureInfo DsdlEngine::IMainGame, 41 DsdlEngine::EngineBaseNode, 29 setValueForKey updateTexure DsdlEngine::FileIO, 33 DsdlEngine::Sprite, 73 USING_NS_DSDL setWidth DsdlEngine::EngineBaseNode, 27 EngineDefines.h, 104 size Vec2 DsdlEngine::Particles, 60 DsdlEngine::Vec2, 75 DsdlEngine::Size, 71 DsdlEngine::Size, 70 wasKeyDown Sprite DsdlEngine::Sprite, 72 DsdlEngine::InputManager, 46 **SPRITE** Window DsdlEngine, 9 DsdlEngine::Window, 77 Sprite.cpp, 150 Window.cpp, 156 Sprite.h, 152 Window.h, 158 SPRITE BTN windowFlag DsdlEngine, 8 DsdlEngine::IMainGame, 42 swapBuffer windowtitle DsdlEngine::Window, 78 DsdlEngine::IMainGame, 42 swipedown writeDocument DsdlEngine::FileIO, 33 DsdlEngine::InputManager, 46 swipeleft DsdlEngine::Vec2, 76 DsdlEngine::InputManager, 46 XML_FILE swiperight DsdlEngine::InputManager, 47 EngineDefines.h, 105 XmlLocalStorage swipeup DsdlEngine::InputManager, 47 DsdlEngine::XmlLocalStorage, 80 Timing.cpp, 153 XmlLocalStorage.cpp, 159 Timing.h, 155 XmlLocalStorage.h, 163 torad xvel DsdlEngine::Particles, 60 DsdlEngine::Particles, 60 TOTAL PARTICLES EngineDefines.h, 104 DsdlEngine::Vec2, 76 yvel update DsdlEngine::IMainGame, 41 DsdlEngine::Particles, 60 DsdlEngine::InputManager, 46 **ZERO** updateLabelText DsdlEngine::Vec2, 76

DsdlEngine::EngineBaseNode, 27