

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
DsdEngine	7
Class Documentation	11
DsdEngine::AudioManager	11
DsdEngine::Button	13
DsdEngine::CollisionShape	16
DsdEngine::DsdGui	19
DsdEngine::EngineBaseNode	22
DsdEngine::EngineMaster	30
DsdEngine::FileIO	31
DsdEngine::FpsLimiter	35
DsdEngine::IMainGame	38
DsdEngine::InputManager	43
DsdEngine::IScene	48
DsdEngine::Label	52
DsdEngine::Layer	54
DsdEngine::Music	57
DsdEngine::Particles	59
DsdEngine::ResourceTexture	62
DsdEngine::SceneManager	65
DsdEngine::SFX	68
DsdEngine::Size	70
DsdEngine::Sprite	72
DsdEngine::Vec2	75
DsdEngine::Window	77
DsdEngine::XmlLocalStorage	80
File Documentation	84
AudioManager.cpp	84
AudioManager.cpp	84
AudioManager.h	86
AudioManager.h	86
Button.cpp	88
Button.cpp	88
Button.h	91
Button.h	91
CollisionShane.cpp	93
CollisionShane.cpp	93
CollisionShape.h	94
CollisionShape.h	94
DsdEngine.cpp	95
DsdEngine.cpp	95
DsdEngine.h	96
DsdEngine.h	96
EngineBaseNode.cpp	98
EngineBaseNode.cpp	98
EngineBaseNode.h	101
EngineBaseNode.h	101
EngineDefines.h	103
EngineDefines.h	105

EngineError.cpp.....	107
EngineError.cpp.....	107
EngineError.h	108
EngineError.h	108
EngineMaster.cpp	109
EngineMaster.cpp	109
EngineMaster.h.....	110
EngineMaster.h.....	110
EngineMath.cpp.....	111
EngineMath.cpp.....	111
EngineMath.h	112
EngineMath.h	112
FileIO.cpp	113
FileIO.cpp	113
FileIO.h.....	117
FileIO.h.....	117
Gui.cpp	119
Gui.cpp	119
Gui.h	121
Gui.h	121
IMainGame.cpp	122
IMainGame.cpp	122
IMainGame.h.....	126
IMainGame.h.....	126
InputManager.cpp.....	128
InputManager.cpp.....	128
InputManager.h.....	131
InputManager.h.....	131
IScene.h	132
IScene.h	132
Label.cpp	134
Label.cpp	134
Label.h	135
Label.h	135
Layer.cpp	136
Layer.cpp	136
Layer.h.....	138
Layer.h.....	138
Particles.cpp.....	139
Particles.cpp.....	139
Particles.h	140
Particles.h	140
ResourceTexture.cpp	141
ResourceTexture.cpp	141
ResourceTexture.h	144
ResourceTexture.h	144
Scene.cpp.....	145
Scene.cpp.....	145
SceneManager.cpp.....	147
SceneManager.cpp.....	147
SceneManager.h	149
SceneManager.h	149
Sprite.cpp.....	150
Sprite.cpp.....	150
Sprite.h.....	152
Sprite.h.....	152

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

Namespace Index

Namespace List

Here is a list of all namespaces with brief descriptions:

DsdEngine	7
---------------------------	---

Hierarchical Index

Class Hierarchy

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

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

Class Index

Class List

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

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

File Index

File List

Here is a list of all files with brief descriptions:

AudioManager.cpp	84
AudioManager.h	86
Button.cpp	88
Button.h	91
CollisionShane.cpp	93
CollisionShape.h	94
DsdEngine.cpp	95
DsdEngine.h	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
FileIO.h	117
Gui.cpp	119
Gui.h	121
IMainGame.cpp	122
IMainGame.h	126
InputManager.cpp	128
InputManager.h	131
IScene.h	132
Label.cpp	134
Label.h	135
Layer.cpp	136
Layer.h	138
Particles.cpp	139
Particles.h	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

<u>Timing.h</u>	155
<u>Window.cpp</u>	156
<u>Window.h</u>	158
<u>XmlLocalStorage.cpp</u>	159
<u>XmlLocalStorage.h</u>	163

Namespace Documentation

DsdEngine Namespace Reference

Classes

- class [AudioManager](#)
- class [Button](#)
- class [CollisionShape](#)
- class [DsdGui](#)
- class [EngineBaseNode](#)
- class [EngineMaster](#)
- class [FileIO](#)
- class [FpsLimiter](#)
- class [MainGame](#)
- class [InputManager](#)
- class [IScene](#)
- class [Label](#)
- class [Layer](#)
- class [Music](#)
- class [Particles](#)
- class [ResourceTexture](#)
- class [SceneManager](#)
- class [SFX](#)
- class [Size](#)
- class [Sprite](#)
- class [Vec2](#)
- class [Window](#)
- class [XmlLocalStorage](#)

Typedefs

- typedef SDL_TimerID [CallBackTimer](#)
- typedef SDL_TimerCallback [CallBack](#)

Enumerations

- enum [NodeType](#) { [NodeType::BASENODE](#), [NodeType::SPRITE](#), [NodeType::LABEL](#), [NodeType::BUTTON](#), [NodeType::PARTICLE](#) }
- enum [ButtonState](#) { [ButtonState::NORMAL](#), [ButtonState::PRESSED](#), [ButtonState::HOVERING](#) }
- enum [ButtonType](#) { [ButtonType::LABEL_BTN](#), [ButtonType::SPRITE_BTN](#) }
- enum [LableType](#) { [LableType::LABEL_STATIC](#), [LableType::LABEL_DYNAMIC](#) }
- enum [SceneState](#) { [SceneState::NONE](#), [SceneState::RUNNING](#), [SceneState::EXIT_APP](#), [SceneState::CHANGE_NEXT](#), [SceneState::CHANGE_PREVIOUS](#) }

Functions

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

Variables

- static [EngineMaster](#) * [Instance](#) = nullptr

- static [FileIO](#) * [Instance](#) = nullptr
 - static [XmlLocalStorage](#) * [Instance](#) = nullptr
-

Detailed Description

Author:

Derek O Brien
Derek O Brien

Class Based of
<https://www.youtube.com/watch?v=Epyih-LEbig&list=PLSPw4ASQYyymu3PfG9gxywSPghnSMiOAW&index=26> tutorial

Typedef Documentation

typedef SDL_TimerCallback [DsdEngine::Callback](#)

Definition at line [75](#) of file [EngineDefines.h](#).

typedef SDL_TimerID [DsdEngine::CallbackTimer](#)

Timer Call Back

Definition at line [74](#) of file [EngineDefines.h](#).

Enumeration Type Documentation

enum [DsdEngine::ButtonState](#) [strong]

[Button](#) State Enum

Enumerator

NORMAL
PRESSED
HOVERING

Definition at line [49](#) of file [EngineDefines.h](#).

enum [DsdEngine::ButtonType](#) [strong]

[Button](#) Type Enum

Enumerator

LABEL_BTN
SPRITE_BTN

Definition at line [58](#) of file [EngineDefines.h](#).

enum [DsdEngine::LableType](#) [strong]

[Label](#) Type Enum

Enumerator

LABEL_STATIC
LABEL_DYNAMIC

Definition at line [66](#) of file [EngineDefines.h](#).

enum [DsdIEngine::NodeType](#) [**strong**]

Node Type Enum

Enumerator

BASENODE
SPRITE
LABEL
BUTTON
PARTICLE

Definition at line [38](#) of file [EngineDefines.h](#).

enum [DsdIEngine::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 [DsdIEngine::init](#) ()

init, Initalize SDL

Definition at line [7](#) of file [DsdIEngine.cpp](#).

template<typename T , typename... Args> std::unique_ptr<T> [DsdIEngine::make_unique](#) (Args &&... args)

Definition at line [17](#) of file [IMainGame.cpp](#).

Variable Documentation

[EngineMaster](#)* [DsdIEngine::Instance](#) = nullptr [**static**]

Definition at line [6](#) of file [EngineMaster.cpp](#).

[FileIO](#)* DsdEngine::Instance = nullptr[static]

Definition at line [11](#) of file [FileIO.cpp](#).

[XmlLocalStorage](#)* DsdEngine::Instance = nullptr[static]

Definition at line [16](#) of file [XmlLocalStorage.cpp](#).

Class Documentation

DsdEngine::AudioManager Class Reference

```
#include <AudioManager.h>
```

Public Member Functions

- [AudioManager](#) ()
- [~AudioManager](#) ()
- void [init](#) ()
- void [destroy](#) ()
- [SFX loadSFX](#) (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

[AudioManager](#) Class. The [AudioManager](#) Class is responsible for loading and playing of audio within a game. The [AudioManager](#) will also cache any loaded audio.

Definition at line [81](#) of file [AudioManager.h](#).

Constructor & Destructor Documentation

DsdEngine::AudioManager::AudioManager () [inline]

[AudioManager](#) Constructor. On call will init the [AudioManager](#).

Definition at line [87](#) of file [AudioManager.h](#).

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

[AudioManager](#) Deconstructor. On call will destroy current copy of the [AudioManager](#) and clear current cache

Definition at line [93](#) of file [AudioManager.h](#).

Member Function Documentation

void DsdEngine::AudioManager::destroy ()

Destroy current copy of [AudioManager](#) and clear cache maps.

Definition at line [24](#) of file [AudioManager.cpp](#).

void DsdEngine::AudioManager::init ()

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

Definition at line [10](#) of file [AudioManager.cpp](#).

[Music](#) DsdEngine::AudioManager::loadMusic (std::string *audioPath*)

Load [Music](#).

Parameters:

<i>std::string</i>	for path to audio.
--------------------	--------------------

Returns:

[Music](#) to play.

Definition at line [87](#) of file [AudioManager.cpp](#).

[SFX](#) DsdEngine::AudioManager::loadSFX (std::string *audioPath*)

Load Sound effect.

Parameters:

<i>std::string</i>	for path to audio.
--------------------	--------------------

Returns:

[SFX](#) chunk to play.

Definition at line [56](#) of file [AudioManager.cpp](#).

Member Data Documentation

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

Private cache map for storing musics.

Definition at line [129](#) of file [AudioManager.h](#).

bool DsdEngine::AudioManager::m_bisInitialized [private]

Private bool for control loading.

Definition at line [134](#) of file [AudioManager.h](#).

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

Private cache map for storing sound effects.

Definition at line [124](#) of file [AudioManager.h](#).

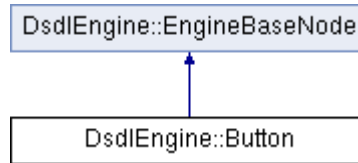
The documentation for this class was generated from the following files:

- [AudioManager.h](#)
- [AudioManager.cpp](#)

DsdEngine::Button Class Reference

```
#include <Button.h>
```

Inheritance diagram for DsdEngine::Button:



Public Member Functions

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

Public Attributes

- [ButtonState](#) [m_eCurrentState](#)

Private Member Functions

- void [onMouseEnters](#) ()
- void [onMouseLeaves](#) ()
- void [onClicked](#) ()

Private Attributes

- [Label](#) * [m_label](#)
- [Sprite](#) * [m_spriteBtn](#)
- std::string [m_buttonName](#)

Additional Inherited Members

Detailed Description

[Button](#) Class subclass of [EngineBaseNode](#). The button class is for creating buttons and handling events on such buttons.

Definition at line [19](#) of file [Button.h](#).

Constructor & Destructor Documentation

DsdEngine::Button::Button ()

[Button](#) Constructor.

Definition at line [6](#) of file [Button.cpp](#).

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

[Button](#) Deconstructor.

Definition at line [11](#) of file [Button.cpp](#).

Member Function Documentation

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

checkInput. Check for input event on the current button.

Parameters:

<i>e</i>	as SDL_Event argument.
----------	------------------------

Definition at line [70](#) of file [Button.cpp](#).

void DsdEngine::Button::createSpriteButton ([Vec2](#) spriteSize, [Vec2](#) position, std::string imagePath, std::string name)

Create button as a [Sprite](#) node.

Parameters:

<i>spriteSize</i>	as a Vec2 argument.
<i>position</i>	as a Vec2 position argument
<i>imagePath</i>	as a std::string path to image
<i>name</i>	as a std::string name of button

Definition at line [33](#) of file [Button.cpp](#).

void DsdEngine::Button::createTextButton ([Vec2](#) pos, [Size](#) size, std::string buttonText, std::string fontPath, SDL_Color textColor, SDL_Color bgColor)

Create button as [Label](#) node.

Parameters:

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

Definition at line [14](#) of file [Button.cpp](#).

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

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

Reimplemented from [DsdEngine::EngineBaseNode](#).

Definition at line [127](#) of file [Button.cpp](#).

std::string DsdEngine::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 DsdEngine::Button::onClicked () [private]

onClicked. Set button state to CLICKED.
Definition at line [64](#) of file [Button.cpp](#).

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

onMouseEnters. Set button state to HOVERING.
Definition at line [53](#) of file [Button.cpp](#).

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

onMouseLeaves. Set button state to NORMAL.
Definition at line [59](#) of file [Button.cpp](#).

Member Data Documentation

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

std::string button name variable.
Definition at line [111](#) of file [Button.h](#).

[ButtonState](#) DsdEngine::Button::m_eCurrentState

ButtonState. Enum class for handling the buttons state.
Definition at line [76](#) of file [Button.h](#).

[Label](#)* DsdEngine::Button::m_label [private]

[Label](#) variable for creating label buttons.
Definition at line [101](#) of file [Button.h](#).

[Sprite](#)* DsdEngine::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](#)
- [Button.cpp](#)

DsdEngine::CollisionShape Class Reference

```
#include <CollisionShape.h>
```

Public Member Functions

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

Protected Attributes

- b2Body * [m_body](#) = nullptr
- b2Fixture * [m_fixtures](#) [1]
- [Vec2](#) [m_dimensions](#)

Detailed Description

[CollisionShape](#) class is for creating a Box2D collision shape around the [Sprite](#) node.

Definition at line [12](#) of file [CollisionShape.h](#).

Constructor & Destructor Documentation

DsdEngine::CollisionShape::CollisionShape ()

Constructor.

Definition at line [6](#) of file [CollisionShane.cpp](#).

DsdEngine::CollisionShape::~~CollisionShape ()

Destructor.

Definition at line [11](#) of file [CollisionShane.cpp](#).

Member Function Documentation

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

destroy shape in the Box2D world.

Parameters:

<i>world</i>	as a b2World pointer argument.
--------------	--------------------------------

Definition at line [45](#) of file [CollisionShane.cpp](#).

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

getBody, get the body for the shape.

Returns:

b2Body pointer.

Definition at line [50](#) of file [CollisionShape.h](#).

const [Vec2](#) DsdEngine::CollisionShape::getDimensions () const [inline]

getDimensions, get the dimensions of the shape.

Returns:

[Vec2](#) dimensions of the shape.

Definition at line [63](#) of file [CollisionShape.h](#).

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

getFixture, get the fixture for index passed in.

Parameters:

<i>index</i>	as a int argument.
--------------	--------------------

Returns:

b2Fixture pointer.

Definition at line [57](#) of file [CollisionShape.h](#).

void DsdEngine::CollisionShape::init (b2World * *world*, [Vec2](#) *position*, [Vec2](#) *dimensions*, float *density*, float *friction*, bool *fixedRotation*)

Initialize shape with arguments passed in.

Parameters:

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

Definition at line [16](#) of file [CollisionShane.cpp](#).

Member Data Documentation

b2Body* DsdEngine::CollisionShape::m_body = nullptr [protected]

b2Body variabel.

Definition at line [69](#) of file [CollisionShape.h](#).

[Vec2](#) DsdEngine::CollisionShape::m_dimensions [protected]

[Vec2](#) dimensions variable.

Definition at line [79](#) of file [CollisionShape.h](#).

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

Array of fixtures for the shape.

Definition at line [74](#) of file [CollisionShape.h](#).

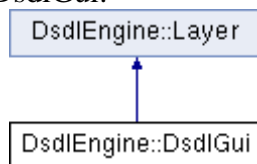
The documentation for this class was generated from the following files:

- [CollisionShape.h](#)
- [CollisionShane.cpp](#)

DsdEngine::DsdGui Class Reference

```
#include <Gui.h>
```

Inheritance diagram for DsdEngine::DsdGui:



Public Member Functions

- [DsdGui](#) ()
- virtual [~DsdGui](#) ()
- void [addButton](#) ([ButtonType](#) type, std::string name, [Vec2](#) pos, [Vec2](#) size, std::string path, SDL_Color color, SDL_Color bgColor, const char *text=NULL)
- void [addLabel](#) ([LableType](#) type, [Vec2](#) pos, std::string text, int fontSize, SDL_Color color, std::string fontFilePath)
- void [addPreDefineLabel](#) ([Label](#) *label, [LableType](#) type)
- void [setGUIPos](#) ()
- void [onSDL_Event](#) (SDL_Event &e)
- void [destroy](#) ()
- [Button](#) * [getButton](#) ()

Public Attributes

- std::vector< [Button](#) * > [GUIElements](#)

Protected Attributes

- [Label](#) * [m_label](#)
- [Button](#) * [m_btn](#)

Detailed Description

GUI [Layer](#) template for creating an a UI [Layer](#). Inherits from layer

Definition at line [20](#) of file [Gui.h](#).

Constructor & Destructor Documentation

DsdEngine::DsdGui::DsdGui ()

Constructor

Definition at line [14](#) of file [Gui.cpp](#).

DsdEngine::DsdGui::~~DsdGui () [virtual]

Destructor

Definition at line [19](#) of file [Gui.cpp](#).

Member Function Documentation

void DsdEngine::DsdGui::addButton ([ButtonType](#) *type*, std::string *name*, [Vec2](#) *pos*, [Vec2](#) *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:

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

Definition at line 24 of file [Gui.cpp](#).

void DsdEngine::DsdGui::addLabel ([LableType](#) *type*, [Vec2](#) *pos*, std::string *text*, int *fontSize*, [SDL_Color](#) *color*, std::string *fontFilePath*)

addLabel, Creates and adds a [Label](#) to the UI layer.

Parameters:

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

Definition at line 46 of file [Gui.cpp](#).

void DsdEngine::DsdGui::addPreDefineLabel ([Label](#) * *label*, [LableType](#) *type*)

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

Parameters:

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

Definition at line 57 of file [Gui.cpp](#).

void DsdEngine::DsdGui::destroy () [virtual]

destroy, Clean up when left scope.

Reimplemented from [DsdEngine::Layer](#).

Definition at line 76 of file [Gui.cpp](#).

[Button](#)* DsdEngine::DsdGui::getButton () [inline]

getButton, Get button from the UIElemets vector

Returns:

[Button](#).

Definition at line 85 of file [Gui.h](#).

void DsdEngine::DsdGui::onSDLEvent (SDL_Event & e)

onSDLEvent, evnet listener for GUI buttons.

Parameters:

<i>e, evnent</i>	to listen on.
------------------	---------------

Definition at line [68](#) of file [Gui.cpp](#).

void DsdEngine::DsdGui::setGUIPos ()

setGUIPos, set the GUI position

Definition at line [63](#) of file [Gui.cpp](#).

Member Data Documentation

std::vector<[Button](#)*> DsdEngine::DsdGui::GUIElements

Vector to hold GUI Elements

Definition at line [79](#) of file [Gui.h](#).

[Button](#)* DsdEngine::DsdGui::m_btn [protected]

Definition at line [90](#) of file [Gui.h](#).

[Label](#)* DsdEngine::DsdGui::m_label [protected]

Definition at line [89](#) of file [Gui.h](#).

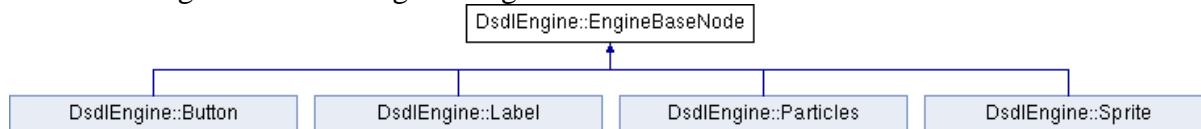
The documentation for this class was generated from the following files:

- [Gui.h](#)
- [Gui.cpp](#)

DsdEngine::EngineBaseNode Class Reference

```
#include <EngineBaseNode.h>
```

Inheritance diagram for DsdEngine::EngineBaseNode:



Public Member Functions

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

Protected Attributes

- std::string [m_assetPath](#)
- [NodeType](#) [nodeType](#) = [NodeType::BASENODE](#)
- [ResourceTexture](#) * [m_engineTexture](#)
- SDL_Rect * [m_objectBoundingBox](#)
- [Vec2](#) [m_position](#)
- [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 [m_labelText](#)
- int [m_textSize](#)
- SDL_Color [m_textColor](#)
- [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

DsdEngine::EngineBaseNode::EngineBaseNode ()

Constructor

Definition at line [10](#) of file [EngineBaseNode.cpp](#).

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

Deconstructor

Definition at line [23](#) of file [EngineBaseNode.cpp](#).

Member Function Documentation

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

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

Reimplemented in [DsdEngine::Label](#).

Definition at line [165](#) of file [EngineBaseNode.cpp](#).

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

virtual destroy, destorys node when it leaves scope.

Reimplemented in [DsdEngine::Sprite](#), [DsdEngine::Label](#), and [DsdEngine::Button](#).

Definition at line [154](#) of file [EngineBaseNode.cpp](#).

std::string DsdEngine::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* DsdIEngine::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 [107](#) of file [EngineBaseNode.h](#).

[ResourceTexture](#)* DsdIEngine::EngineBaseNode::getEngineTexture () [inline]

getEngineTexture, get the [ResourceTexture](#) for the node.

Returns:

[ResourceTexture](#) pointer.

Definition at line [168](#) of file [EngineBaseNode.h](#).

[NodeType](#) DsdIEngine::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](#) DsdIEngine::EngineBaseNode::getPosition () const [inline]

getPosition, get position of the node.

Returns:

const [Vec2](#).

Definition at line [83](#) of file [EngineBaseNode.h](#).

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

isTextureChange, check if the texture was changed.

Returns:

bool.

Definition at line [200](#) of file [EngineBaseNode.h](#).

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

load, load node as SDL_Texture.

Parameters:

<i>r</i>	as SDL_Renderer argument.
----------	---------------------------

Returns:

bool.

Definition at line [61](#) of file [EngineBaseNode.cpp](#).

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

render, Render node to window.

Parameters:

<i>r</i>	as SDL_Renderer argument
----------	--------------------------

Definition at line 30 of file [EngineBaseNode.cpp](#).

void DsdEngine::EngineBaseNode::renderCollisionShape (SDL_Renderer * r, [CollisionShape](#) * shape)

renderCollisionShape, Render collision shape for node.

Parameters:

<i>r</i>	as a SDL_Renderer argument.
<i>shape</i>	as a CollisionShape pointer argument.

Definition at line 170 of file [EngineBaseNode.cpp](#).

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

scaleHeight, Scale the height of the node.

Parameters:

<i>scale</i>	as a float argument
--------------	---------------------

Definition at line 131 of file [EngineBaseNode.h](#).

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

scaleNode, Scale the node size by value passed in.

Parameters:

<i>scale</i>	as a float argument
--------------	---------------------

Definition at line 119 of file [EngineBaseNode.h](#).

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

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

Definition at line 125 of file [EngineBaseNode.h](#).

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

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

Parameters:

<i>path</i>	as a std::string argument.
-------------	----------------------------

Definition at line 138 of file [EngineBaseNode.h](#).

void DsdEngine::EngineBaseNode::setBoundingBox ([Vec2](#) pos, [Vec2](#) size)

setBoundingBox, the the bounding box for the node.

Parameters:

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

Definition at line 131 of file [EngineBaseNode.cpp](#).

void DsdEngine::EngineBaseNode::setEngineNodeType ([NodeType](#) type)[inline]

setEngineNodeType, set the type for a specific node.

Parameters:

<i>type</i>	as a NodeType argument.
-------------	---

Definition at line [156](#) of file [EngineBaseNode.h](#).

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

setHeight, Set height of the node.

Parameters:

<i>h</i>	as a const int argument
----------	-------------------------

Definition at line [101](#) of file [EngineBaseNode.h](#).

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

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

Parameters:

<i>opacity</i>	as an int argument between 0 - 255
----------------	------------------------------------

Definition at line [144](#) of file [EngineBaseNode.cpp](#).

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

setPosition, Set position of the node.

Parameters:

<i>pos</i>	as a const Vec2 argument
------------	--

Definition at line [65](#) of file [EngineBaseNode.h](#).

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

setPositionX, Set X position of the node.

Parameters:

<i>x</i>	as a const int argument
----------	-------------------------

Definition at line [71](#) of file [EngineBaseNode.h](#).

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

setPositionY, Set Y position of the node.

Parameters:

<i>Y</i>	as a const int argument
----------	-------------------------

Definition at line [77](#) of file [EngineBaseNode.h](#).

void DsdEngine::EngineBaseNode::setSize ([Size](#) si)[inline]

setSize, Set size of the node.

Parameters:

<i>si</i>	as a const Size argument
-----------	--

Definition at line [89](#) of file [EngineBaseNode.h](#).

void DsdIEngine::EngineBaseNode::setUpdateTextureTrue (bool *value*) [inline]

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

Parameters:

<i>value</i>	as a bool argument.
--------------	---------------------

Definition at line [194](#) of file [EngineBaseNode.h](#).

void DsdIEngine::EngineBaseNode::setWidth (int *w*) [inline]

setWidth, Set width of the node.

Parameters:

<i>w</i>	as a const int argument
----------	-------------------------

Definition at line [95](#) of file [EngineBaseNode.h](#).

void DsdIEngine::EngineBaseNode::updateLabelText (std::string *text*)

updateLabelText, change the display text of a label.

Parameters:

<i>text</i>	as a std::string argument.
-------------	----------------------------

Definition at line [139](#) of file [EngineBaseNode.cpp](#).

Member Data Documentation

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

std::string asset path

Definition at line [205](#) of file [EngineBaseNode.h](#).

[CollisionShape](#)* DsdIEngine::EngineBaseNode::m_CollisionShape [protected]

Box2D collision shape of the node

Definition at line [233](#) of file [EngineBaseNode.h](#).

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

the current frame rect

Definition at line [220](#) of file [EngineBaseNode.h](#).

[ResourceTexture](#)* DsdIEngine::EngineBaseNode::m_engineTexture [protected]

[ResourceTexture](#) for the node

Definition at line [208](#) of file [EngineBaseNode.h](#).

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

the font to use for labels

Definition at line [224](#) of file [EngineBaseNode.h](#).

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

std::map for caching the font

Definition at line [225](#) of file [EngineBaseNode.h](#).

int DsdEngine::EngineBaseNode::m_frame [protected]

Definition at line [215](#) of file [EngineBaseNode.h](#).

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

frames rect for animation

Definition at line [219](#) of file [EngineBaseNode.h](#).

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

std::string to hold label display text

Definition at line [227](#) of file [EngineBaseNode.h](#).

int DsdEngine::EngineBaseNode::m_numFrames [protected]

Definition at line [215](#) of file [EngineBaseNode.h](#).

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

SDL_Rect bounding box for the node

Definition at line [209](#) of file [EngineBaseNode.h](#).

int DsdEngine::EngineBaseNode::m_opacity [protected]

int values for frames & opacity

Definition at line [215](#) of file [EngineBaseNode.h](#).

[Vec2](#) DsdEngine::EngineBaseNode::m_position [protected]

[Vec2](#) Position of the node

Definition at line [212](#) of file [EngineBaseNode.h](#).

[Vec2](#) DsdEngine::EngineBaseNode::m_size [protected]

[Vec2](#) Size of the node

Definition at line [213](#) of file [EngineBaseNode.h](#).

SDL_Color DsdEngine::EngineBaseNode::m_textColor [protected]

color of the label

Definition at line [229](#) of file [EngineBaseNode.h](#).

int DsdEngine::EngineBaseNode::m_textSize [protected]

int for size of font

Definition at line [228](#) of file [EngineBaseNode.h](#).

[NodeType](#) DsdEngine::EngineBaseNode::nodeType = [NodeType::BASENODE](#) [protected]

NodeType for containing node type

Definition at line [206](#) of file [EngineBaseNode.h](#).

bool DsdEngine::EngineBaseNode::updateTextureInfo [protected]

bool for texture control

Definition at line [217](#) of file [EngineBaseNode.h](#).

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

- [EngineBaseNode.h](#)
- [EngineBaseNode.cpp](#)

DsdEngine::EngineMaster Class Reference

```
#include <EngineMaster.h>
```

Static Public Member Functions

- static [EngineMaster](#) * [getInstance](#) ()

Protected Member Functions

- [EngineMaster](#) ()
- virtual [~EngineMaster](#) ()

Detailed Description

[EngineMaster](#) is a static singleton helper class

Definition at line [14](#) of file [EngineMaster.h](#).

Constructor & Destructor Documentation

DsdEngine::EngineMaster::EngineMaster () [[inline](#)], [[protected](#)]

Constructor

Definition at line [27](#) of file [EngineMaster.h](#).

virtual DsdEngine::EngineMaster::~~EngineMaster () [[inline](#)], [[protected](#)], [[virtual](#)]

Destructor

Definition at line [32](#) of file [EngineMaster.h](#).

Member Function Documentation

[EngineMaster](#) * **DsdEngine::EngineMaster::getInstance ()** [[static](#)]

getInstance, create [EngineMaster](#) as a Static instance. static instance of [EngineMaster](#)

Definition at line [7](#) of file [EngineMaster.cpp](#).

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

- [EngineMaster.h](#)
- [EngineMaster.cpp](#)

DsdEngine::FileIO Class Reference

```
#include <FileIO.h>
```

Public Member Functions

- std::string [getSuitableFOpen](#) (const std::string &filenameUtf8) const
- std::string [getWritablePath](#) ()
- void [setAssetsPath](#) (std::string assetsPath)
- std::string [getFileToOpen](#) ()
- void [setFileToOpen](#) (std::string file)
- bool [loadDocument](#) (const char *filepath, char **doc_contents)
- bool [writeDocument](#) (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 [createXMLFile](#) ()

Static Public Member Functions

- static [FileIO](#) * [getInstance](#) ()

Protected Member Functions

- [FileIO](#) ()
- virtual [~FileIO](#) ()

Private Attributes

- std::string [m_path](#)
- std::string [m_fileName](#)

Detailed Description

[FileIO](#) class handles open and closing of xml files in the framework. Handles XML parsing and Saving
Definition at line [21](#) of file [FileIO.h](#).

Constructor & Destructor Documentation

DsdEngine::FileIO::FileIO () [[inline](#)], [[protected](#)]

Constructor

Definition at line [104](#) of file [FileIO.h](#).

virtual DsdEngine::FileIO::~FileIO () [[inline](#)], [[protected](#)], [[virtual](#)]

Destructor

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 [54](#) of file [FileIO.h](#).

[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:

<i>std::string</i>	file path.
--------------------	------------

Returns:

std::string file path to open.

Definition at line [22](#) of file [FileIO.cpp](#).

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

getWriteablePath, get the full path to the file.

Returns:

std::string.

Definition at line [37](#) of file [FileIO.cpp](#).

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:

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

Returns:

XMLElement the element matching the key.

Definition at line [98](#) of file [FileIO.cpp](#).

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

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

Parameters:

<i>filepath</i>	const char path to file,
<i>doc_contents</i>	buffer to hold the file contents.

Returns:

bool.

Definition at line [53](#) of file [FileIO.cpp](#).

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

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

Parameters:

<i>std::string</i>	path for file.
--------------------	----------------

Definition at line [47](#) of file [FileIO.h](#).

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

setFileToOpen, set the name of the file to open.

Parameters:

<i>std::string</i>	file name.
--------------------	------------

Definition at line [60](#) of file [FileIO.h](#).

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

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

Parameters:

<i>vlaue</i>	the value to be set.
<i>key</i>	the key to look for.

Definition at line [156](#) of file [FileIO.cpp](#).

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

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

Parameters:

<i>filepath</i>	const char path to file,
<i>doc_contents</i>	buffer cotaining the file contents.

Returns:

bool.

Definition at line [79](#) of file [FileIO.cpp](#).

Member Data Documentation

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

name of xml file to load

Definition at line [114](#) of file [FileIO.h](#).

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

path to folder which contains file

Definition at line [109](#) of file [FileIO.h](#).

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

- [FileIO.h](#)
- [FileIO.cpp](#)

DsdEngine::FpsLimiter Class Reference

```
#include <Timing.h>
```

Public Member Functions

- [FpsLimiter](#) ()
- [~FpsLimiter](#) ()
- void [init](#) (float maxFPS)
- void [setMaxFPS](#) (float maxFPS)
- void [begin](#) ()
- float [end](#) ()

Private Member Functions

- void [calculateFPS](#) ()

Private Attributes

- float [m_fFps](#)
- float [m_fMaxFPS](#)
- float [m_fFrameTime](#)
- unsigned int [m_iStartTicks](#)

Detailed Description

Timing file handles setting up of calculating and controlling frame rate of the engine.

Definition at line [12](#) of file [Timing.h](#).

Constructor & Destructor Documentation

DsdEngine::FpsLimiter::FpsLimiter ()

Constructor

Definition at line [10](#) of file [Timing.cpp](#).

DsdEngine::FpsLimiter::~~FpsLimiter ()

Destructor

Definition at line [13](#) of file [Timing.cpp](#).

Member Function Documentation

void DsdEngine::FpsLimiter::begin ()

Start the Frame Rate Timer

Definition at line [28](#) of file [Timing.cpp](#).

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

Calculate the running fps and keep it under control

Definition at line [45](#) of file [Timing.cpp](#).

float DsdEngine::FpsLimiter::end ()

End the frame rate timer

Returns:

float, the current fps vlaue.

Definition at line [33](#) of file [Timing.cpp](#).

void DsdEngine::FpsLimiter::init (float *maxFPS*)

Initializes the FPS limiter.

Parameters:

<i>maxFPS, the</i>	max frame rate allowed.
--------------------	-------------------------

Definition at line [18](#) of file [Timing.cpp](#).

void DsdEngine::FpsLimiter::setMaxFPS (float *maxFPS*)

Sets the desired max FPS

Parameters:

<i>maxFPS, the</i>	desired Frame Rate.
--------------------	---------------------

Definition at line [23](#) of file [Timing.cpp](#).

Member Data Documentation

float DsdEngine::FpsLimiter::m_fFps [private]

Definition at line [53](#) of file [Timing.h](#).

float DsdEngine::FpsLimiter::m_fFrameTime [private]

float values for claculations

Definition at line [53](#) of file [Timing.h](#).

float DsdEngine::FpsLimiter::m_fMaxFPS [private]

Definition at line [53](#) of file [Timing.h](#).

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

starting timestamp

Definition at line [54](#) of file [Timing.h](#).

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

- [Timing.h](#)
- [Timing.cpp](#)

DsdEngine::IMainGame Class Reference

```
#include <IMainGame.h>
```

Public Member Functions

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

Public Attributes

- [InputManager m_InputManager](#)

Protected Attributes

- std::unique_ptr< [SceneManager](#) > [m_pSceneManager](#)
- [IScene](#) * [m_pCurrentRunning](#)
- bool [m_bIsRunning](#)
- bool [m_bIsPaused](#)
- [Window m_Window](#)
- SDL_Renderer * [m_pGameRenderer](#)
- [AudioManager m_audioManager](#)

Private Member Functions

- const float [getFps](#) () const
- void [mainLoop](#) ()
- void [update](#) ()
- void [draw](#) ()
- bool [init](#) ()
- bool [initSystems](#) ()
- void [exitGame](#) ()

Private Attributes

- float [m_fFps](#)
- unsigned int [windowFlag](#)
- int [m_windowWidth](#)
- int [m_windowHeight](#)
- std::string [windowtitle](#)
- std::string [mainAssetsPath](#)

Detailed Description

[IMainGame](#) is the heart of the engine as it contains the main 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

DsdEngine::IMainGame::IMainGame ()

Constructor

Definition at line [22](#) of file [IMainGame.cpp](#).

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

Destructor

Definition at line [26](#) of file [IMainGame.cpp](#).

Member Function Documentation

virtual void DsdEngine::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 DsdEngine::IMainGame::checkPaused () [inline]

checkPaused, check if the game is currently paused.

Returns:

bool

Definition at line [99](#) of file [IMainGame.h](#).

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

exitGame. Cleans up and exits the game.

Definition at line [246](#) of file [IMainGame.cpp](#).

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

getFps, Get the running Frame Rate.

Returns:

float fps,

Definition at line [132](#) of file [IMainGame.h](#).

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

init, Initilazie the engine subsystems.

Returns:

bool.

Definition at line [129](#) of file [IMainGame.cpp](#).

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 [35](#) of file [IMainGame.cpp](#).

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::onInit () [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:

<i>envt</i>	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 [58](#) of file [IMainGame.cpp](#).

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

setFps, Set the desired frame rate for the game.

Parameters:

<i>fps</i>	as a float value
------------	------------------

Definition at line [58](#) of file [IMainGame.h](#).

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

setPaused, Pauses the main game loop.

Definition at line [88](#) of file [IMainGame.h](#).

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

setRunning, Starts the game loop running if paused.

Definition at line [93](#) of file [IMainGame.h](#).

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

setupWindow, sets up the window defaults for Windows Platform.

Parameters:

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

Definition at line [110](#) of file [IMainGame.cpp](#).

void DsdEngine::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

[AudioManager](#) DsdEngine::IMainGame::m_audioManager [protected]

the main [AudioManager](#)

Definition at line [113](#) of file [IMainGame.h](#).

bool DsdEngine::IMainGame::m_bIsPaused [protected]

bool variables for control

Definition at line [108](#) of file [IMainGame.h](#).

bool DsdEngine::IMainGame::m_bIsRunning [protected]

Definition at line [108](#) of file [IMainGame.h](#).

float DsdEngine::IMainGame::m_fFps [private]

engines fps

Definition at line [118](#) of file [IMainGame.h](#).

[InputManager](#) DsdEngine::IMainGame::m_InputManager

Main games inputmanage object

Definition at line [101](#) of file [IMainGame.h](#).

[IScene](#)* DsdEngine::IMainGame::m_pCurrentRunning [protected]

current running scene

Definition at line [107](#) of file [IMainGame.h](#).

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

the engine renderer

Definition at line [111](#) of file [IMainGame.h](#).

std::unique_ptr<[SceneManager](#)> DsdEngine::IMainGame::m_pSceneManager [protected]

Main Scene Manager for the Engine

Definition at line [104](#) of file [IMainGame.h](#).

[Window](#) DsdEngine::IMainGame::m_Window [protected]

the main window variable

Definition at line [110](#) of file [IMainGame.h](#).

int DsdEngine::IMainGame::m_windowHeight [private]

window height variable

Definition at line [123](#) of file [IMainGame.h](#).

int DsdEngine::IMainGame::m_windowWidth [private]

window width variabel

Definition at line [122](#) of file [IMainGame.h](#).

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

asset path to windows assets folder

Definition at line [125](#) of file [IMainGame.h](#).

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

windowFlag variable

Definition at line [121](#) of file [IMainGame.h](#).

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

window title variable

Definition at line [124](#) of file [IMainGame.h](#).

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

- [IMainGame.h](#)
- [IMainGame.cpp](#)

DsdEngine::InputManager Class Reference

```
#include <InputManager.h>
```

Public Member Functions

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

Private Member Functions

- bool [wasKeyDown](#) (unsigned int keyID)

Private Attributes

- std::unordered_map< unsigned int, bool > [keyMap](#)
- std::unordered_map< unsigned int, bool > [previousKeyMap](#)
- bool [swipeup](#)
- bool [swipedown](#)
- bool [swipeleft](#)
- bool [swiperight](#)
- bool [fingerDown](#)
- bool [fingerUp](#)

Detailed Description

[InputManager](#) Class handles all input in the game

Definition at line [15](#) of file [InputManager.h](#).

Constructor & Destructor Documentation

DsdEngine::InputManager::InputManager ()

Constructor

Definition at line [6](#) of file [InputManager.cpp](#).

DsdEngine::InputManager::~~InputManager ()

Deconstructor

Definition at line [13](#) of file [InputManager.cpp](#).

Member Function Documentation

bool DsdEngine::InputManager::isKeyDown (unsigned int *keyID*)

isKeyDown, check if key is down.

Parameters:

<i>keyID</i> , <i>ID</i>	of key to be checked.
--------------------------	-----------------------

Returns:

bool.

Definition at line [39](#) of file [InputManager.cpp](#).

bool DsdEngine::InputManager::isKeyPressed (unsigned int *keyID*)

isKeyPressed, check if key was just pressed.

Parameters:

<i>keyID</i> , <i>ID</i>	of key to be checked.
--------------------------	-----------------------

Returns:

bool.

Definition at line [69](#) of file [InputManager.cpp](#).

bool DsdEngine::InputManager::isKeyReleased (unsigned int *KeyID*)

isKeyReleased, check if key was just released.

Parameters:

<i>keyID</i> , <i>ID</i>	of key to be checked.
--------------------------	-----------------------

Returns:

bool.

Definition at line [79](#) of file [InputManager.cpp](#).

bool DsdEngine::InputManager::isSwipe (SDL_Event & *evnt*)

isSwipe, check if it was a swipe event.

Parameters:

<i>evnt</i> , <i>event</i>	to be checked.
----------------------------	----------------

Returns:

bool.

Definition at line [100](#) of file [InputManager.cpp](#).

bool DsdEngine::InputManager::isSwipeDown ()

isSwipeDown, check if event was a swipe down.

Returns:

bool.

Definition at line [154](#) of file [InputManager.cpp](#).

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

isSwipeLeft, check if event was a swipe left.

Returns:

bool.

Definition at line [159](#) of file [InputManager.cpp](#).

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

isSwipeRight, check if event was a swipe right.

Returns:

bool.

Definition at line [164](#) of file [InputManager.cpp](#).

bool DsdLEngine::InputManager::isSwipeUp ()

isSwipeUP, check if event was a swipe up.

Returns:

bool.

Definition at line [149](#) of file [InputManager.cpp](#).

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

isTouch, check if it was touch event.

Parameters:

<i>keyID, ID</i>	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:

<i>keyID, the</i>	id of the key that was pressed.
-------------------	---------------------------------

Definition at line [25](#) of file [InputManager.cpp](#).

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

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

Parameters:

<i>keyID, the</i>	id of the key that was pressed.
-------------------	---------------------------------

Definition at line [30](#) of file [InputManager.cpp](#).

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

setMouseCoords, set the coordinates for the mouse.

Parameters:

<i>x,float</i>	value for mouse x location.
<i>y,float</i>	value for mouse y location.

Definition at line [34](#) of file [InputManager.cpp](#).

void DsdEngine::InputManager::update ()

update, loops through the key map.

Definition at line [18](#) of file [InputManager.cpp](#).

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

wasKeyDown, Check if key was down.

Parameters:

<i>keyID,ID</i>	of key to be checked.
-----------------	-----------------------

Returns:

bool.

Definition at line [54](#) of file [InputManager.cpp](#).

Member Data Documentation**std::unordered_map<unsigned int, bool> DsdEngine::InputManager::_keyMap [private]**

map to hold current keys

Definition at line [119](#) of file [InputManager.h](#).

**std::unordered_map<unsigned int, bool>
DsdEngine::InputManager::_previousKeyMap [private]**

map to hold previous keys

Definition at line [120](#) of file [InputManager.h](#).

bool DsdEngine::InputManager::fingerDown [private]

Definition at line [123](#) of file [InputManager.h](#).

bool DsdEngine::InputManager::fingerUp [private]

bools to control touch checking

Definition at line [123](#) of file [InputManager.h](#).

bool DsdEngine::InputManager::swipedown [private]

Definition at line [122](#) of file [InputManager.h](#).

bool DsdEngine::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 [122](#) of file [InputManager.h](#).

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

- [InputManager.h](#)
- [InputManager.cpp](#)

DsdEngine::IScene Class Reference

```
#include <IScene.h>
```

Public Member Functions

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

Public Attributes

- std::vector< [Layer](#) * > [sceneLayers](#)

Protected Attributes

- [SceneState](#) [m_eCurrentState](#) = [SceneState::NONE](#)
- [IMainGame](#) * [m_game](#) = nullptr
- int [m_iSceneIndex](#) = [SCENE_INDEX_NO_SCENE](#)
- [InputManager](#) [m_inputManager](#)

Friends

- class [SceneManager](#)
Friend Classes.
- class [InputManager](#)

Detailed Description

[IScene](#) is a interface class to inherith from when creating a scene in the game.

Definition at line [35](#) of file [IScene.h](#).

Constructor & Destructor Documentation

DsdEngine::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:

<i>layer</i> , Layer	to add to the scene.
--------------------------------------	----------------------

Definition at line [123](#) 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:

<i>r</i> , SDL_Renderer	to use when rendering.
---	------------------------

Definition at line [141](#) of file [IScene.h](#).

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 [93](#) of file [IScene.h](#).

[SceneState](#) DsdlEngine::~IScene::getSceneState () const [inline]

Get the current scenes state.

Returns:

SceneState int.

Definition at line [99](#) of file [IScene.h](#).

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

Load the scene and its layers.

Parameters:

<i>r,SDL_Renderer</i>	to use when loading
-----------------------	---------------------

Definition at line [131](#) of file [IScene.h](#).

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

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

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

Pure virtual function. Called when scene leaves focus.

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

Virtual function for scene specific input.

Definition at line [12](#) of file [Scene.cpp](#).

void DsdEngine::IScene::setParentGame (IMainGame * game)[inline]

Set the game that the scene belongs to. game. the [IMainGame](#) the scene belongs to.

Definition at line [110](#) of file [IScene.h](#).

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

Set a scene running by setting the state.

Definition at line [104](#) of file [IScene.h](#).

virtual void DsdEngine::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 [150](#) of file [IScene.h](#).

friend class [SceneManager](#) [friend]

Friend Classes.

Definition at line [149](#) of file [IScene.h](#).

Member Data Documentation

[SceneState](#) DsdEngine::IScene::m_eCurrentState = [SceneState::NONE](#) [protected]

Scenes current state variabel

Definition at line [152](#) of file [IScene.h](#).

[IMainGame](#)* DsdEngine::IScene::m_game = nullptr [protected]

parent game.

Definition at line [154](#) of file [IScene.h](#).

[InputManager](#) DsdEngine::IScene::m_inputManager [protected]

scenes input manager

Definition at line [156](#) of file [IScene.h](#).

int DsdEngine::IScene::m_iSceneIndex = [SCENE_INDEX_NO_SCENE](#) [protected]

scene index int

Definition at line [155](#) of file [IScene.h](#).

std::vector<[Layer](#)> DsdEngine::IScene::sceneLayers

Vector to hold scenes game [Layer](#)

Definition at line [117](#) of file [IScene.h](#).

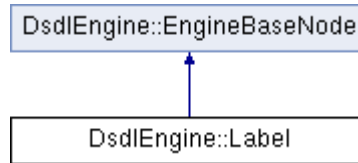
The documentation for this class was generated from the following files:

- [IScene.h](#)
- [Scene.cpp](#)

DsdEngine::Label Class Reference

```
#include <Label.h>
```

Inheritance diagram for DsdEngine::Label:



Public Member Functions

- [Label](#) ()
- virtual [~Label](#) ()
- void [create](#) ([Vec2](#) pos, std::string text, int fontSize, SDL_Color color, std::string fontFilePath)
- void [setType](#) ([LableType](#) type)
- const int [getType](#) ()
- void [destroy](#) ()
- void [cleanup](#) ()

Protected Attributes

- [LableType](#) m_labelType
-

Detailed Description

[Label](#) class is the base for all labels in the engine it inherits for [EngineBaseNode](#).

Definition at line [12](#) of file [Label.h](#).

Constructor & Destructor Documentation

DsdEngine::Label::Label ()

Definition at line [11](#) of file [Label.cpp](#).

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

Deconstructor

Definition at line [16](#) of file [Label.cpp](#).

Member Function Documentation

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

Cleanup the lable texture.

Reimplemented from [DsdEngine::EngineBaseNode](#).

Definition at line [39](#) of file [Label.cpp](#).

void DsdEngine::Label::create ([Vec2](#) pos, std::string text, int fontSize, SDL_Color color, std::string fontFilePath)

Create a basic [Label](#) pos, [Vec2](#) 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 [21](#) of file [Label.cpp](#).

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

Destroy the label.

Reimplemented from [DsdEngine::EngineBaseNode](#).

Definition at line [34](#) of file [Label.cpp](#).

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

Get the type of label it is.

Returns:

int label type,

Definition at line [44](#) of file [Label.h](#).

void DsdEngine::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](#) DsdEngine::Label::m_labelType [protected]

LabelType variable

Definition at line [57](#) of file [Label.h](#).

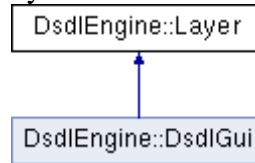
The documentation for this class was generated from the following files:

- [Label.h](#)
- [Label.cpp](#)

DsdEngine::Layer Class Reference

```
#include <Layer.h>
```

Inheritance diagram for DsdEngine::Layer:



Public Member Functions

- [Layer](#) ()
- virtual [~Layer](#) ()
- virtual void [destroy](#) ()
- void [loadNodes](#) (SDL_Renderer *r)
- void [drawNodes](#) (SDL_Renderer *r)
- void [addNodeToLayer](#) ([EngineBaseNode](#) *node)
- void [removeNodeFromLayer](#) ([EngineBaseNode](#) *node)

Public Attributes

- std::vector< [EngineBaseNode](#) * > [layerNodes](#)

Friends

- class [Gui](#)

Detailed Description

Definition at line [15](#) of file [Layer.h](#).

Constructor & Destructor Documentation

DsdEngine::Layer::Layer ()

Constructor

Definition at line [13](#) of file [Layer.cpp](#).

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

Destructor

Definition at line [19](#) of file [Layer.cpp](#).

Member Function Documentation

void DsdEngine::Layer::addNodeToLayer ([EngineBaseNode](#) * node)

Add A node to the layer.

Parameters:

<i>node</i> , EngineBaseNode	to add to the Layer
--	-------------------------------------

Definition at line [38](#) of file [Layer.cpp](#).

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

Destroy the layer and all its contents

Reimplemented in [DsdEngine::DsdGui](#).

Definition at line [24](#) of file [Layer.cpp](#).

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

Draw all nodes in the layer

Parameters:

<i>r</i> , SDL_Renderer	to be used when rendering
---	---------------------------

Definition at line [57](#) of file [Layer.cpp](#).

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

Load all nodes in the layer

Parameters:

<i>r</i> , SDL_Renderer	to be used when loading
---	-------------------------

Definition at line [50](#) of file [Layer.cpp](#).

void DsdEngine::Layer::removeNodeFromLayer ([EngineBaseNode](#) * node)

Remove a node from the layer.

Parameters:

<i>node</i> , EngineBaseNode	to be removed from the layer
--	------------------------------

Definition at line [44](#) of file [Layer.cpp](#).

Friends And Related Function Documentation**friend class Gui [friend]**

Definition at line [18](#) of file [Layer.h](#).

Member Data Documentation**std::vector<[EngineBaseNode](#)> DsdEngine::Layer::layerNodes**

vector to hold layer nodes

Definition at line [59](#) of file [Layer.h](#).

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

- [Layer.h](#)
- [Layer.cpp](#)

DsdEngine::Music Class Reference

```
#include <AudioManager.h>
```

Public Member Functions

- void [play](#) (int loop=-1)
- void [audioPauseBG](#) ()
- void [audioResumeBG](#) ()
- void [audioStopBG](#) ()

Private Attributes

- Mix_Music * [m_Music](#)

Friends

- class [AudioManager](#)

Detailed Description

[Music](#) class for interfacing with SDL Mix_Music.

Definition at line [39](#) of file [AudioManager.h](#).

Member Function Documentation

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

Pause [Music](#) currently playeing.

Definition at line [55](#) of file [AudioManager.h](#).

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

Resume [Music](#) that is currently paused.

Definition at line [60](#) of file [AudioManager.h](#).

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

Stop [Music](#) that is currently playing.

Definition at line [65](#) of file [AudioManager.h](#).

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

Play [Music](#).

Parameters:

<i>loops</i>	== -1 : loop forever, 0 : loop once, 1+ : loop that many times
--------------	--

Definition at line [50](#) of file [AudioManager.h](#).

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* DsdEngine::Music::m_Music [private]

Private Mix_Music Variable

Definition at line [65](#) of file [AudioManager.h](#).

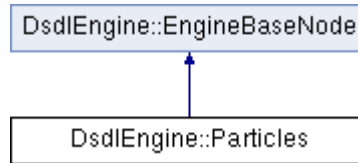
The documentation for this class was generated from the following file:

- [AudioManager.h](#)

DsdlEngine::Particles Class Reference

```
#include <Particles.h>
```

Inheritance diagram for DsdlEngine::Particles:



Public Member Functions

- [Particles](#) (int x, int y)
- [~Particles](#) ()
- bool [isDead](#) ([Particles](#) *p)

Static Public Member Functions

- static float [torad](#) (float [angle](#))

Private Attributes

- int [life](#)
- float [mPosX](#)
- float [mPosY](#)
- float [xvel](#)
- float [yvel](#)
- float [angle](#)
- float [size](#)
- Uint32 [endtime](#)

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 DsdEngine::Particles::isDead ([Particles](#) * *p*)

Definition at line [19](#) of file [Particles.cpp](#).

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

Definition at line [18](#) of file [Particles.h](#).

Member Data Documentation

float DsdEngine::Particles::angle [private]

Definition at line [26](#) of file [Particles.h](#).

UInt32 DsdEngine::Particles::endtime [private]

Definition at line [27](#) of file [Particles.h](#).

int DsdEngine::Particles::life [private]

Definition at line [25](#) of file [Particles.h](#).

float DsdEngine::Particles::mPosX [private]

Definition at line [26](#) of file [Particles.h](#).

float DsdEngine::Particles::mPosY [private]

Definition at line [26](#) of file [Particles.h](#).

float DsdEngine::Particles::size [private]

Definition at line [26](#) of file [Particles.h](#).

float DsdEngine::Particles::xvel [private]

Definition at line [26](#) of file [Particles.h](#).

float DsdEngine::Particles::yvel [private]

Definition at line [26](#) of file [Particles.h](#).

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

- [Particles.h](#)
- [Particles.cpp](#)

DsdEngine::ResourceTexture Class Reference

```
#include <ResourceTexture.h>
```

Public Member Functions

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

Private Attributes

- SDL_Texture * [m_Texture](#)
- std::map< std::string, SDL_Texture * > [m_TextureMap](#)
- int [m_iWidth](#)
- int [m_iHeight](#)

Detailed Description

[ResourceTexture](#) is responsible for loading and rendering all textures in the game. it is the base class for all textures.

Definition at line [15](#) of file [ResourceTexture.h](#).

Constructor & Destructor Documentation

DsdEngine::ResourceTexture::ResourceTexture ()

Constructor

Definition at line [24](#) of file [ResourceTexture.cpp](#).

DsdEngine::ResourceTexture::~~ResourceTexture ()

Destructor

Definition at line [31](#) of file [ResourceTexture.cpp](#).

Member Function Documentation

void DsdEngine::ResourceTexture::destroy ()

Destroy the texture

Definition at line [129](#) of file [ResourceTexture.cpp](#).

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

LoadTexture loads in sprite texture from the giving asset path.

Parameters:

<i>texturePath</i> ,std:: <i>string</i> <i>ring</i>	to the asset.
<i>r,the</i>	Renderer to use in loading

Returns:

bool

Definition at line [37](#) of file [ResourceTexture.cpp](#).

bool DsdEngine::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:

<i>text,text</i>	to display onthe texture
<i>color,the</i>	SDL_Color to use for the texture.
<i>myFont,the</i>	TTF_Font to use.
<i>r,the</i>	SDL_Renderer to use.

Returns:

bool.

Definition at line [80](#) of file [ResourceTexture.cpp](#).

void DsdEngine::ResourceTexture::render ([Vec2](#) *p*, [Vec2](#) *s*, SDL_Renderer * *r*, SDL_Rect * *clip* = NULL)

Render a texture to the window

Parameters:

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

Definition at line [107](#) of file [ResourceTexture.cpp](#).

void DsdEngine::ResourceTexture::setAlpha (Uint8 *alpha*)

Set the Alpha for a texture

Parameters:

<i>alpha,UNit8</i>	value of Alpha to use.
--------------------	------------------------

Definition at line [150](#) of file [ResourceTexture.cpp](#).

void DsdEngine::ResourceTexture::setBlendMode (SDL_BlendMode *blending*)

Set the blend mode of the texture.

Parameters:

<i>blending.</i>	Blendmode to use.
------------------	-------------------

Definition at line [141](#) of file [ResourceTexture.cpp](#).

Member Data Documentation

int DsdEngine::ResourceTexture::m_iHeight [private]

widht and height of the SDL_Texture

Definition at line [75](#) of file [ResourceTexture.h](#).

int DsdEngine::ResourceTexture::m_iWidth [private]

Definition at line [75](#) of file [ResourceTexture.h](#).

SDL_Texture* DsdEngine::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*> DsdEngine::ResourceTexture::m_TextureMap [private]

std::Map to cache the textures

Definition at line [73](#) of file [ResourceTexture.h](#).

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

- [ResourceTexture.h](#)
- [ResourceTexture.cpp](#)

DsdEngine::SceneManager Class Reference

```
#include <SceneManager.h>
```

Public Member Functions

- [SceneManager](#) ([IMainGame](#) *game)
- [~SceneManager](#) ()
- [IScene](#) * [moveNext](#) ()
- [IScene](#) * [movePrevious](#) ()
- void [setScene](#) (int nextScene)
- void [addScene](#) ([IScene](#) *newScene)
- void [destroy](#) ()
- [IScene](#) * [getCurrentScene](#) ()

Protected Attributes

- [IMainGame](#) * [m_pGame](#)
- std::vector< [IScene](#) * > [m_pScenes](#)
- 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

DsdEngine::SceneManager::SceneManager ([IMainGame](#) * *game*)

Constructor.

Parameters:

<i>game</i> , the	IMainGame the manager belongs to
-------------------	--

Definition at line [10](#) of file [SceneManager.cpp](#).

DsdEngine::SceneManager::~SceneManager () [[inline](#)]

Deconstructor.

Definition at line [31](#) of file [SceneManager.h](#).

Member Function Documentation

void DsdEngine::SceneManager::addScene ([IScene](#) * *newScene*)

Add a Scene to the Scene Manager.

Parameters:

<i>newScene</i> , the	IScene to add to the Manager.
-----------------------	---

Definition at line [39](#) of file [SceneManager.cpp](#).

void DsdEngine::SceneManager::destroy ()

Destroy the [SceneManager](#) and all of its Scenes

Definition at line [46](#) of file [SceneManager.cpp](#).

[IScene](#) * DsdEngine::SceneManager::getCurrentScene ()

Get the Current Scene been managed

Returns:

[IScene](#), the current scene.

Definition at line [56](#) of file [SceneManager.cpp](#).

[IScene](#) * DsdEngine::SceneManager::moveNext ()

Move to Next scene in vector

Returns:

[IScene](#), the scene to move to.

Definition at line [16](#) of file [SceneManager.cpp](#).

[IScene](#) * DsdEngine::SceneManager::movePrevious ()

Move to Previous scene in vector

Returns:

[IScene](#), the scene to move to.

Definition at line [25](#) of file [SceneManager.cpp](#).

void DsdEngine::SceneManager::setScene (int *nextScene*)

Sets the current Scene

Parameters:

<i>nextScene</i> , the	current scene.
------------------------	----------------

Definition at line [34](#) of file [SceneManager.cpp](#).

Member Data Documentation

int DsdEngine::SceneManager::m_iCurrentSceneIndex [protected]

index for the current Scene

Definition at line [74](#) of file [SceneManager.h](#).

[IMainGame](#)* DsdEngine::SceneManager::m_pGame [protected]

Main Game which scenemanager belongs too

Definition at line [70](#) of file [SceneManager.h](#).

std::vector<[IScene](#)> DsdEngine::SceneManager::m_pScenes [protected]

Vector to hold the game scenes

Definition at line [72](#) of file [SceneManager.h](#).

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

- [SceneManager.h](#)
- [SceneManager.cpp](#)

DsdEngine::SFX Class Reference

```
#include <AudioManager.h>
```

Public Member Functions

- void [play](#) (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 DsdEngine::SFX::play (int *loop* = 0)

Play sound effect.

Parameters:

<i>int</i>	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* DsdEngine::SFX::m_Chunk [private]

Private Mix_Chunk Variable

Definition at line [33](#) of file [AudioManager.h](#).

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

- [AudioManager.h](#)
- [AudioManager.cpp](#)

DsdEngine::Size Class Reference

```
#include <EngineMath.h>
```

Public Member Functions

- [Size](#) ()
- [Size](#) (float w, float h)
- [Size](#) (const [Size](#) &s)
- [~Size](#) ()

Public Attributes

- float [w](#)
- float [h](#)

Detailed Description

[Size](#) is a class for creating a 2 point size variable

Definition at line [55](#) of file [EngineMath.h](#).

Constructor & Destructor Documentation

DsdEngine::Size::Size ()

Constructor Defaults values to 0 , 0

Definition at line [36](#) of file [EngineMath.cpp](#).

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

Constructor Set values on creation.

Parameters:

<i>w</i>	as a float argument
<i>h</i>	as a float argument

Definition at line [38](#) of file [EngineMath.cpp](#).

DsdEngine::Size::Size (const [Size](#) & s)

Constructor. Create a [Size](#) object with another [Size](#)

Parameters:

<i>s</i>	as a Size argument
----------	------------------------------------

Definition at line [40](#) of file [EngineMath.cpp](#).

DsdEngine::Size::~~Size ()

Deconstructor

Definition at line [45](#) of file [EngineMath.cpp](#).

Member Data Documentation

float DsdEngine::Size::h_

float value for height

Definition at line [87](#) of file [EngineMath.h](#).

float DsdEngine::Size::w_

float value for width

Definition at line [86](#) of file [EngineMath.h](#).

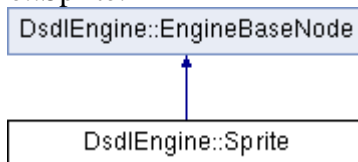
The documentation for this class was generated from the following files:

- [EngineMath.h](#)
- [EngineMath.cpp](#)

DsdEngine::Sprite Class Reference

```
#include <Sprite.h>
```

Inheritance diagram for DsdEngine::Sprite:



Public Member Functions

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

Additional Inherited Members

Detailed Description

[Sprite](#) file for creating in game sprites. Inherits from [EngineBaseNode](#)

Definition at line [13](#) of file [Sprite.h](#).

Constructor & Destructor Documentation

DsdEngine::Sprite::Sprite ()

Constructor

Definition at line [11](#) of file [Sprite.cpp](#).

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

Destructor

Definition at line [21](#) of file [Sprite.cpp](#).

Member Function Documentation

void DsdEngine::Sprite::create ([Vec2](#) spriteSize, [Vec2](#) position, std::string path)

Create basic sprite with one frame.

Parameters:

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

Definition at line [27](#) of file [Sprite.cpp](#).

void DsdEngine::Sprite::create ([Vec2](#) *spriteSize*, [Vec2](#) *position*, std::string *path*, int *numFrames*)

Create basic sprite with more then one frame.

Parameters:

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

Definition at line [43](#) of file [Sprite.cpp](#).

void DsdEngine::Sprite::createWithPhysics (b2World * *world*, [Vec2](#) *spriteSize*, [Vec2](#) *position*, std::string *path*, int *numFrames*, float *den*, float *fri*, bool *FixedRotation*)

Create basic sprite with one frame.

Parameters:

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

Definition at line [59](#) of file [Sprite.cpp](#).

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

Destroy the sprite.

Reimplemented from [DsdEngine::EngineBaseNode](#).

Definition at line [79](#) of file [Sprite.cpp](#).

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

Get the collision body of the sprite.

Returns:

b2Body, the sprites collision body

Definition at line [68](#) of file [Sprite.h](#).

void DsdEngine::Sprite::updateTexure ([Vec2](#) *spriteSize*, [Vec2](#) *position*, std::string *path*, int *numFrames*)

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

Parameters:

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

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

Definition at line [84](#) of file [Sprite.cpp](#).

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

- [Sprite.h](#)
- [Sprite.cpp](#)

DsdEngine::Vec2 Class Reference

```
#include <EngineMath.h>
```

Public Member Functions

- [Vec2](#) ()
- [Vec2](#) (float x, float y)
- [Vec2](#) (const [Vec2](#) &v)
- [~Vec2](#) ()

Public Attributes

- float [x](#)
- float [y](#)

Static Public Attributes

- static const [Vec2](#) [ZERO](#)

Detailed Description

[Vec2](#) is a class for creating a 2 point position variable

Definition at line [11](#) of file [EngineMath.h](#).

Constructor & Destructor Documentation

DsdEngine::Vec2::Vec2 ()

Constructor Defaults values to 0 , 0

Definition at line [11](#) of file [EngineMath.cpp](#).

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

Constructor Set values on creation.

Parameters:

<i>x</i>	as a float argument
<i>y</i>	as a float argument

Definition at line [15](#) of file [EngineMath.cpp](#).

DsdEngine::Vec2::Vec2 (const [Vec2](#) & v)

Constructor. Create a [Vec2](#) object with another [Vec2](#)

Parameters:

<i>v</i>	as a Vec2 argument
----------	------------------------------------

Definition at line [18](#) of file [EngineMath.cpp](#).

DsdEngine::Vec2::~Vec2 ()

Destructor

Definition at line [24](#) of file [EngineMath.cpp](#).

Member Data Documentation

float DsdEngine::Vec2::x_

float value for x position

Definition at line [46](#) of file [EngineMath.h](#).

float DsdEngine::Vec2::y_

float value for y position

Definition at line [47](#) of file [EngineMath.h](#).

const [Vec2](#) DsdEngine::Vec2::ZERO [static]

const position set to origin

Definition at line [44](#) of file [EngineMath.h](#).

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

- [EngineMath.h](#)
- [EngineMath.cpp](#)

DsdEngine::Window Class Reference

```
#include <Window.h>
```

Public Member Functions

- [Window](#) ()
- virtual [~Window](#) ()
- int [createWindow](#) (std::string windowNname, int screenWidth, int screenHeight, unsigned int flag)
- void [swapBuffer](#) ()
- int [getScreenWidth](#) ()
- int [getScreenHeight](#) ()
- SDL_Renderer * [getRenderer](#) ()
- void [destroy](#) ()

Private Attributes

- SDL_Window * [m_pSdlWindow](#)
- SDL_Renderer * [m_pSdlRenderer](#)
- SDL_Surface * [m_pScreenSurface](#)
- int [m_screenHeight](#)
- int [m_screenWidth](#)

Detailed Description

[Window](#) 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

DsdEngine::Window::Window ()

Constructor

Definition at line [10](#) of file [Window.cpp](#).

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

Destructor

Definition at line [14](#) of file [Window.cpp](#).

Member Function Documentation

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

Create the SDL_Window with the arguments passed in.

Parameters:

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

Returns:

int, 0 upon success,

Definition at line [19](#) of file [Window.cpp](#).

void DsdEngine::Window::destroy ()

Destroy the window

Definition at line [74](#) of file [Window.cpp](#).

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

Get the SDL_Renderer for the window.

Returns:

SDL_Renderer.

Definition at line [60](#) of file [Window.h](#).

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

Get the height of the window.

Returns:

int, height of the window,

Definition at line [54](#) of file [Window.h](#).

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

Get the width of the window.

Returns:

int, width of the window.

Definition at line [48](#) of file [Window.h](#).

void DsdEngine::Window::swapBuffer ()

Swap the OpenGL window buffer.

Definition at line [69](#) of file [Window.cpp](#).

Member Data Documentation**SDL_Surface* DsdEngine::Window::m_pScreenSurface [private]**

the SDL_Surface variable

Definition at line [70](#) of file [Window.h](#).

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

the SDL_Renderer variable

Definition at line [69](#) of file [Window.h](#).

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

the SDL_Window variable

Definition at line [68](#) of file [Window.h](#).

int DsdEngine::Window::m_screenHeight [private]

Definition at line [72](#) of file [Window.h](#).

int DsdEngine::Window::m_screenWidth [private]

int variables for screenHeight and screenWidth

Definition at line [72](#) of file [Window.h](#).

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

- [Window.h](#)
- [Window.cpp](#)

DsdEngine::XmlLocalStorage Class Reference

```
#include <XmlLocalStorage.h>
```

Public Member Functions

- void [setIntegerForKey](#) (int value, const char *key)
- void [setBoolForKey](#) (bool value, const char *key)
- void [setDoubleForKey](#) (double value, const char *key)
- void [setFloatForKey](#) (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 [deleteValueForKey](#) (const char *key)

Static Public Member Functions

- static [XmlLocalStorage](#) * [getInstance](#) ()

Protected Member Functions

- [XmlLocalStorage](#) ()
- virtual [~XmlLocalStorage](#) ()

Detailed Description

[XmlLocalStorage](#) 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

DsdEngine::XmlLocalStorage::XmlLocalStorage () [[inline](#)], [[protected](#)]

Constructor

Definition at line [108](#) of file [XmlLocalStorage.h](#).

virtual DsdEngine::XmlLocalStorage::~XmlLocalStorage () [[inline](#)], [[protected](#)], [[virtual](#)]

Destructor

Definition at line [113](#) of file [XmlLocalStorage.h](#).

Member Function Documentation

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

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

Definition at line [211](#) of file [XmlLocalStorage.cpp](#).

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

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

Parameters:

<i>key</i>	The key to get value.
------------	-----------------------

Returns:

bool value by *key*.

Definition at line [55](#) of file [XmlLocalStorage.cpp](#).

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

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

Parameters:

<i>key</i>	The key to get value.
------------	-----------------------

Returns:

double value by *key*.

Definition at line [84](#) of file [XmlLocalStorage.cpp](#).

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

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

Parameters:

<i>key</i>	The key to get value.
------------	-----------------------

Returns:

float value by *key*.

Definition at line [112](#) of file [XmlLocalStorage.cpp](#).

[XmlLocalStorage](#) * DsdEngine::XmlLocalStorage::getInstance () [static]

getInstance gets a static instance for the [XmlLocalStorage](#) class.

Returns:

[XmlLocalStorage](#) instance

Definition at line [17](#) of file [XmlLocalStorage.cpp](#).

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

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

Parameters:

<i>key</i>	The key to get value.
------------	-----------------------

Returns:

int value by *key*.

Definition at line [27](#) of file [XmlLocalStorage.cpp](#).

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

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

Parameters:

<i>key</i>	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:

<i>key</i>	The key to set.
<i>value,bool</i>	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:

<i>key</i>	The key to set.
<i>value,double</i>	value to be saved

Definition at line [176](#) of file [XmlLocalStorage.cpp](#).

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

Set a float value for a key in xml.

Parameters:

<i>key</i>	The key to set.
<i>value,float</i>	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:

<i>key</i>	The key to set.
<i>value,int</i>	value to be saved

Definition at line [144](#) of file [XmlLocalStorage.cpp](#).

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

Set a string value for a key in xml.

Parameters:

<i>key</i>	The key to set.
<i>value,string</i>	value to be saved

Definition at line [203](#) of file [XmlLocalStorage.cpp](#).

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

- [XmlLocalStorage.h](#)
- [XmlLocalStorage.cpp](#)

File Documentation

AudioManager.cpp File Reference

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

Namespaces

- [DsdEngine](#)

AudioManager.cpp

```
1 #include "AudioManager.h"
2 #include "FileIO.h"
3
4
5
6
7 namespace DsdEngine {
8
9     //init audio manager
10    void AudioManager::init() {
11        //init audio
12        if (Mix_Init(MIX_INIT_MP3 | MIX_INIT_OGG) == -1) {
13            SDL_Log("Mix Init error: %s ", Mix_GetError());
14        }
15        //open audio defaults
16        if (Mix_OpenAudio(MIX_DEFAULT_FREQUENCY, MIX_DEFAULT_FORMAT, 2, 2048) == -1) {
17            SDL_Log("Mix_OpenAudio error: %s ", Mix_GetError());
18        }
19
20        m_bisInitialized = true;
21    }
22
23    //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
37        m_bgAudioMap.clear();
38        m_sfxAudioMap.clear();
39
40        //Close and Quit Audio
41        Mix_CloseAudio();
42        Mix_Quit();
43    }
44
45    //Play sound effect
46    void SFX::play(int loop) {
47        if (Mix_PlayChannel(-1, m_Chunk, loop) == -1) {
48            if (Mix_PlayChannel(0, m_Chunk, loop) == -1) {
49                SDL_Log("Mix PlayChannel error: %s", Mix_GetError());
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
66     //Check if allready cached
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     }
77     //it is cached
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\_bgAudioMap.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 [DsdEngine::SFX](#)
- class [DsdEngine::Music](#)
- class [DsdEngine::AudioManager](#)

Namespaces

- [DsdEngine](#)

AudioManager.h

```
1 #ifndef _AUDIOMANAGER_
2 #define _AUDIOMANAGER_
3
4 #include "EngineDefines.h"
5 /*
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 DsdEngine {
12
13     class SFX {
14     public:
15
16         friend class AudioManager;
17
18         void play(int loop = 0);
19     private:
20         Mix_Chunk* m_Chunk;
21     };
22
23     class Music {
24     public:
25         friend class AudioManager;
26
27         void play(int loop = -1) { Mix_PlayMusic(m_Music, loop); };
28
29         void audioPauseBG() { Mix_PauseMusic(); };
30
31         void audioResumeBG() { Mix_ResumeMusic(); };
32
33         void audioStopBG() { Mix_HaltMusic(); };
34     private:
35         Mix_Music* m_Music;
36     };
37
38     class AudioManager {
39     public:
40         AudioManager() { init(); };
41
42         ~AudioManager() { destroy(); };
43
44         void init();
45
46         void destroy();
47     };
48 }
```

```
105
111     SFX loadSFX(std::string audioPath);
112
118     Music loadMusic(std::string audioPath);
119
120 private:
124     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

- [DsdEngine](#)

Button.cpp

```
1 #include "Button.h"
2 #include "IScene.h"
3
4 namespace DsdEngine {
5
6     Button::Button() {
7         setEngineNodeType(NodeType::LABEL);
8         m_eCurrentState = ButtonState::NORMAL;
9     }
10
11     Button::~Button() { destroy(); }
12
13     //create button as label
14     void Button::createTextButton(Vec2 pos, Size btnsize, std::string buttonText, std::string
fontPath, SDL_Color color, SDL_Color bgColor) {
15
16         m_size.y = btnsize.h;
17         m_size.x = buttonText.length() * btnsize.h;
18
19         m_position.x = pos.x;
20         m_position.y = pos.y;
21
22
23         m_labelText = buttonText;
24         m_textSize = btnsize.h;
25         m_textColor = color;
26         setAssetPath(fontPath);
27
28         m_label = new Label();
29         m_label->create(pos, m_labelText, m_textSize, m_textColor, fontPath);
30     }
31
32     //create button as sprite
33     void Button::createSpriteButton(Vec2 spriteSize, Vec2 position, std::string imagePath,
std::string name) {
34         m_size.x = spriteSize.x;
35         m_size.y = spriteSize.y;
36
37         setAssetPath(imagePath);
38
39         m_buttonName = name;
40
41         m_position.x = position.x;
42         m_position.y = position.y;
43         setPosition(position);
44
45         setEngineNodeType(NodeType::SPRITE);
46
47         m_numFrames = 1;
48         m_spriteBtn = new Sprite();
49         m_spriteBtn->create(m_size, m_position, imagePath, 1);
50     }
51
52     //Set State to Hovering
53     void Button::onMouseEnters() {
```

```

54     m_eCurrentState = ButtonState::HOVERING;
55     SDL_Log("Mouse over button");
56 }
57
58 //Set State Back to Normal
59 void Button::onMouseLeaves() {
60     m_eCurrentState = ButtonState::NORMAL;
61 }
62
63 //Set State to Pressed, Perform Action
64 void Button::onClicked() {
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     //check if mouse over button
72     if (e.type == SDL_MOUSEBUTTONDOWN || e.type == SDL_MOUSEBUTTONUP) {
73         int x, y;
74         SDL_GetMouseState(&x, &y);
75
76         //Check if mouse inside button area
77         bool inside = true;
78
79         //Check if mouse inside button
80         if (x < m_position.x) {
81             inside = false;
82         }
83         else if (x > m_position.x + m_size.y) {
84             inside = false;
85         }
86         else if (y < m_position.y) {
87             inside = false;
88         }
89         else if (y > m_position.y + m_size.x) {
90             inside = false;
91         }
92     }
93
94     //If mouse outside button
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;
106             case SDL_MOUSEBUTTONDOWN:
107                 onClicked();
108                 break;
109             //Touch down
110             case SDL_FINGERDOWN:
111                 onClicked();
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 Button::destroy() {  
128  
129     }  
130 }
```

Button.h File Reference

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

Classes

- class [DsdEngine::Button](#)

Namespaces

- [DsdEngine](#)

Button.h

```
1 #ifndef _BUTTON_
2 #define _BUTTON_
3
4 #include "<a href='\"#\"'>EngineBaseNode.h\">EngineBaseNode.h"
5
6 #include "<a href='\"#\"'>Label.h\">Label.h"
7 #include "<a href='\"#\"'>Sprite.h\">Sprite.h"
8
13 namespace DsdEngine {
14
19     class <a href='\"#\"'>Button : public EngineBaseNode {
20
21     public:
25         <a href='\"#\"'>Button();
26
30         virtual ~<a href='\"#\"'>Button();
31
35         void <a href='\"#\"'>destroy();
36
46         void <a href='\"#\"'>createTextButton(<a href='\"#\"'>Vec2 pos, <a href='\"#\"'>Size size, std::string buttonText, std::string
fontPath, SDL Color textColor, SDL Color bgColor);
47
55         void <a href='\"#\"'>createSpriteButton(<a href='\"#\"'>Vec2 spriteSize, <a href='\"#\"'>Vec2 position, std::string imagePath,
std::string name);
56
62         void <a href='\"#\"'>checkInput(SDL_Event& e);
63
64
70         std::string <a href='\"#\"'>getButtonName() { return m_buttonName; }
71
76         <a href='\"#\"'>ButtonState m_eCurrentState;
77     private:
78
83         void <a href='\"#\"'>onMouseEnters();
84
89         void <a href='\"#\"'>onMouseLeaves();
90
95         void <a href='\"#\"'>onClicked();
96
97
101         <a href='\"#\"'>Label* m_label;
102
106         <a href='\"#\"'>Sprite* m_spriteBtn;
107
111         std::string m_buttonName;
112
113     };
114 }
115
```



```
116 #endif // !_BUTTON_
```

CollisionShane.cpp File Reference

#include "CollisionShape.h"

Namespaces

- [DsdEngine](#)

CollisionShane.cpp

```
1 #include "CollisionShape.h"
2
3
4 namespace DsdEngine {
5
6     CollisionShape::CollisionShape() {
7         //Empty
8     }
9
10
11     CollisionShape::~CollisionShape() {
12         //Empty
13     }
14
15
16     void CollisionShape::init(b2World* world,
17         Vec2 position,
18         Vec2 dimensions,
19         float density,
20         float friction,
21         bool fixedRotation) {
22
23         m_dimensions = dimensions;
24
25         // Make the body
26         b2BodyDef bodyDef;
27         bodyDef.type = b2_dynamicBody;
28         bodyDef.position.Set(position.x, position.y);
29         bodyDef.fixedRotation = fixedRotation;
30         m_body = world->CreateBody(&bodyDef);
31
32
33         // Create the box
34         b2PolygonShape boxShape;
35         boxShape.SetAsBox(dimensions.x / 2.0f, (dimensions.y - dimensions.x) / 2.0f);
36
37         b2FixtureDef fixtureDef;
38         fixtureDef.shape = &boxShape;
39         fixtureDef.density = density;
40         fixtureDef.friction = friction;
41         m_fixtures[0] = m_body->CreateFixture(&fixtureDef);
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 [DsdEngine::CollisionShape](#)

Namespaces

- [DsdEngine](#)

CollisionShape.h

```
1 #ifndef __COLLISIONSHAPE__
2 #define __COLLISIONSHAPE__
3
4 #include "EngineDefines.h"
5
6 namespace DsdEngine {
7
8     class CollisionShape
9     {
10     public:
11
12         CollisionShape();
13
14         ~CollisionShape();
15
16         void init(b2World* world,
17                 Vec2 position,
18                 Vec2 dimensions,
19                 float density,
20                 float friction,
21                 bool fixedRotation);
22
23         void destroy(b2World* world);
24
25         b2Body* getBody() const { return m_body; }
26
27         b2Fixture* getFixture(int index) const { return m_fixtures[index]; }
28
29         const Vec2 getDimensions() const { return m_dimensions; }
30
31     protected:
32         b2Body* m_body = nullptr;
33
34         b2Fixture* m_fixtures[1];
35
36         Vec2 m_dimensions;
37     };
38 }
39
40 #endif // !__COLLISIONSHAPE__
```

DsdlEngine.cpp File Reference

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

Namespaces

- [DsdlEngine](#)

Functions

- int [DsdlEngine::init](#) ()

DsdlEngine.cpp

```
1  
2 #include "EngineDefines.h"  
3 #include "DsdlEngine.h"  
4  
5 namespace DsdlEngine{  
6  
7     int init() {  
8         //Initialize SDL  
9         SDL_Init(SDL_INIT_AUDIO | SDL_INIT_EVENTS | SDL_INIT_TIMER | SDL_INIT_VIDEO);  
10  
11         SDL_Log("Log Print SDL Finised Init!\n");  
12  
13         //Initialize TTF  
14         TTF_Init();  
15  
16         SDL_GL_SetAttribute(SDL_GL_ACCELERATED_VISUAL, 1);  
17  
18         return 0;  
19     }  
20 }
```

DsdIEngine.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

- [DsdIEngine](#)

Functions

- int [DsdIEngine::init](#) ()

DsdIEngine.h

```
1 #ifndef DSDLENGINE
2
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"
24
25 #include "Window.h"
```

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

EngineBaseNode.cpp File Reference

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

Namespaces

- [DsdEngine](#)

EngineBaseNode.cpp

```
1
2 #include "EngineBaseNode.h"
3 #include "FileIO.h"
4
5 namespace DsdEngine {
6
7     //Constructor
8     EngineBaseNode::EngineBaseNode() {
9
10         m_engineTexture = NULL;
11         setEngineNodeType(NodeType::BASENODE);
12         m_frame = 0;
13         m_numFrames = 1;
14         m_opacity = 255;
15         m_objectBoundingBox = new SDL_Rect();
16
17         updateTextureInfo = false;
18     }
19
20     //Deconstructor
21     EngineBaseNode::~EngineBaseNode() {
22         destroy();
23     }
24
25     //Render Node by type
26     void EngineBaseNode::render(SDL_Renderer* r) {
27         if (nodeType == NodeType::SPRITE) {
28             m_currentFrame = &m_gSpriteClips[m_frame / m_numFrames];
29             m_engineTexture->setAlpha(m_opacity);
30
31             //Draw Bounding Box
32             //SDL_SetRenderDrawColor(r, 0, 255, 255, 255);
33             //SDL_RenderDrawRect(r, m_objectBoundingBox);
34
35             //render texture
36             m_engineTexture->render(m_position, m_size, r, m_currentFrame);
37             ++m_frame;
38
39             if (m_frame / m_numFrames >= m_numFrames) {
40                 m_frame = 0;
41             }
42         }
43         else if (nodeType == NodeType::LABEL) {
44             m_engineTexture->render(m_position, m_size, r);
45         }
46         else if (nodeType == NodeType::BUTTON) {
47             m_engineTexture->render(m_position, m_size, r);
48         }
49         else if (nodeType == NodeType::PARTICLE) {
50
51         }
52     }
53 }
54
55 }
```

```

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("Failed to load sprite");
69
70         else {
71
72             int temp = 0;
73
74             for (int i = 0; i < m_numFrames; i++) {
75                 m_gSpriteClips[i].x = temp;
76                 m_gSpriteClips[i].y = 0;
77                 m_gSpriteClips[i].w = m_size.x;
78                 m_gSpriteClips[i].h = m_size.y;
79
80                 temp += m_size.x;
81             }
82             m_objectBoundingBox->x = m_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             TTF_Init();
94         }
95
96         std::string temp;
97
98         temp = FileIO::getInstance()->getWritablePath() + m_assetPath;
99
100         //Using getWritablePath
101         FileIO::getInstance()->getWritablePath() + m_assetPath;
102
103         //Check if font in cache
104         auto it = m_FontMap.find(temp);
105
106         // if not load and create texture
107         if (it == m_FontMap.end()) {
108
109             //open font
110             m_font = TTF_OpenFont(temp.c_str(), m_textSize);
111             if (m_font == NULL) {
112                 SDL_Log("TTF_OpenFont Error : %s", TTF_GetError());
113             }
114
115             m_engineTexture->loadTTF(m_labelText, m_textColor, m_font, r);
116
117             m_FontMap[temp] = m_font;
118         }
119         else { //create texture
120             m_font = it->second;
121             m_engineTexture->loadTTF(m_labelText, m_textColor, m_font, r);
122         }
123
124         return true;
125     }
126     else
127         return false;
128 }
129

```



```

130
131 void EngineBaseNode::setBoundingBox(Vec2 pos, Vec2 size) {
132     m_objectBoundingBox = new SDL_Rect();
133     m_objectBoundingBox->x = pos.x ;
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
144 void EngineBaseNode::setOpacity(int op) {
145     if (op > 255 || op < 0) {
146         SDL_Log("Invalid opacity value passed in.");
147         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;
161     m_CollisionShape = nullptr;
162     m_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;
174     collisionbox.y = shape->getBody()->GetPosition().y;
175     collisionbox.w = shape->getDimensions().x ;
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 [DsdEngine::EngineBaseNode](#)

Namespaces

- [DsdEngine](#)

EngineBaseNode.h

```
1 #ifndef _ENGINEBASENODE_
2 #define _ENGINEBASENODE_
3
4 #include "EngineDefines.h"
5 #include "ResourceTexture.h"
6 #include "CollisionShape.h"
7
8
14 namespace DsdEngine {
18     class EngineBaseNode {
19     public:
23         EngineBaseNode();
24
28         virtual ~EngineBaseNode();
29
33         virtual void destroy();
34
38         virtual void cleanup();
39
45         bool load(SDL_Renderer* r);
46
51         void render(SDL_Renderer* r);
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
71         void setPositionX(int x) { m_position.x = x; }
72
77         void setPositionY(int y) { m_position.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
112        //Rotate Node
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 setAssetPath(std::string path) { m assetPath = path; }
139
144 std::string getAssetsPath() { return m assetPath; }
145
150 NodeType getNodeType() { return nodeType; }
151
156 void setEngineNodeType(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
180 SDL_Rect* getBoundingBox() { return m objectBoundingBox; }
181
187 void setBoundingBox(Vec2 pos, Vec2 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;
208 ResourceTexture* m engineTexture;
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;
223 // For labels
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
233 CollisionShape* m CollisionShape;
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 "EngineMaster.h"
#include <Box2D\Box2D.h>
```

Namespaces

- [DsdEngine](#)

Macros

- #define [USING_NS_DSDL](#) using namespace DsdEngine
- #define [NS_DSDL_START](#) namespace DsdEngine{
- #define [NS_DSDL_END](#) }
- #define [DEFAULT_ROOT_NAME](#) "DefaultRoot"
- #define [XML_FILE](#) "Default.xml"
- #define [TOTAL_PARTICLES](#) 30
- #define [METRESTOPIXELS](#) 30
- *Box2D scaling defines.*
- #define [PIXELSTOMETRES](#) 1/30.0f
- #define [RADTODEG](#) (-180/3.1415926536f)
- #define [DEGTORAD](#) -0.0174532925199432957f
- #define [GRAVITYSCALE](#) 9.0f

Typedefs

- typedef SDL_TimerID [DsdEngine::CallBackTimer](#)
- typedef SDL_TimerCallback [DsdEngine::CallBack](#)

Enumerations

- enum [DsdEngine::NodeType](#) { [DsdEngine::NodeType::BASENODE](#), [DsdEngine::NodeType::SPRITE](#), [DsdEngine::NodeType::LABEL](#), [DsdEngine::NodeType::BUTTON](#), [DsdEngine::NodeType::PARTICLE](#) }
- enum [DsdEngine::ButtonState](#) { [DsdEngine::ButtonState::NORMAL](#), [DsdEngine::ButtonState::PRESSED](#), [DsdEngine::ButtonState::HOVERING](#) }
- enum [DsdEngine::ButtonType](#) { [DsdEngine::ButtonType::LABEL_BTN](#), [DsdEngine::ButtonType::SPRITE_BTN](#) }
- enum [DsdEngine::LableType](#) { [DsdEngine::LableType::LABEL_STATIC](#), [DsdEngine::LableType::LABEL_DYNAMIC](#) }

Macro Definition Documentation

#define DEFAULT_ROOT_NAME "DefaultRoot"

Definition at line [77](#) of file [EngineDefines.h](#).

#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 [85](#) of file [EngineDefines.h](#).

#define NS_DSDL_END }

Definition at line [32](#) of file [EngineDefines.h](#).

#define NS_DSDL_START namespace DsdEngine{

Definition at line [31](#) of file [EngineDefines.h](#).

#define PIXELSTOMETRES 1/30.0f

Definition at line [86](#) of file [EngineDefines.h](#).

#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 DsdEngine

Definition at line [30](#) of file [EngineDefines.h](#).

#define XML_FILE "Default.xml"

Definition at line [78](#) of file [EngineDefines.h](#).

EngineDefines.h

```
1
2 #ifndef ENGINEDEFINES
3 #define _ENGINEDEFINES_
4
5
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>
27
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 {
35     enum class NodeType {
36         BASENODE,
37         SPRITE,
38         LABEL,
39         BUTON,
40         PARTICLE
41     };
42
43     enum class ButtonState {
44         NORMAL,
45         PRESSED,
46         HOVERING
47     };
48
49     enum class ButtonType {
50         LABEL_BTN,
51         SPRITE_BTN
52     };
53
54     enum class LableType {
55         LABEL_STATIC,
56         LABEL_DYNAMIC
57     };
58
59     typedef SDL_TimerID CallbackTimer;
60     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
83
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

- [DsdEngine](#)

EngineError.cpp

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


EngineError.h File Reference

#include "EngineDefines.h"

Namespaces

- [DsdEngine](#)

Macros

- #define [DEBUG_DSDL](#) 1
- #define [DEBUG_MSG](#)(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

```
1
2 #ifndef _ENGINEERROR_
3 #define _ENGINEERROR_
4
5 #include "<a href='\"#\"'>EngineDefines.h</a>"
6
7 namespace <a href='\"#\"'>DsdEngine</a>{
8
9     #define DEBUG_DSDL 1
10    #if defined DEBUG_DSDL
11        #if (DEBUG_DSDL == 1)
12            #define DEBUG_MSG(x) (std::cout << (x) <<std::endl)
13        #else
14            #define DEBUG_MSG(x)
15        #endif
16    #else
17        #define DEBUG_MSG(x)
18    #endif
19 }
20
21
22
23 #endif
```

EngineMaster.cpp File Reference

```
#include "EngineMaster.h"
```

Namespaces

- [DsdEngine](#)

Variables

- static EngineMaster * [DsdEngine::Instance](#) = nullptr

EngineMaster.cpp

```
1 #include "EngineMaster.h"
2
3 namespace DsdEngine {
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 [DsdEngine::EngineMaster](#)

Namespaces

- [DsdEngine](#)

EngineMaster.h

```
1 #ifndef _ENGINEMASTER_
2 #define _ENGINEMASTER_
3
4 #include "EngineDefines.h"
5
10 namespace DsdEngine{
14     class EngineMaster{
15     public:
16
21         static EngineMaster* getInstance();
22
23     protected:
27         EngineMaster(){};
28
32         virtual ~EngineMaster(){};
33
34     private:
35
36     };
37 }
38
39
40 #endif // !_ENGINEMASTER_
```

EngineMath.cpp File Reference

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

Namespaces

- [DsdEngine](#)

EngineMath.cpp

```
1 #include "EngineMath.h"
2 #include "XmlLocalStorage.h"
7 namespace DsdEngine{
8
9
10 //Defaults to position of (0 , 0)
11 Vec2::Vec2() : x (0), y (0){
12 }
13
14 //Set position to values passed in (x , y)
15 Vec2::Vec2(float x, float y) : x(x), y(y){
16 }
17
18 Vec2::Vec2(const Vec2& v){
19     this->x = v.x ;
20     this->y = v.y ;
21 }
22
23
24 Vec2::~Vec2() {}
25
26 /*
27     SDL Window Coordintes
28     origin is top left corner
29 */
30 const Vec2 Vec2::ZERO(0, 0);
31 }
32
33 namespace DsdEngine{
34
35
36 Size::Size() : w_(0), h_(0){}
37
38 Size::Size(float w, float h) : w(w), h(h){}
39
40 Size::Size(const Size& s){
41     this->h = s.h ;
42     this->w = s.w ;
43 }
44
45 Size::~Size() {}
46
47 }
```

EngineMath.h File Reference

Classes

- class [DsdEngine::Vec2](#)
- class [DsdEngine::Size](#)

Namespaces

- [DsdEngine](#)

EngineMath.h

```
1 #ifndef _ENGINEMATH_
2 #define _ENGINEMATH_
3
4
5
6
7 namespace DsdEngine{
11     class Vec2{
12
13     public:
14
15
16
17
18         Vec2();
19
20
21
22         Vec2(float x, float y);
23
24
25
26         Vec2(const Vec2& v);
27
28
29         ~Vec2();
30
31
32         static const Vec2 ZERO;
33
34         float x;
35         float y;
36     };
37
38     class Size{
39     public:
40
41
42
43         Size();
44
45
46         Size(float w, float h);
47
48
49         Size(const Size& s);
50
51         ~Size();
52
53         float w;
54         float h;
55     };
56 }
57
58 #endif
```

FileIO.cpp File Reference

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

Namespaces

- [DsdEngine](#)

Variables

- static FileIO * [DsdEngine::Instance](#) = nullptr

FileIO.cpp

```
1 #include "FileIO.h"
2 #include <fstream>
3
4 namespace DsdEngine{
5
6     //Create As Singleton
7     static FileIO* Instance = nullptr;
8
9     FileIO* FileIO::getInstance() {
10         if (!Instance){
11             Instance = new (std::nothrow) FileIO();
12         }
13         return Instance;
14     }
15
16     std::string FileIO::getSuitableFOpen(const std::string& filenameUtf8) const{
17         return filenameUtf8;
18     }
19
20     /*
21      * Get Path to file
22      *
23      * if defs here for different platfoms as windows needs to find assets in root folder which
24      * i have created
25      * but android needs to find assets in the jni/assets folder. android is allready set up
26      * to go look in this folder
27      * there for all that was needed was the name and in the windows platfom i add on path to
28      * the assets folder so it just has to look for name of file
29      *
30      * this will need to be done to each asset type loding function eg. audio, fonts, images
31      */
32     std::string FileIO::getWritablePath() {
33         //For Windows
34         #ifdef WIN32
35             m\_path;
36         #endif
37
38         //For Android
39         #ifdef __ANDROID__
40             m\_path = "";
41         #endif
42         return m\_path;
43     }
44
45     //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");
57         if (file != nullptr) {
58
59             //Get length of file
60             size_t file_length = SDL_RWseek(file, 0, SEEK_END);
61             (*doc_contents) = new char[file_length + 1];
62             SDL_RWseek(file, 0, SEEK_SET);
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
81         SDL_RWops *file = SDL_RWFromFile(filepath, "w");
82         if (file != nullptr) {
83             //Length of data to write
84             size_t len = SDL_strlen(*doc_contents);
85
86             //Write the data
87             SDL_RWwrite(file, *doc_contents, 1, len);
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
98     XMLNode* FileIO::getXMLNodeForKey(const char* pKey, XMLNode** rootNode,
XMLDocument** doc) {
99
100         XMLNode* 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) {
127             SDL_Log("read root node error ");
128         }
129
130         // find the node
131         curNode = (*rootNode)->FirstChildElement();
132         while (curNode != nullptr)
133         {
134             const char* nodeName = curNode->Value();
135             if (!strcmp(nodeName, pKey)) {
136                 break;
137             }
138             curNode = curNode->NextSiblingElement();
139         }
140     }
141     else {
142         SDL_Log("Could not load doc: ");
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;
159     XMLDocument* doc;
160     XMLElement* node;
161     XMLPrinter printer;
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 {
180             XMLText* content = doc->NewText(value);
181             node->LinkEndChild(content);
182         }
183     } //Create new node and set value
184     else {
185         if (rootNode) {
186             XMLElement* temp = doc->NewElement(key);
187             rootNode->LinkEndChild(temp);
188             XMLText* content = doc->NewText(value);
189             temp->LinkEndChild(content);
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();
213         if (nullptr == doc) {
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         XElement *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
232         path = getWritablePath() + "Default.xml";
233
234
235         bRet = XML_SUCCESS ==
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 [DsdEngine::FileIO](#)

Namespaces

- [DsdEngine](#)

FileIO.h

```
1 #ifndef _FILEIO_
2 #define _FILEIO_
3
4 #include "EngineDefines.h"
5 #include <sys/stat.h>
6 #include "../dependencies/tinyxml/tinyxml2.h"
11 namespace DsdEngine{
12
14     using namespace tinyxml2;
15     using namespace std;
16
21     class FileIO{
22
23     public:
28         static FileIO* getInstance();
29
35         std::string getSuitableFOpen(const std::string& filenameUtf8) const;
36
41         std::string getWritablePath();
42
47         void setAssetsPath( std::string assetsPath) { m_path = assetsPath; }
48
49
54         std::string getFileToOpen() { return m_fileName; }
55
60         void setFileToOpen(std::string file) { m_fileName = file; }
61
68         bool loadDocument(const char* filepath, char** doc_contents);
69
76         bool writeDocument(const char* filepath, const char** doc contents);
77
85         XMLNode* getXMLNodeForKey(const char* pKey, XMLNode** 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 ~FileIO(){};
110
111     private:
112
113         std::string m_path;
114         std::string m\_fileName;
116     };
117 }
```

```
118  
119 #endif // !_FILEIO_
```

Gui.cpp File Reference

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

Namespaces

- [DsdEngine](#)

Gui.cpp

```
1 #include "Gui.h"
2 #include "Button.h"
3 #include "Window.h"
4 #include "Label.h"
5
6 /*
7   File: Gui
8   Author: Derek O'Brien
9   Description: GUI Layer template for creating an a UI Layer. Inherits from layer
10 */
11 namespace DsdEngine{
12
13   //Constructor
14   DsdGui::DsdGui() {
15     //Empty
16   }
17
18   //Destructor
19   DsdGui::~DsdGui() {
20     destroy();
21   }
22
23   //Add Button to GUI
24   void DsdGui::addButton(ButtonType type, std::string name, Vec2 pos, Vec2 size, std::string
path, SDL\_Color color, SDL\_Color bgColor, const char* text) {
25
26     m\_btn = new 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);
30       m\_btn->setPosition(pos);
31     }
32
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
40     //Add button to gui elements array
41     GUIElements.push_back(m\_btn);
42     layerNodes.push_back(m\_btn);
43   }
44
45   //Add Label to Gui Layer
46   void DsdGui::addLabel(LabelType type, Vec2 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, LabelType type) {
58     label->setType(type);
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) {
69     //Loop and Check each button for input
70     for (size_t i = 0; i < GUIElements.size(); i++){
71         GUIElements.at(i)->checkInput(e);
72     }
73 }
74
75 //Destroy
76 void DsdlGui::destroy() {
77
78     if (layerNodes.size() > 0) {
79
80         for (size_t i = 0; i < layerNodes.size(); i++) {
81             layerNodes.erase(std::remove(layerNodes.begin(), layerNodes.end(),
layerNodes[i]), layerNodes.end());
82
83             layerNodes[i]->destroy();
84
85         }
86         // layerNodes.clear();
87     }
88     // GUIElements.clear();
89 }
90 }
91
92

```

Gui.h File Reference

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

Classes

- class [DsdEngine::DsdGui](#)

Namespaces

- [DsdEngine](#)

Gui.h

```
1 #ifndef _GUI_
2 #define _GUI_
3
4 #include "EngineDefines.h"
5 #include "Layer.h"
6 #include "IScene.h"
7
11 namespace DsdEngine{
12
13     //Forward Decalare Classes
14     class Button;
15     class Label;
16
20     class DsdGui : public Layer{
21     public:
25         DsdGui();
26
30         virtual ~DsdGui();
31
43         void addButton(ButtonType type, std::string name, Vec2 pos, Vec2 size, std::string path,
SDL Color color, SDL Color bgColor, const char* text = NULL);
44
54         void addLabel(LableType type, Vec2 pos, std::string text, int fontSize, SDL_Color color,
std::string fontFilePath);
55
61         void addPreDefineLabel(Label* label, LableType type);
62
66         void setGUIPos();
67
72         void onSDLEvent(SDL Event& e);
73
77         void destroy();
78
79         std::vector<Button*> GUIElements;
85         Button* getButton() { return m_btn; }
86
87     protected:
88         //Variables
89         Label* m_label;
90         Button* m_btn;
91     };
92 }
93
94 #endif
```

IMainGame.cpp File Reference

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

Namespaces

- [DsdEngine](#)

Functions

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

IMainGame.cpp

```
1 #include "IMainGame.h"  
2  
3  
4 #include "SceneManager.h"  
5 #include "IScene.h"  
6  
7  
8 namespace DsdEngine {  
9  
10     /*  
11     *Added template version of make_unique as Ndk did not support it in its version of STL  
12     *Error was make_unique not part of std::  
13     *After research this was the easiest solution to solve error  
14     *Ndk-build now builds apk as of 26/01/2016  
15     */  
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  
21     //Constructor  
22     IMainGame::IMainGame() {  
23         m\_pSceneManager = DsdEngine::make\_unique<SceneManager>(this);  
24     }  
25  
26     IMainGame::~IMainGame() {  
27         //Empty  
28     }  
29  
30  
31     /*  
32     Main Game Loop  
33  
34     */  
35     void IMainGame::mainLoop() {  
36         if (!init()) return;  
37         FpsLimiter fpsLimit;  
38         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 IMainGame::run() {
59     mainLoop();
60 }
61
62
63 /*
64     Main Inputmanager control
65 */
66 void IMainGame::onSDL_Event(SDL_Event& evnt) {
67     m_InputManager.update();
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;
77             case SDL_MOUSEMOTION:
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;
86             case SDL_MOUSEBUTTONDOWN:
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;
95             case SDL_FINGERMOTION:
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;
113     windowTitle = windowName;
114     windowFlag = flag;
115
116     mainAssetsPath = path;
117     //Set windowss asset path
118 #ifdef WIN32
119     FileIO::getInstance()->setAssetsPath(mainAssetsPath);

```



```

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()) {
140             SDL_Log("InitSystems Failed : Window not created");
141             return false;
142         }
143
144         //Add all Scenes
145         addScenes();
146
147         //Load up First Scene
148         m_pCurrentRunning = m_pSceneManager->getCurrentScene();
149         m_pCurrentRunning->setSceneRunning();
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);
171         m_pGameRenderer = m_Window.getRenderer();
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) {
183             switch (m_pCurrentRunning->getSceneState()) {
184                 case SceneState::RUNNING:
185                     m_pCurrentRunning->updateScene();
186                     break;
187                 case SceneState::CHANGE_NEXT:
188                     m_pCurrentRunning->onExitScene();
189                     m_pCurrentRunning = m_pSceneManager->moveNext();
190                     if (m_pCurrentRunning) {

```

```

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 SceneState::CHANGE_PREVIOUS:
200         m_pCurrentRunning->onExitScene();
201         m_pCurrentRunning = m_pSceneManager->movePrevious();
202         if (m_pCurrentRunning) {
203             m_pCurrentRunning->setSceneRunning();
204             m_pCurrentRunning->onEntryScene();
205             //Load all scene Children nodes for previous scene
206             for (size_t i = 0; i < m_pCurrentRunning->sceneLayers.size(); i++) {
207                 m_pCurrentRunning->loadScene(m_pGameRenderer);
208             }
209         }
210     break;
211     case SceneState::EXIT_APP:
212         exitGame();
213     break;
214     default:
215         break;
216 }
217 }
218 else {
219     exitGame();
220 }
221 }
222
223 /*
224  Render all Scene nodes to screen
225 */
226
227 void IMainGame::draw() {
228     if (m_pCurrentRunning && m_pCurrentRunning->getSceneState() == SceneState::RUNNING) {
229         SDL_RenderClear(m_pGameRenderer);
230
231         //for running scene render each node that is in the child vector
232         for (size_t i = 0; i < m_pCurrentRunning->sceneLayers.size(); i++) {
233             m_pCurrentRunning->drawScene(m_pGameRenderer);
234         }
235         SDL_RenderPresent(m_pGameRenderer);
236     }
237 }
238
239 /*
240  Exit Game
241 */
242 void IMainGame::exitGame() {
243     m_pCurrentRunning->onExitScene();
244     if (m_pSceneManager) {
245         m_pSceneManager->destroy();
246         m_pSceneManager.reset();
247     }
248     m_bIsRunning = false;
249 }
250
251 }
252
253 }
254 }

```

IMainGame.h File Reference

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

Classes

- class [DsdEngine::IMainGame](#)

Namespaces

- [DsdEngine](#)

IMainGame.h

```
1 #ifndef  MAINGAME
2 #define  _MAINGAME_
3
4 #include "EngineDefines.h"
5 #include "DsdEngine.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"
12
13 namespace DsdEngine{
14
15     class SceneManager;
16     class IScene;
17
18     class IMainGame{
19     public:
20         IMainGame() ;
21
22         virtual ~IMainGame() ;
23
24         void run() ;
25
26         void setupWindow(int w, int h, std::string windowName, std::string path, int flag);
27
28         void setFps(float fps){ m\_fFps = fps; }
29
30         virtual void onInit() = 0;
31
32         virtual void addScenes() = 0;
33
34         virtual void onExit() = 0;
35
36         void onSDLEvent(SDL_Event& evnt);
37
38         void setPaused() { m\_bIsPaused = true; }
39
40         void setRunning() { m\_bIsPaused = false; m\_bIsRunning = true; }
41
42         bool checkPaused() { return m\_bIsPaused; }
43
44     };
45
46 }
```

```

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

```

InputManager.cpp File Reference

```
#include "InputManager.h"
```

Namespaces

- [DsdEngine](#)

InputManager.cpp

```
1 #include "InputManager.h"
2
3 namespace DsdEngine{
4
5
6     InputManager::InputManager() {
7         swipedown = false;
8         swipeup = false;
9         swipeleft = false;
10        swiperight = false;
11    }
12
13    InputManager::~InputManager() {
14        //Empty
15    }
16
17
18    void InputManager::update() {
19        // Loop through _keyMap using a for each loop, and copy it over to _previousKeyMap
20        for (auto& it : keyMap) {
21            _previousKeyMap[it.first] = it.second;
22        }
23    }
24
25    void InputManager::pressKey(unsigned int keyID) {
26        // if keyID doesn't already exist in keyMap, it will get added
27        keyMap[keyID] = true;
28    }
29
30    void InputManager::releaseKey(unsigned int keyID) {
31        keyMap[keyID] = false;
32    }
33
34    void InputManager::setMouseCoords(float x, float y) {
35
36    }
37
38
39    bool InputManager::isKeyDown(unsigned int keyID) {
40        // We dont want to use the associative array approach here
41        // because we don't want to create a key if it doesnt exist.
42        // So we do it manually
43        auto it = keyMap.find(keyID);
44        if (it != keyMap.end()) {
45            // Found the key
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.
57        // So we do it manually
```

```

58     auto it = previousKeyMap.find(keyID);
59     if (it != previousKeyMap.end()) {
60         // Found the key
61         return it->second;
62     }
63     else {
64         // Didn't find the key
65         return false;
66     }
67 }
68
69 bool InputManager::isKeyPressed(unsigned int keyID) {
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);
73         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);
107         startY = ((float)evnt.tfinger.y);
108         fingerDown = true;
109
110         if (evnt.type == SDL_FINGERUP) {
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
122             SDL_Log("SWIPE DOWN---Start : %f ----- End : %f", startX, endX);
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
132             swipeleft = true;
133             SDL_Log("SWIPE LEFT---Start : %f ----- End : %f", startY, endY);
134         }
135         else if (startY < endY) {
136             //swipe right
137             swiperight = true;
138             SDL_Log("SWIPE RIGHT---Start : %f ----- End : %f", startY, endY);
139         }
140         else if (startX == endX) {
141             SDL_Log("Only touch happened not swipe");
142         }
143     }
144     return true;
145 }
146
147
148
149 bool InputManager::isSwipeUp() {
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 [DsdEngine::InputManager](#)

Namespaces

- [DsdEngine](#)

InputManager.h

```
1 #ifndef _INPUTMANAGER_
2 #define _INPUTMANAGER_
3
4 #include "EngineDefines.h"
11 namespace DsdEngine{
15     class InputManager{
16     public:
20         InputManager();
21
25         ~InputManager();
26
30         void update();
31
36         void pressKey(unsigned int keyID);
37
42         void releaseKey(unsigned int keyID);
43
49         void setMouseCoords(float x, float y);
50
56         bool isKeyDown(unsigned int keyID);
57
63         bool isKeyPressed(unsigned int keyID);
64
70         bool isKeyReleased(unsigned int KeyID);
71
77         bool isTouch(unsigned int keyID);
78
84         bool isSwipe(SDL_Event& evnt);
85
90         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
119        std::unordered_map<unsigned int, bool> keyMap;
120        std::unordered_map<unsigned int, bool> previousKeyMap;
122        bool swipeup, swipedown, swipeleft, swiperight;
123        bool fingerDown, fingerUp;
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 [DsdEngine::IScene](#)

Namespaces

- [DsdEngine](#)

Macros

- #define [SCENE_INDEX_NO_SCENE](#) -1

Enumerations

- enum [DsdEngine::SceneState](#) { [DsdEngine::SceneState::NONE](#), [DsdEngine::SceneState::RUNNING](#), [DsdEngine::SceneState::EXIT_APP](#), [DsdEngine::SceneState::CHANGE_NEXT](#), [DsdEngine::SceneState::CHANGE_PREVIOUS](#) }

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_
3
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
11 namespace DsdEngine {
12
13     //forward declaration of class
14     class IMainGame;
15
16     enum class SceneState {
17         NONE,
18         RUNNING,
19         EXIT\_APP,
20         CHANGE\_NEXT,
21         CHANGE\_PREVIOUS
22     };
23 }
```

```

31
35 class IScene {
36 public:
40     IScene() {
41         //Empty
42     };
43
47     virtual ~IScene() {
48         //Empty
49     };
50
55     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
110     void setParentGame(IMainGame* game) { m_game = game; }
111
115     virtual void onInput();
116
117     std::vector<Layer*> sceneLayers;
123     void addLayerToScene(Layer* layer) {
124         sceneLayers.push_back(layer);
125     }
126
131     void loadScene(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++) {
143             sceneLayers.at(i)->drawNodes(r);
144         }
145     }
146
147 protected:
149     friend class SceneManager;
150     friend class InputManager;
151
152     SceneState m_eCurrentState = SceneState::NONE;
154     IMainGame* m_game = nullptr;
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

- [DsdEngine](#)

Label.cpp

```
1 #include "Label.h"
2 /*
3     Base Label Class
4     author: Derek O Brien
5     Description: label class for creating labels. inherits for EngineBaseNode
6 */
7
8 namespace DsdEngine{
9
10     //Constructor
11     Label::Label() {
12         setEngineNodeType(NodeType::LABEL);
13     }
14
15     //Deconstructor
16     Label::~Label() {
17         destroy();
18     }
19
20     //Create Label
21     void Label::create(Vec2 pos, std::string text, int txtsize, SDL_Color color, std::string
fontFilePath) {
22
23         m_labelText = text;
24         m_textSize = txtsize;
25         m_textColor = color;
26
27         setAssetPath(fontFilePath);
28
29         m_position.x = pos.x;
30         m_position.y = pos.y;
31     }
32
33     //Destroy Label
34     void Label::destroy() {
35         EngineBaseNode::destroy();
36     }
37
38     //Cleanup Label
39     void Label::cleanup() {
40         EngineBaseNode::cleanup();
41     }
42 }
```

Label.h File Reference

#include "EngineBaseNode.h"

Classes

- class [DsdEngine::Label](#)

Namespaces

- [DsdEngine](#)

Label.h

```
1 #ifndef _LABEL_
2 #define _LABEL_
3
4 #include "EngineBaseNode.h"
5
6 namespace DsdEngine{
7
8     class Label : public EngineBaseNode{
9     public:
10         /**
11          *   Constructor
12          */
13         Label();
14
15         virtual ~Label();
16
17         void create(Vec2 pos, std::string text, int fontSize, SDL_Color color, std::string
fontFilePath);
18
19         void setType(LabelType type) { m_labelType = type; };
20
21         const int getType() { return (int)m_labelType; }
22
23         void destroy();
24
25         void cleanup();
26
27     protected:
28         LabelType m_labelType;
29     };
30 }
31 #endif // !_LABEL_
```

Layer.cpp File Reference

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

Namespaces

- [DsdEngine](#)

Layer.cpp

```
1 #include "Layer.h"
2 #include "Sprite.h"
3
4 /*
5     Base Layer Class
6     author: Derek O Brien
7     Description: Layer base class for all layers in game.
8 */
9
10 namespace DsdEngine {
11
12     //Constructor
13     Layer::Layer() {
14         //Empty
15         layerNodes.reserve(20);
16     }
17
18     //Destructor
19     Layer::~Layer() {
20         destroy();
21     }
22
23     //Destroy layer nodes and cleanup
24     void Layer::destroy() {
25         for (size_t i = 0; i < layerNodes.size(); i++) {
26             layerNodes.erase(std::remove(layerNodes.begin(), layerNodes.end(),
layerNodes[i]), layerNodes.end());
27
28             layerNodes[i]->destroy();
29
30             //layerNodes.shrink_to_fit();
31             //delete(layerNodes[i]);
32         }
33         layerNodes.clear();
34     }
35
36
37     //Add Engine node to layer for loading and rendering
38     void Layer::addNodeToLayer(EngineBaseNode* node) {
39         layerNodes.push_back(node);
40     }
41
42
43     //Remove Node from scene 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) {
51         for (size_t i = 0; i < layerNodes.size(); i++) {
52             layerNodes.at(i)->load(r);
53         }
54     }
55 }
```

```

54     }
55
56     //Render all nodes added to layer
57     void Layer::drawNodes(SDL_Renderer* r) {
58         for (size_t i = 0; i < layerNodes.size(); i++) {
59
60             if (layerNodes.at(i)->getNode_type() == NodeType::SPRITE) {
61
62                 //Reload texture if the texture has been changed
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 release its memory so no extra memory been taking up
74             */
75             if (layerNodes.at(i)->getNode_type() == 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 [DsdEngine::Layer](#)

Namespaces

- [DsdEngine](#)

Layer.h

```
1 #ifndef _LAYER_  
2 #define _LAYER_  
3  
4 #include "EngineBaseNode.h"  
5 #include "ResourceTexture.h"  
6  
11 namespace DsdEngine {  
12     /**  
13      *   Base layer class for the engine  
14      */  
15     class Layer {  
16     public:  
17         //Add Gui Class As friend class  
18         friend class Gui;  
19  
23         Layer();  
24  
28         virtual ~Layer();  
29  
33         virtual void destroy();  
34  
39         void loadNodes(SDL_Renderer* r);  
40  
45         void drawNodes(SDL_Renderer* r);  
46  
51         void addNodeToLayer(EngineBaseNode* node);  
52  
57         void removeNodeFromLayer(EngineBaseNode* node);  
58  
59         std::vector<EngineBaseNode> layerNodes;  
60     private:  
61  
62     };  
63 }  
64 #endif // !_LAYER_
```

Particles.cpp File Reference

```
#include "Particles.h"
```

Namespaces

- [DsdlEngine](#)

Particles.cpp

```
1
2 #include "Particles.h"
3
4 namespace DsdlEngine {
5
6     Particles::Particles(int x, int y) {
7         //Set offsets
8         mPosX = x - 5 + (rand() % 25);
9         mPosY = y - 5 + (rand() % 25);
10
11         //Initialize animation
12         m_frame = rand() % 5;
13
14     }
15
16     Particles::~Particles() {}
17
18
19     bool Particles::isDead(Particles *p)
20     {
21         if ((p->life < 0) || (p->size < 1))
22             return true;
23         return false;
24     }
25
26
27 }
```


Particles.h File Reference

#include "DsdEngine.h"

Classes

- class [DsdEngine::Particles](#)

Namespaces

- [DsdEngine](#)

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 "<a href='\"#\"'>DsdEngine.h\"</a>"
4
5 namespace <a href='\"#\"'>DsdEngine\"</a> {
6
7 #define MAL_PARTICLE_LIFE 500
8
9
10 class <a href='\"#\"'>Particles\"</a> : public <a href='\"#\"'>EngineBaseNode\"</a>{
11 public:
12     <a href='\"#\"'>Particles\"</a>(int x, int y);
13     ~<a href='\"#\"'>Particles\"</a>();
14
15     bool <a href='\"#\"'>isDead\"</a>(<a href='\"#\"'>Particles\"</a> *p);
16
17     static inline float <a href='\"#\"'>torad\"</a>(float <a href='\"#\"'>angle\"</a>){
18         return (angle * M_PI) / 180;
19     }
20
21 private:
22
23     int <a href='\"#\"'>life\"</a>;
24     float <a href='\"#\"'>mPosX\"</a>, <a href='\"#\"'>mPosY\"</a>, <a href='\"#\"'>xvel\"</a>, <a href='\"#\"'>yvel\"</a>, <a href='\"#\"'>angle\"</a>, <a href='\"#\"'>size\"</a>;
25     Uint32 <a href='\"#\"'>endtime\"</a>;
26
27 };
28
29 }
30
31 #endif // !__Particles__
```

ResourceTexture.cpp File Reference

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

Namespaces

- [DsdEngine](#)

ResourceTexture.cpp

```
1
2  /*
3      Engine Texture
4      Handles all the loading for different textures in the engine
5      Handles main call to render texture
6
7      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 DsdEngine{
22
23     //Constructor
24     ResourceTexture::ResourceTexture() {
25         m\_Texture = NULL;
26         m\_iHeight = 0;
27         m\_iWidth = 0;
28     }
29
30     //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
43         SDL_Texture* newTexture = NULL;
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)
52                 SDL_Log("SDL_image Error : %s ", IMG_GetError());
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;
62             m_iHeight = loadedSurface->h;
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 }
73 //Return success
74 m_Texture = newTexture;
75 return m_Texture != NULL;
76 }
77
78
79 //Load ttf to sdl texture
80 bool ResourceTexture::loadTTF(std::string text, SDL_Color color, TTF_Font* myfont,
SDL_Renderer* r){
81
82     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     }
89     else{//Convert Surface to the Texture
90         m_Texture = SDL_CreateTextureFromSurface(r, textSurface);
91         if (m_Texture == NULL){
92             SDL_Log("TTF RenderText Blended Error : %s" , TTF_GetError());
93         }
94         else{
95             m_iWidth = textSurface->w;
96             m_iHeight = textSurface->h;
97         }
98         //Free surface as no longer needed
99         SDL_FreeSurface(textSurface);
100     }
101     //Return it
102     return m_Texture != NULL;
103 }
104
105
106 //Basic render
107 void ResourceTexture::render(Vec2 p, Vec2 s, SDL_Renderer* r, SDL_Rect* clip){
108     //Set rendering space and render to screen
109     SDL_Rect renderQuad;
110     if (s.x != NULL && s.y != NULL) { //For Sprites
111         renderQuad = { p.x, p.y, s.x, s.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){
119         renderQuad.w = clip->w;
120         renderQuad.h = clip->h;
121     }
122
123     //Render to screen
124     SDL_RenderCopy(r, m_Texture, clip, &renderQuad);

```

```

125     }
126
127
128     //Clean up
129     void ResourceTexture::destroy() {
130         if (m Texture != NULL) {
131             SDL_DestroyTexture(m Texture);
132             m Texture = NULL;
133             m iWidth = 0;
134             m iHeight = 0;
135         }
136     }
137
138     /*
139         Set Texture Blend Mode
140     */
141     void ResourceTexture::setBlendMode(SDL_BlendMode blend){
142         SDL_SetTextureBlendMode(m Texture, blend);
143     }
144
145
146     /*
147         Set Alpha value of texture for transperence
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 [DsdEngine::ResourceTexture](#)

Namespaces

- [DsdEngine](#)

ResourceTexture.h

```
1 #ifndef _RESOURCETEXTURE_
2 #define _RESOURCETEXTURE_
3
4 #include "EngineDefines.h"
5
6 namespace DsdEngine{
7
8     class ResourceTexture{
9     public:
10         ResourceTexture();
11
12         ~ResourceTexture();
13
14         bool loadTexture(std::string texturePath, SDL_Renderer* r);
15
16         bool loadTTF(std::string text, SDL_Color color, TTF_Font* myFont, SDL_Renderer* r);
17
18         void render(Vec2 p, Vec2 s, SDL_Renderer* r, SDL_Rect* clip = NULL);
19
20         void setBlendMode(SDL_BlendMode blending);
21
22         void setAlpha(Uint8 alpha);
23
24         void destroy();
25     private:
26         SDL_Texture* m\_Texture;
27         std::map<std::string, SDL_Texture*> m\_TextureMap;
28         int m\_iWidth, m\_iHeight;
29     };
30 }
31
32 #endif // !_RESOURCETEXTURE_
```

Scene.cpp File Reference

```
#include "IScene.h"
```

Namespaces

- [DsdEngine](#)

Scene.cpp

```
1 #include "IScene.h"
2 /*
3  Base Scene Class
4  author: @Derek O Brien
5  Description: Interface for base scene in game.
6 */
7
8
9 namespace DsdEngine {
10
11
12 void IScene::onInput() {
13     SDL_Event evnt;
14     m_inputManager.update();
15
16     while (SDL_PollEvent(&evnt)) {
17
18         if (evnt.key.repeat == 0) {
19
20
21             switch (evnt.type) {
22                 case SDL_QUIT:
23                     exit(1);
24                     break;
25                 case SDL_MOUSEMOTION:
26                     m_inputManager.setMouseCoords(evnt.motion.x, evnt.motion.y);
27                     break;
28                 case SDL_KEYDOWN:
29                     m_inputManager.pressKey(evnt.key.keysym.sym);
30                     break;
31                 case SDL_KEYUP:
32                     m_inputManager.releaseKey(evnt.key.keysym.sym);
33                     break;
34                 case SDL_MOUSEBUTTONDOWN:
35                     m_inputManager.pressKey(evnt.button.button);
36                     break;
37                 case SDL_MOUSEBUTTONUP:
38                     m_inputManager.releaseKey(evnt.button.button);
39                     break;
40                 //Touch down
41                 case SDL_FINGERDOWN:
42                     m_inputManager.pressKey(evnt.button.button);
43                     break;
44                 case SDL_FINGERMOTION:
45                     m_inputManager.setMouseCoords((float)evnt.motion.x,
46 (float)evnt.motion.y);
47                     break;
48                 case SDL_FINGERUP:
49                     m_inputManager.releaseKey(evnt.button.button);
50                     break;
51                 default:
52                     break;
53             }
54         }
55     }
56 }
```


SceneManager.cpp File Reference

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

Namespaces

- [DsdEngine](#)

SceneManager.cpp

```
1 #include "SceneManager.h"
2 #include "IScene.h"
3 namespace DsdEngine{
4
5     //Constructor sets links scene manager and ImainGame
6     SceneManager::SceneManager(IMainGame* game) :
7         m_pGame(game){
8         //Empty
9     }
10
11     //Move to next scene
12     IScene* SceneManager::moveNext(){
13         IScene* currentScene = getCurrentScene();
14         if (currentScene->getNextSceneIndex() != SCENE_INDEX_NO_SCENE){
15             m_iCurrentSceneIndex = currentScene->getNextSceneIndex();
16         }
17         return getCurrentScene();
18     }
19
20     //Move to Previous Scene
21     IScene* SceneManager::movePrevious(){
22         IScene* currentScene = getCurrentScene();
23         if (currentScene->getPreviousSceneIndex() != SCENE_INDEX_NO_SCENE){
24             m_iCurrentSceneIndex = currentScene->getPreviousSceneIndex();
25         }
26         return getCurrentScene();
27     }
28
29     //Set Current scene
30     void SceneManager::setScene(int nextScene){
31         m_iCurrentSceneIndex = nextScene;
32     }
33
34     //Add a scene to game
35     void SceneManager::addScene(IScene* newScene){
36         newScene->m_iSceneIndex = m_pScenes.size();
37         m_pScenes.push_back(newScene);
38         newScene->setParentGame(m_pGame);
39     }
40
41     //Clean up scenes
42     void SceneManager::destroy(){
43         for (size_t i = 0; i < m_pScenes.size(); i++){
44             m_pScenes[i]->destroyScene();
45             //delete m_pScenes[i];
46         }
47         m_pScenes.clear();
48         m_iCurrentSceneIndex = SCENE_INDEX_NO_SCENE;
49     }
50
51     //Return the current scene
52     IScene* SceneManager::getCurrentScene(){
53         if (m_iCurrentSceneIndex == SCENE_INDEX_NO_SCENE) {
54             return nullptr;
55         }
56     }
```



```
60     return m\_pScenes[m\_iCurrentSceneIndex];  
61 }  
62  
63 }
```

SceneManager.h File Reference

#include "EngineDefines.h"

Classes

- class [DsdEngine::SceneManager](#)

Namespaces

- [DsdEngine](#)

SceneManager.h

```
1 #ifndef _SCREENMANAGER_
2 #define _SCREENMANAGER_
3
4 #include "EngineDefines.h"
5
10 namespace DsdEngine{
11
12     //Forward declaration of classes
13     class IMainGame;
14     class IScene;
15
19     class SceneManager{
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
66         IScene* getCurrentScene();
67
68     protected:
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

- [DsdEngine](#)

Sprite.cpp

```
1 #include "Sprite.h"
2
3 /*
4     File: Sprite.h
5     Author: Derek O Brien
6     Description: Sprite file for creating in game sprites. Inherits from engine base node
7 */
8 namespace DsdEngine{
9
10     //Constructor
11     Sprite::Sprite(){
12         setEngineNodeType(NodeType::SPRITE);
13         m_frame = 0;
14         m_numFrames = 1;
15         m_opacity = 255;
16         m_objectBoundingBox = new SDL_Rect();
17         m_engineTexture = new ResourceTexture();
18     }
19
20     //DeConstructor
21     Sprite::~Sprite() {
22         destroy();
23     }
24
25
26     //Create basic sprite with one frame
27     void Sprite::create(Vec2 spriteSize, Vec2 position, std::string path){
28         setAssetPath(path);
29
30         m_size.x = spriteSize.x;
31         m_size.y = spriteSize.y;
32
33         m_position.x = position.x;
34         m_position.y = position.y;
35         setPosition(position);
36
37         m_numFrames = 1;
38         setBoundingBox(position, spriteSize);
39     }
40
41
42     //Create basic sprite with more than one frame
43     void Sprite::create(Vec2 spriteSize, Vec2 position, std::string path, int nf){
44         setAssetPath(path);
45
46         m_size.x = spriteSize.x;
47         m_size.y = spriteSize.y;
48
49         m_position.x = position.x;
50         m_position.y = position.y;
51         setPosition(position);
52
53
54         setBoundingBox(position, spriteSize);
55         m_numFrames = nf;
56     }
57 }
```

```

58 //Create basic sprite with more than one frame and physics body attached
59 void Sprite::createWithPhysics(b2World* world, Vec2 spriteSize, Vec2 position, std::string
path, int numFrames, float den, float fri, bool FixedRotation) {
60     setAssetPath(path);
61
62     m\_size.x = spriteSize.x;
63     m\_size.y = spriteSize.y;
64
65     m\_position.x = position.x;
66     m\_position.y = position.y;
67
68     setPosition(position);
69
70     setBoundingBox(position, spriteSize);
71     m\_numFrames = numFrames;
72
73     //Create new collision shape
74     m\_CollisionShape = new CollisionShape();
75     m\_CollisionShape->init(world, position, spriteSize, den, fri, FixedRotation);
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
90     m\_size.x = spriteSize.x;
91     m\_size.y = spriteSize.y;
92
93     m\_position.x = position.x;
94     m\_position.y = position.y;
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 [DsdEngine::Sprite](#)

Namespaces

- [DsdEngine](#)

Sprite.h

```
1 #ifndef _SPRITE_
2 #define _SPRITE_
3
4 #include "EngineBaseNode.h"
5 namespace DsdEngine{
6
7     class Sprite : public EngineBaseNode{
8     public:
9         Sprite();
10
11         virtual ~Sprite();
12
13         void create(Vec2 spriteSize, Vec2 position, std::string path);
14
15         void create(Vec2 spriteSize, Vec2 position, std::string path, int numFrames);
16
17         void createWithPhysics(b2World* world, Vec2 spriteSize, Vec2 position, std::string
18 path, int numFrames, float den, float fri, bool FixedRotation);
19
20         void updateTexure(Vec2 spriteSize, Vec2 position, std::string path, int numFrames);
21
22         b2Body* getCollisionBody() { return m CollisionShape->getBody(); }
23
24         void destroy();
25
26     private:
27
28     };
29 }
30 #endif // !_SPRITE_
```

Timing.cpp File Reference

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

Namespaces

- [DsdEngine](#)

Timing.cpp

```
1 #include "EngineDefines.h"
2 #include "Timing.h"
3
4 /*
5     File: timing.cpp
6     Author: Derek O'Brien
7 */
8 namespace DsdEngine{
9
10     FpsLimiter::FpsLimiter() {
11         //Empty
12     }
13     FpsLimiter::~FpsLimiter() {
14         //Empty
15     }
16
17     //Initilaze frame rate
18     void FpsLimiter::init(float maxFPS) {
19         setMaxFPS(maxFPS);
20     }
21
22     //Set Max Frame Rate
23     void FpsLimiter::setMaxFPS(float maxFPS) {
24         m_fMaxFPS = maxFPS;
25     }
26
27     //Get start timer
28     void FpsLimiter::begin() {
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);
37         //Limit the FPS to the max FPS
38         if (1000.0f / m_fMaxFPS > frameTicks) {
39             SDL_Delay((Uint32)(1000.0f / m_fMaxFPS - frameTicks));
40         }
41
42         return m_fFps;
43     }
44
45     void FpsLimiter::calculateFPS() {
46         //The number of frames to average
47         static const int NUM_SAMPLES = 10;
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
51         static int currentFrame = 0;
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
59     m fFrameTime = (float)(currentTicks - prevTicks);
60     frameTimes[currentFrame % NUM_SAMPLES] = m fFrameTime;
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++;
69     if (currentFrame < NUM_SAMPLES) {
70         count = currentFrame;
71     }
72     else {
73         count = NUM_SAMPLES;
74     }
75
76     //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) {
85         m fFps = 1000.0f / frameTimeAverage;
86     }
87     else {
88         m fFps = 120.0f;    //MAX ALLOWED
89     }
90 }
91 }

```

Timing.h File Reference

Classes

- class [DsdEngine::FpsLimiter](#)

Namespaces

- [DsdEngine](#)

Timing.h

```
1 #ifndef TIMING
2 #define TIMING
3
4
5
6
7 namespace DsdEngine{
8
9
10
11
12     class FpsLimiter {
13     public:
14         FpsLimiter();
15
16         ~FpsLimiter();
17
18         void init(float maxFPS);
19
20         void setMaxFPS(float maxFPS);
21
22         void begin();
23
24         float end();
25
26     private:
27         void calculateFPS();
28
29         float m\_fFps, m\_fMaxFPS, m\_fFrameTime;
30         unsigned int m\_iStartTicks;
31     };
32 }
33
34 #endif
```


Window.cpp File Reference

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

Namespaces

- [DsdlEngine](#)

Window.cpp

```
1 #include "Window.h"
2 #include "EngineError.h"
3 /*
4     File : Window.h
5     Author: Derek O Brien
6     Description: set up and create Window and render for sdl window
7 */
8 namespace DsdlEngine{
9
10     Window::Window() {
11         //Empty
12     }
13
14     Window::~~Window() {
15         destroy();
16     }
17
18     //Create Sdl Window
19     int Window::createWindow(std::string windowName, int screenWidth, int screenHeight,
unsigned int flag){
20
21         m_screenHeight = screenHeight;
22         m_screenWidth = screenWidth;
23
24         //Screen dimensions
25         SDL_Rect gScreenRect = { 0, 0, 320, 240 };
26         SDL_DisplayMode displayMode;
27         if (SDL_GetCurrentDisplayMode(0, &displayMode) == 0)
28         {
29             gScreenRect.w = displayMode.w;
30             gScreenRect.h = displayMode.h;
31         }
32
33         //Load Window for windows using size passed in
34 #ifdef _WIN32_
35         SDL_Log("Windows Created for Windows Platform");
36         m_pSdlWindow = SDL_CreateWindow(windowName.c_str(), 0, 0, screenWidth, screenHeight,
flag);
37         m_pSdlRenderer = SDL_CreateRenderer(m_pSdlWindow, -1, SDL_RENDERER_TARGETTEXTURE |
SDL_RENDERER_ACCELERATED | SDL_RENDERER_PRESENTVSYNC);
38         SDL_SetRenderDrawColor(m_pSdlRenderer, 0, 0, 0, 120);
39 #endif
40
41         //Load Window for Android using device screen Size
42 #ifdef _ANDROID_
43         SDL_Log("Windows Created for Android Platform");
44         m_pSdlWindow = SDL_CreateWindow(windowName.c_str(), SDL_WINDOWPOS_UNDEFINED,
SDL_WINDOWPOS_UNDEFINED, gScreenRect.w, gScreenRect.h, SDL_WINDOW_ALLOW_HIGHDPI);
45         m_pSdlRenderer = SDL_CreateRenderer(m_pSdlWindow, -1, SDL_RENDERER_TARGETTEXTURE |
SDL_RENDERER_ACCELERATED | SDL_RENDERER_PRESENTVSYNC);
46         SDL_SetRenderDrawColor(m_pSdlRenderer, 0, 0, 0, 120);
47 #endif
48
49
50         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
57     int imgFlags = IMG_INIT_PNG;
58     if (!(IMG_Init(&imgFlags) & imgFlags)){
59         SDL_Log("SDL_image could not initialize! SDL_image Error %s\n", IMG_GetError());
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);
76     SDL_DestroyWindow(m_pSdlWindow);
77     m_pScreenSurface = nullptr;
78     SDL_Quit();
79 }
80
81 }

```

Window.h File Reference

#include "EngineDefines.h"

Classes

- class [DsdEngine::Window](#)

Namespaces

- [DsdEngine](#)

Window.h

```
1 #ifndef _WINDOW_
2 #define _WINDOW_
3
4 #include "EngineDefines.h"
5
10 //Wrap Frame Work Code in Namespace
11 namespace DsdEngine{
12
16     class Window{
17     public:
21         Window();
22
26         virtual ~Window();
27
28
37         int createWindow(std::string windowNname, int screenWidth, int screenHeight, unsigned
int flag);
38
42         void swapBuffer();
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;
69         SDL_Renderer* m_pSdlRenderer;
70         SDL_Surface* m_pScreenSurface;
72         int m_screenHeight, m_screenWidth;
74     };
75 }
76 #endif // !_WINDOW_
```

XmlLocalStorage.cpp File Reference

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

Namespaces

- [DsdEngine](#)

Variables

- static XmlLocalStorage * [DsdEngine::Instance](#) = nullptr

XmlLocalStorage.cpp

```
1 #include "XmlLocalStorage.h"  
2 #include "FileIO.h"  
3  
4 /*  
5     File: XmlLocalStorage  
6     Author: Derek O'Brien  
7     Description: For loading and saving values to XML file  
8 */  
9  
10 using namespace tinycl2;  
11 using namespace std;  
12  
13 namespace DsdEngine {  
14  
15     //Create As Singleton static instance  
16     static XmlLocalStorage* Instance = nullptr;  
17     XmlLocalStorage* XmlLocalStorage::getInstance() {  
18         if (!Instance) {  
19             Instance = new (std::nothrow) XmlLocalStorage();  
20         }  
21         return Instance;  
22     }  
23  
24  
25  
26     //get integer value for key passed in  
27     int XmlLocalStorage::getIntegerForKey(const char* key) {  
28  
29         const char* value = nullptr;  
30         XMLElement* rootNode;  
31         XMLDocument* doc;  
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  
43         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
57     const char* value = nullptr;
58     XElement* rootNode;
59     XMLDocument* doc;
60     XElement* node;
61
62     //Get node from xml file
63     node = FileIO::getInstance()->getXMLNodeForKey(key, &rootNode, &doc);
64
65     //Get the value from the node
66     if (node && node->FirstChild()) {
67         value = (const char*)(node->FirstChild()->Value());
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
86     const char* value = nullptr;
87     XElement* rootNode;
88     XMLDocument* doc;
89     XElement* node;
90
91     //Get node from xml file
92     node = FileIO::getInstance()->getXMLNodeForKey(key, &rootNode, &doc);
93
94     //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
111 //Get float value for key passed in
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
120     const char* value = nullptr;
121     XElement* rootNode;
122     XMLDocument* doc;

```

```

123     XMLNode* 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
134     string temp = "No Value Found";
135
136     if (value) {
137         temp = string(value);
138     }
139
140     return temp;
141 }
142
143 //Set a string value for the key
144 void XmlLocalStorage::setIntegerForKey(int value, const char* key) {
145     // check key
146     if (!key) {
147         return;
148     }
149
150     // format the value as char for saving
151     char tmp[50];
152     memset(tmp, 0, 50);
153 #ifdef __WIN32__
154     sprintf_s(tmp, "%d", value);
155 #endif
156
157 #ifdef __ANDROID__
158     sprintf(tmp, "%d", value);
159 #endif
160
161     //Save the Value and key
162     FileIO::getInstance()->setIntegerForKey(tmp, key);
163 }
164
165 //Set bool value for key passed in
166 void XmlLocalStorage::setBoolForKey(bool value, const char* key) {
167     if (value == true) {
168         setStringForKey("true", key);
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
178     if (!key) {
179         return;
180     }
181
182     // format the value as char for saving
183     char tmp[50];
184     memset(tmp, 0, 50);
185 #ifdef __WIN32__
186     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
198 void XmlLocalStorage::setFloatForKey(float value, const char* key) {
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);
207 }
208
209
210 //Delete value fo node
211 void XmlLocalStorage::deleteValueForKey(const char* key) {
212
213     XMLElement* rootNode;
214     XMLDocument* doc;
215     XMLElement* node;
216     XMLPrinter printer;
217
218     // check the params
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
227     if (!node) {
228         return;
229     }
230
231     if (doc){
232         //Delete Node
233         doc->DeleteNode(node);
234         std::string path;
235         path = FileIO::getInstance()->getWritablePath() + "Default.xml";
236
237         // attach printer to the document you want to convert in to a std::string
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);
246         }
247         delete doc;
248     }
249 }
250 }

```

XmlLocalStorage.h File Reference

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

Classes

- class [DsdEngine::XmlLocalStorage](#)

Namespaces

- [DsdEngine](#)

XmlLocalStorage.h

```
1 #ifndef _XMLLOCALSTORAGE_
2 #define _XMLLOCALSTORAGE_
3
4 #include "DsdEngine.h"
5 #include "EngineDefines.h"
6 #include "../dependencies/tinyxml/tinyxml2.h"
7
11 namespace DsdEngine {
12
13     class XmlLocalStorage {
14     public:
15
16         static XmlLocalStorage* getInstance();
17
18         void setIntegerForKey(int value, const char* key);
19
20         void setBoolForKey(bool value, const char* key);
21
22         void setDoubleForKey(double value, const char* key);
23
24         void setFloatForKey(float value, const char* key);
25
26         void setStringForKey(std::string value, const char* key);
27
28         int getIntegerForKey(const char* key);
29
30         bool getBoolForKey(const char* key);
31
32         double getDoubleForKey(const char* key);
33
34         float getFloatForKey(const char* key);
35
36         std::string getStringForKey(const char* key);
37
38         void deleteValueForKey(const char* key);
39
40     protected:
41
42         XmlLocalStorage() {};
43
44         virtual ~XmlLocalStorage() {};
45
46     private:
47     };
48 }
49 #endif // !_XMLLOCALSTORAGE_
```


Index

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

- DsdEngine::Label, 52
- CollisionShane.cpp, 93
- CollisionShape
 - DsdEngine::CollisionShape, 16
- CollisionShape.h, 94
- create
 - DsdEngine::Label, 53
 - DsdEngine::Sprite, 72, 73
- createSpriteButton
 - DsdEngine::Button, 14
- createTextButton
 - DsdEngine::Button, 14
- createWindow
 - DsdEngine::Window, 77
- createWithPhysics
 - DsdEngine::Sprite, 73
- createXMLFile
 - DsdEngine::FileIO, 32
- DEBUG_DSDL
 - EngineError.h, 108
- DEBUG_MSG
 - EngineError.h, 108
- DEFAULT_ROOT_NAME
 - EngineDefines.h, 104
- DEGTORAD
 - EngineDefines.h, 104
- deleteValueForKey
 - DsdEngine::XmlLocalStorage, 81
- destroy
 - DsdEngine::AudioManager, 11
 - DsdEngine::Button, 14
 - DsdEngine::CollisionShape, 16
 - DsdEngine::DsdGui, 20
 - DsdEngine::EngineBaseNode, 23
 - DsdEngine::Label, 53
 - DsdEngine::Layer, 55
 - DsdEngine::ResourceTexture, 62
 - DsdEngine::SceneManager, 66
 - DsdEngine::Sprite, 73
 - DsdEngine::Window, 78
- destroyScene
 - DsdEngine::IScene, 49
- draw
 - DsdEngine::IMainGame, 39
- drawNodes
 - DsdEngine::Layer, 55
- drawScene
 - DsdEngine::IScene, 49
- DsdEngine, 7
 - BASENODE, 9
 - BUTTON, 9
 - ButtonState, 8
 - ButtonType, 8
 - CallBack, 8
 - CallBackTimer, 8
 - CHANGE_NEXT, 9

- CHANGE_PREVIOUS, 9
- EXIT_APP, 9
- HOVERING, 8
- init, 9
- Instance, 9, 10
- LABEL, 9
- LABEL_BTN, 8
- LABEL_DYNAMIC, 9
- LABEL_STATIC, 9
- LableType, 8
- make_unique, 9
- NodeType, 9
- NONE, 9
- NORMAL, 8
- PARTICLE, 9
- PRESSED, 8
- RUNNING, 9
- SceneState, 9
- SPRITE, 9
- SPRITE_BTN, 8
- DsdEngine.cpp, 95
- DsdEngine.h, 96
- DsdEngine::AudioManager, 11
 - ~AudioManager, 11
 - AudioManager, 11
 - destroy, 11
 - init, 12
 - loadMusic, 12
 - loadSFX, 12
 - m_bgAudioMap, 12
 - m_bisInitialized, 12
 - m_sfxAudioMap, 12
- DsdEngine::Button, 13
 - ~Button, 14
 - Button, 13
 - checkInput, 14
 - createSpriteButton, 14
 - createTextButton, 14
 - destroy, 14
 - getButtonName, 14
 - m_buttonName, 15
 - m_eCurrentState, 15
 - m_label, 15
 - m_spriteBtn, 15
 - onClicked, 15
 - onMouseEnters, 15
 - onMouseLeaves, 15
- DsdEngine::CollisionShape, 16
 - ~CollisionShape, 16
 - CollisionShape, 16
 - destroy, 16
 - getBody, 17
 - getDimensions, 17
 - getFixture, 17
 - init, 17
 - m_body, 17

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

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

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

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

- DsdEngine::InputManager, 44
- isSwipe
 - DsdEngine::InputManager, 44
- isSwipeDown
 - DsdEngine::InputManager, 44
- isSwipeLeft
 - DsdEngine::InputManager, 45
- isSwipeRight
 - DsdEngine::InputManager, 45
- isSwipeUp
 - DsdEngine::InputManager, 45
- isTextureChanged
 - DsdEngine::EngineBaseNode, 24
- isTouch
 - DsdEngine::InputManager, 45
- Label
 - DsdEngine::Label, 52
- LABEL
 - DsdEngine, 9
- Label.cpp, 134
- Label.h, 135
- LABEL_BTN
 - DsdEngine, 8
- LABEL_DYNAMIC
 - DsdEngine, 9
- LABEL_STATIC
 - DsdEngine, 9
- LableType
 - DsdEngine, 8
- Layer
 - DsdEngine::Layer, 54
- Layer.cpp, 136
- Layer.h, 138
- layerNodes
 - DsdEngine::Layer, 55
- life
 - DsdEngine::Particles, 60
- load
 - DsdEngine::EngineBaseNode, 24
- loadDocument
 - DsdEngine::FileIO, 33
- loadMusic
 - DsdEngine::AudioManager, 12
- loadNodes
 - DsdEngine::Layer, 55
- loadScene
 - DsdEngine::IScene, 50
- loadSFX
 - DsdEngine::AudioManager, 12
- loadTexture
 - DsdEngine::ResourceTexture, 63
- loadTTF
 - DsdEngine::ResourceTexture, 63
- m_assetPath
 - DsdEngine::EngineBaseNode, 27
- m_audioManager

- DsdEngine::IMainGame, 41
- m_bgAudioMap
 - DsdEngine::AudioManager, 12
- m_bisInitialized
 - DsdEngine::AudioManager, 12
- m_bIsPaused
 - DsdEngine::IMainGame, 41
- m_bIsRunning
 - DsdEngine::IMainGame, 41
- m_body
 - DsdEngine::CollisionShape, 17
- m_btn
 - DsdEngine::DsdGui, 21
- m_buttonName
 - DsdEngine::Button, 15
- m_Chunk
 - DsdEngine::SFX, 68
- m_CollisionShape
 - DsdEngine::EngineBaseNode, 27
- m_currentFrame
 - DsdEngine::EngineBaseNode, 27
- m_dimensions
 - DsdEngine::CollisionShape, 17
- m_eCurrentState
 - DsdEngine::Button, 15
 - DsdEngine::IScene, 51
- m_engineTexture
 - DsdEngine::EngineBaseNode, 27
- m_fFps
 - DsdEngine::FpsLimiter, 36
 - DsdEngine::IMainGame, 41
- m_fFrameTime
 - DsdEngine::FpsLimiter, 36
- m_fileName
 - DsdEngine::FileIO, 33
- m_fixtures
 - DsdEngine::CollisionShape, 17
- m_fMaxFPS
 - DsdEngine::FpsLimiter, 36
- m_font
 - DsdEngine::EngineBaseNode, 27
- m_FontMap
 - DsdEngine::EngineBaseNode, 27
- m_frame
 - DsdEngine::EngineBaseNode, 28
- m_game
 - DsdEngine::IScene, 51
- m_gSpriteClips
 - DsdEngine::EngineBaseNode, 28
- m_iCurrentSceneIndex
 - DsdEngine::SceneManager, 66
- m_iHeight
 - DsdEngine::ResourceTexture, 64
- m_inputManager
 - DsdEngine::IScene, 51
- m_InputManager

- DsdEngine::IMainGame, 41
- m_iSceneIndex
 - DsdEngine::IScene, 51
- m_iStartTicks
 - DsdEngine::FpsLimiter, 36
- m_iWidth
 - DsdEngine::ResourceTexture, 64
- m_label
 - DsdEngine::Button, 15
 - DsdEngine::DsdGui, 21
- m_labelText
 - DsdEngine::EngineBaseNode, 28
- m_labelType
 - DsdEngine::Label, 53
- m_Music
 - DsdEngine::Music, 58
- m_numFrames
 - DsdEngine::EngineBaseNode, 28
- m_objectBoundingBox
 - DsdEngine::EngineBaseNode, 28
- m_opacity
 - DsdEngine::EngineBaseNode, 28
- m_path
 - DsdEngine::FileIO, 34
- m_pCurrentRunning
 - DsdEngine::IMainGame, 41
- m_pGame
 - DsdEngine::SceneManager, 66
- m_pGameRenderer
 - DsdEngine::IMainGame, 42
- m_position
 - DsdEngine::EngineBaseNode, 28
- m_pSceneManager
 - DsdEngine::IMainGame, 42
- m_pScenes
 - DsdEngine::SceneManager, 66
- m_pScreenSurface
 - DsdEngine::Window, 78
- m_pSdlRenderer
 - DsdEngine::Window, 78
- m_pSdlWindow
 - DsdEngine::Window, 79
- m_screenHeight
 - DsdEngine::Window, 79
- m_screenWidth
 - DsdEngine::Window, 79
- m_sfxAudioMap
 - DsdEngine::AudioManager, 12
- m_size
 - DsdEngine::EngineBaseNode, 28
- m_spriteBtn
 - DsdEngine::Button, 15
- m_textColor
 - DsdEngine::EngineBaseNode, 28
- m_textSize
 - DsdEngine::EngineBaseNode, 28
- m_Texture
 - DsdEngine::ResourceTexture, 64
- m_TextureMap
 - DsdEngine::ResourceTexture, 64
- m_Window
 - DsdEngine::IMainGame, 42
- m_windowHeight
 - DsdEngine::IMainGame, 42
- m_windowWidth
 - DsdEngine::IMainGame, 42
- mainAssetsPath
 - DsdEngine::IMainGame, 42
- mainLoop
 - DsdEngine::IMainGame, 40
- make_unique
 - DsdEngine, 9
- MAL_PARTICLE_LIFE
 - Particles.h, 140
- METRESTOPIXELS
 - EngineDefines.h, 104
- moveNext
 - DsdEngine::SceneManager, 66
- movePrevious
 - DsdEngine::SceneManager, 66
- mPosX
 - DsdEngine::Particles, 60
- mPosY
 - DsdEngine::Particles, 60
- nodeType
 - DsdEngine::EngineBaseNode, 29
- NodeType
 - DsdEngine, 9
- NONE
 - DsdEngine, 9
- NORMAL
 - DsdEngine, 8
- NS_DSDDL_END
 - EngineDefines.h, 104
- NS_DSDDL_START
 - EngineDefines.h, 104
- onClicked
 - DsdEngine::Button, 15
- onEntryScene
 - DsdEngine::IScene, 50
- onExit
 - DsdEngine::IMainGame, 40
- onExitScene
 - DsdEngine::IScene, 50
- onInit
 - DsdEngine::IMainGame, 40
- onInput
 - DsdEngine::IScene, 50
- onMouseEnters
 - DsdEngine::Button, 15
- onMouseLeaves
 - DsdEngine::Button, 15

- onSDLEvent
 - DsdLEngine::DsdLGui, 21
 - DsdLEngine::IMainGame, 40
- PARTICLE
 - DsdLEngine, 9
- Particles
 - DsdLEngine::Particles, 59
- Particles.cpp, 139
- Particles.h, 140
 - MAL_PARTICLE_LIFE, 140
- PIXELSTOMETRES
 - EngineDefines.h, 104
- play
 - DsdLEngine::Music, 57
 - DsdLEngine::SFX, 68
- PRESSED
 - DsdLEngine, 8
- pressKey
 - DsdLEngine::InputManager, 45
- RADTODEG
 - EngineDefines.h, 104
- releaseKey
 - DsdLEngine::InputManager, 45
- removeNodeFromLayer
 - DsdLEngine::Layer, 55
- render
 - DsdLEngine::EngineBaseNode, 25
 - DsdLEngine::ResourceTexture, 63
- renderCollisionShape
 - DsdLEngine::EngineBaseNode, 25
- ResourceTexture
 - DsdLEngine::ResourceTexture, 62
- ResourceTexture.cpp, 141
- ResourceTexture.h, 144
- run
 - DsdLEngine::IMainGame, 40
- RUNNING
 - DsdLEngine, 9
- scaleHeight
 - DsdLEngine::EngineBaseNode, 25
- scaleNode
 - DsdLEngine::EngineBaseNode, 25
- scaleWidth
 - DsdLEngine::EngineBaseNode, 25
- Scene.cpp, 145
- SCENE_INDEX_NO_SCENE
 - IScene.h, 132
- sceneLayers
 - DsdLEngine::IScene, 51
- SceneManager
 - DsdLEngine::IScene, 50
 - DsdLEngine::SceneManager, 65
- SceneManager.cpp, 147
- SceneManager.h, 149
- SceneState
 - DsdLEngine, 9
- setAlpha
 - DsdLEngine::ResourceTexture, 63
- setAssetPath
 - DsdLEngine::EngineBaseNode, 25
- setAssetsPath
 - DsdLEngine::FileIO, 33
- setBlendMode
 - DsdLEngine::ResourceTexture, 63
- setBoolForKey
 - DsdLEngine::XmlLocalStorage, 82
- setBoundingBox
 - DsdLEngine::EngineBaseNode, 25
- setDoubleForKey
 - DsdLEngine::XmlLocalStorage, 82
- setEngineNodeType
 - DsdLEngine::EngineBaseNode, 26
- setFileToOpen
 - DsdLEngine::FileIO, 33
- setFloatForKey
 - DsdLEngine::XmlLocalStorage, 82
- setFps
 - DsdLEngine::IMainGame, 40
- setGUIPos
 - DsdLEngine::DsdLGui, 21
- setHeight
 - DsdLEngine::EngineBaseNode, 26
- setIntegerForKey
 - DsdLEngine::XmlLocalStorage, 82
- setMaxFPS
 - DsdLEngine::FpsLimiter, 36
- setMouseCoords
 - DsdLEngine::InputManager, 45
- setOpacity
 - DsdLEngine::EngineBaseNode, 26
- setParentGame
 - DsdLEngine::IScene, 50
- setPaused
 - DsdLEngine::IMainGame, 40
- setPosition
 - DsdLEngine::EngineBaseNode, 26
- setPositionX
 - DsdLEngine::EngineBaseNode, 26
- setPositionY
 - DsdLEngine::EngineBaseNode, 26
- setRunning
 - DsdLEngine::IMainGame, 41
- setScene
 - DsdLEngine::SceneManager, 66
- setSceneRunning
 - DsdLEngine::IScene, 50
- setSize
 - DsdLEngine::EngineBaseNode, 26
- setStringForKey
 - DsdLEngine::XmlLocalStorage, 82
- setType
 - DsdLEngine::Label, 53

- setUpdateTextureTrue
 - DsdEngine::EngineBaseNode, 27
- setupWindow
 - DsdEngine::IMainGame, 41
- setValueForKey
 - DsdEngine::FileIO, 33
- setWidth
 - DsdEngine::EngineBaseNode, 27
- size
 - DsdEngine::Particles, 60
- Size
 - DsdEngine::Size, 70
- Sprite
 - DsdEngine::Sprite, 72
- SPRITE
 - DsdEngine, 9
- Sprite.cpp, 150
- Sprite.h, 152
- SPRITE_BTN
 - DsdEngine, 8
- swapBuffer
 - DsdEngine::Window, 78
- swipedown
 - DsdEngine::InputManager, 46
- swipeleft
 - DsdEngine::InputManager, 46
- swiperight
 - DsdEngine::InputManager, 47
- swipeup
 - DsdEngine::InputManager, 47
- Timing.cpp, 153
- Timing.h, 155
- torad
 - DsdEngine::Particles, 60
- TOTAL_PARTICLES
 - EngineDefines.h, 104
- update
 - DsdEngine::IMainGame, 41
 - DsdEngine::InputManager, 46
- updateLabelText
 - DsdEngine::EngineBaseNode, 27
- updateScene
 - DsdEngine::IScene, 50
- updateTextureInfo
 - DsdEngine::EngineBaseNode, 29
- updateTexure
 - DsdEngine::Sprite, 73
- USING_NS_DSDL
 - EngineDefines.h, 104
- Vec2
 - DsdEngine::Vec2, 75
- w_
 - DsdEngine::Size, 71
- wasKeyDown
 - DsdEngine::InputManager, 46
- Window
 - DsdEngine::Window, 77
- Window.cpp, 156
- Window.h, 158
- windowFlag
 - DsdEngine::IMainGame, 42
- windowtitle
 - DsdEngine::IMainGame, 42
- writeDocument
 - DsdEngine::FileIO, 33
- x_
 - DsdEngine::Vec2, 76
- XML_FILE
 - EngineDefines.h, 105
- XmlLocalStorage
 - DsdEngine::XmlLocalStorage, 80
- XmlLocalStorage.cpp, 159
- XmlLocalStorage.h, 163
- xvel
 - DsdEngine::Particles, 60
- y_
 - DsdEngine::Vec2, 76
- yvel
 - DsdEngine::Particles, 60
- ZERO
 - DsdEngine::Vec2, 76