# Table of Contents

Todo List ......................................................................................................................... 2
Namespace Index .................................................................................................................. 3
Hierarchical Index .................................................................................................................. 4
Class Index ........................................................................................................................... 5
  rag::fs .............................................................................................................................. 8
Class Documentation .......................................................................................................... 9
  rag::Bitmap ....................................................................................................................... 9
  rag::BMPFont .................................................................................................................... 11
  rag::Chrono ..................................................................................................................... 12
  rag::Color ......................................................................................................................... 13
  rag::Color4B .................................................................................................................... 15
  rag::DisplayObject ........................................................................................................... 16
  rag::DropShadowFilter ................................................................................................... 22
  rag::Ease .......................................................................................................................... 23
  events::Event .................................................................................................................... 24
  events::EventDispatcher ................................................................................................... 25
  events::EventListener ....................................................................................................... 26
  rag::File ........................................................................................................................... 27
  rag::MovieClip::Frame ..................................................................................................... 29
  rag::Image ....................................................................................................................... 30
  rag::ImageLoader ............................................................................................................. 32
  rag::ImageLoaderJP ......................................................................................................... 33
  rag::ImageLoaderPNG ..................................................................................................... 34
  rag::ImageLoaderPVR ...................................................................................................... 35
  rag::InputManager .......................................................................................................... 36
  rag::ITextFont .................................................................................................................. 37
  rag::Keyboard .................................................................................................................. 38
  events::KeyboardEvent .................................................................................................... 40
  rag::KeyboardManager .................................................................................................... 41
  rag::Material .................................................................................................................... 42
  rag::MovieClip .................................................................................................................. 44
  rag::fs::path ...................................................................................................................... 46
  rag::Program .................................................................................................................... 47
  rag::Rectangle ................................................................................................................... 48
  rag::Renderer ................................................................................................................... 49
  rag::RenderTarget ............................................................................................................ 51
  rag::Resource ................................................................................................................... 52
  rag::ResourceMgr ............................................................................................................. 54
  rag::Screen ........................................................................................................................ 56
  rag::Shader ....................................................................................................................... 57
  spine::SkeletonDrawable ................................................................................................... 58
  rag::TextEdit ................................................................................................................... 60
  events::TextEvent ............................................................................................................ 61
  rag::TextField .................................................................................................................. 62
  rag::TextInput ................................................................................................................... 64
  rag::Timer .......................................................................................................................... 65
  events::TouchEvent .......................................................................................................... 66
  rag::TTFFont ..................................................................................................................... 67
  rag::TUniformVar ............................................................................................................. 68
  rag::Vertex ....................................................................................................................... 69
  rag::XFLBinaryParser ..................................................................................................... 70
  rag::XFLParser .................................................................................................................. 71
Todo List

**Member rag::DisplayObject::autoScaleOnTouch**
autoScale is weird, the only thing that should autoscale should be buttons. Actually buttons should also by default captureInput and checkhitpoint.

**Member rag::DisplayObject::onNativeEvent  (events::TouchEvent &event)**
This API should be moved to an input UI panel

**Member rag::DisplayObject::soundName**
This should be part of a button.

**Class rag::KeyboardManager**
It's confusing to have a Keyboard and a KeyboardManager.

**Class rag::Material**
Make a clearer design taking into account efficiency and naming convention. Refactor classes like Program, Shader. May make sense to allow change shader uniforms in Shader class instead of Material. DisplayObjects should use Material references, and then have the ability to make copies if they need to modify something, (e.g., change shader uniforms).
Namespace Index

Namespace List
Here is a list of all documented namespaces with brief descriptions:

rag::fs (File system namespace ) ...........................................................................................................8
Hierarchical Index

Class Hierarchy

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

rag::Chrono .......................................................... 12
rag::Color ........................................................... 13
rag::Color4B ......................................................... 15
rag::DropShadowFilter ........................................... 22
rag::Ease ............................................................. 23
events::Event ......................................................... 24
  events::KeyboardEvent .......................................... 40
  events::TextEvent ................................................ 60
  events::TouchEvent .............................................. 66
events::EventDispatcher ......................................... 25
  rag::DisplayObject .......................................... 16
    rag::Bitmap ................................................... 9
    rag::Keyboard ............................................... 38
    rag::MovieClip .............................................. 44
    rag::TextEdit ............................................... 60
    rag::TextField .............................................. 62
    rag::TextInput ............................................. 64
spine::SkeletonDrawable ........................................ 58

  rag::KeyboardManager ....................................... 41

events::EventListener ........................................... 26
  rag::Keyboard ............................................... 38
  rag::TextInput ............................................. 64

rag::File ................................................................ 27
rag::MovieClip::Frame ........................................... 29
rag::ImageLoader ............................................... 32
  rag::ImageLoaderJPG .......................................... 33
  rag::ImageLoaderPNG .......................................... 34
  rag::ImageLoaderPVR .......................................... 35

  rag::InputManager ............................................ 36
  rag::ITextFont ................................................ 37
    rag::BMPFont ................................................ 11
    rag::TTFFont ............................................... 67

  rag::Material ................................................. 42
  rag::fs::path ............................................... 46
rag::Program ................................................................. 47
rag::Rectangle .................................................................. 48
rag::Renderer ................................................................. 49
rag::RenderTarget ............................................................. 51
rag::Resource .................................................................... 52
    rag::Image .................................................................. 30

rag::ResourceMgr ............................................................. 54
rag::Screen ...................................................................... 56
rag::Shader ................................................................. 57
rag::Timer ................................................................. 65
rag::TUniformVar .......................................................... 68
rag::Vertex ...................................................................... 69
rag::XFLBinaryParser ...................................................... 70
rag::XFLParser ............................................................. 71
Class Index

Class List

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

rag::Bitmap (Provides the ability to show images) .............................................. 9
rag::BMPFont (Font system based on bitmap fonts) ............................................ 11
rag::Chrono (Helper class to count time elapsed from a moment in time) ............. 12
rag::Color (Represents RGBA color) .................................................................... 13
rag::Color4B (Color representation using 4 bytes) ............................................. 15
rag::DisplayObject (Core object used to display things in screen) ..................... 16
rag::DropShadowFilter (Shadow effect for TextField instances) ....................... 22
rag::Ease (Collection of code-generated curves useful to create procedural tween animations) ........................................... 23
events::Event (Base class for event system) .................................................. 24
events::EventDispatcher (Base class used to dispatch events) ......................... 25
events::EventListener (Interface that allows to listen events) ......................... 26
rag::File (File multiplatform abstraction to read contents of a file) .................... 27
rag::MovieClip::Frame (Internal of MovieClip, represents a single frame) .......... 29
rag::Image (Image object) .................................................................................. 30
rag::ImageLoader (Interface to load images) .................................................. 32
rag::ImageLoaderJPG (Loader for .jpg format) .................................................. 33
rag::ImageLoaderPNG (Loader for .png format) ............................................... 34
rag::ImageLoaderPVR (Loader for .pvr compressed format) ......................... 35
rag::KeyboardManager (Singleton class that dispatches Keyboard events) ......... 40
rag::Keyboard (Multiplatform keyboard abstraction) ........................................ 41
rag::Material (Represents a render material that would affect how objects will be rendered) ........................................... 42
rag::MovieClip (Allows to use imported animations created by Flash CS tool) .... 44
rag::fs::path (Mimics boost fs::path class with some limited functionality) ......... 46
rag::Program (Represents a Shader Program) ............................................. 47
rag::Rectangle (Represents a Rectangle) .................................................. 48
rag::Renderer (Contains methods to render objects) ......................................... 49
rag::RenderTarget (Object where DisplayObject instances with render capability are suposed to render) ........................................... 51
rag::Resource (Abstract class the represent a game Resource, typically something costly to loaded) ........................................... 52
rag::ResourceMgr (Handles Resource management, including loading and unloading Resource instances) ........................................... 54
rag::Screen (Contains information about the current device) ............................ 56
rag::Shader (Represents a GL Shader) .......................................................... 57
spine::SkeletonDrawable (Display 2D Skeletal Animations made with the 3rd party tool Spine) ........................................... 58
rag::TextEdit (Creates a native window to edit a text) ..................................... 60
events::TextEvent (Dispatched by InputText when user writes one character) .... 61
rag::TextField (High level abstraction to render texts in display list ) ...........................................62
rag::TextInput (Helper object to add input to a TextField ) ...............................................................64
rag::Timer (Provides time-related functionality ) .................................................................................65
events::TouchEvent (Event for handle input from screen ) .................................................................66
rag::TTFFont (Implementation of ITextFint based on TrueType or OpenType Fonts ) .................67
rag::TUniformVar .......................................................................................................................................68
rag::Vertex (Vertex representation ) ....................................................................................................69
rag::XFLBinaryParser .............................................................................................................................70
rag::XFLParser (Parser of XFL documents generated by the 3rd party editor tool FlashCS ) ........71
Namespace Documentation

rag::fs Namespace Reference

file system namespace

Classes
- class path

Mimics boost fs::path class with some limited functionality.

Detailed Description

file system namespace
Class Documentation

rag::Bitmap Class Reference

Provides the ability to show images.
#include <Bitmap.h>
Inheritance diagram for rag::Bitmap:

```
Public Member Functions
- Bitmap(const std::string &path)
  Create a Bitmap using an image in a given path.
- Bitmap(Image *image)
  Create a Bitmap with an existing image.
- virtual void render() override
  Renders the DisplayObject in the screen.
- virtual void prerender() override
  Temporal transition to new automatic batch render.
```

Public Attributes
- Image *image
  Shared image.
- glm::vec4 uv
  Texture coordinates. used with texture atlases.

Additional Inherited Members

Detailed Description

Provides the ability to show images.

A Bitmap can be used with a path to an image or directly with an image. The Bitmap will represent an arbitrary image and supports all basic transformations as any DisplayObject, such as scale, rotation, skew. All those transformations can be inherited in the Display List hierarchy.
Constructor & Destructor Documentation

**Bitmap::Bitmap (const std::string & path)**

Create a **Bitmap** using an image in a given path.
It is assumed the image exists in the path, otherwise an error is logged, and nothing is shown.

**See also:**
Image

**Bitmap::Bitmap (Image * image)**

Create a **Bitmap** with an existing image.
The **Bitmap** is created with an already existant **Image**, so images downloaded from The Internet or procedurally generated images can be used.

Member Data Documentation

**glm::vec4 rag::Bitmap::uv**

Texture coordinates. used with texture atlases.
Will default to (0, 1), (0, 1) coordinates using the entire image, but can be set to any pair of coordinates to use like a sprite inside an image atlas.

The documentation for this class was generated from the following files:
- D:/prj/rag/include/rag/Bitmap.h
- D:/prj/rag/include/rag/Bitmap.cpp
rag::BMPFont Class Reference

Font system based on bitmap fonts.
#include <BMPFont.h>

Inheritance diagram for rag::BMPFont:

```
  rag::ITextFont
    ↓
  rag::BMPFont
```

Public Member Functions

- **BMPFont** (const std::string &path, const std::string &name)
- virtual int **getWidth** (const std::string &text)
  Returns the width of a text.
- virtual void **print** (const std::string &text, const glm::mat4 &matrix)
  Renders text. Assumes ortho projection 1:1.

Detailed Description

Font system based on bitmap fonts.
This kind of fonts assumes the output generated with the AngelCode tool that can be found here: http://www.angelcode.com/products/bmfont/

Constructor & Destructor Documentation

**BMPFont::BMPFont** (const std::string & path, const std::string & name)

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>The path where the font descriptor file is. i.e., &quot;assets/&quot;</td>
</tr>
<tr>
<td>name</td>
<td>The font descriptor file name. It is assumed that font descriptor and font texture are in the same folder.</td>
</tr>
</tbody>
</table>

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

- D:/prj/rag/include/rag/BMPFont.h
- D:/prj/rag/include/rag/BMPFont.cpp
rag::Chrono Class Reference

Helper class to count time elapsed from a moment in time.
#include <Timer.h>

Public Member Functions

- float getElapsedTime ()
  Get elapsed time from chrono construction.
- void reset ()
  Reset time to 0.

Detailed Description

Helper class to count time elapsed from a moment in time.

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

- D:/prj/rag/include/rag/Timer.h
- D:/prj/rag/include/rag/Timer.cpp
rag::Color Class Reference

Represents RGBA color.
#include <Color.h>

Public Member Functions
- Color (float r=1, float g=1, float b=1, float a=1)  
  Construts a Color object default to white.
- Color (std::string color)  
  Construts a Color object with a string representing the color in hexadecimal.
- unsigned int toRGBA () const  
  Writes the color in a single 32 bit int.
- unsigned int toABGR () const  
  Writes the color in a single 32 bit int reversed.
- Color & operator*= (const Color &rhs)
- Color & operator/= (float value)
- Color & operator/= (const Color &rhs)
- Color & operator/= (float value)
- Color & operator+= (const Color &rhs)
- const Color operator* (const Color &rhs) const
- const Color operator* (float value) const
- const Color operator/ (const Color &rhs) const
- const Color operator/ (float value) const
- const Color operator+ (const Color &rhs) const
- bool operator== (const Color &rhs)
- bool operator!=(const Color &rhs)
- std::string toString ()
  Returns a string representation of the Color object.

Static Public Member Functions
- static unsigned int createRGBA (int r, int g, int b, int a)  
  Returns an unsigned int from color values.
- static unsigned int createABGR (int r, int g, int b, int a)  
  Returns an unsigned int from color values.

Public Attributes
- float r
- float g
- float b
- float a

Static Public Attributes
- static const Color black
- static const Color white
Detailed Description
Represents RGBA color.
Provides functionality to operate with colors.

Constructor & Destructor Documentation

Color::Color (std::string color)

Constructs a Color object with a string representing the color in hexadecimal.
The expected format of the string is like [x|#]RRGGBB[AA].

The documentation for this class was generated from the following files:
- D:/prj/rag/include/rag/Color.h
- D:/prj/rag/include/rag/Color.cpp


rag::Color4B Struct Reference

Color representation using 4 bytes.
#include <RenderTarget.h>

Public Attributes

- unsigned char r
- unsigned char g
- unsigned char b
- unsigned char a

Detailed Description

Color representation using 4 bytes.

The documentation for this struct was generated from the following file:
- D:/prj/rag/include/rag/RenderTarget.h
rag::DisplayObject Class Reference

Core object used to display things in screen.
#include <DisplayObject.h>
Inheritance diagram for rag::DisplayObject:

Public Member Functions

- void addChild (DisplayObject *child)
  Adds a DisplayObject child.
- void addChildAt (DisplayObject *child, int index)
  Adds a DisplayObject child in a specific order.
- void removeChild (DisplayObject *child)
  Removes a child DisplayObject.
- void deleteChild (DisplayObject *child)
  Deletes a specific child from the DisplayList.
- DisplayObject * getChildByName (const std::string &name)
  Returns a child by name.
- DisplayObject * getChildAt (int index)
  Returns a child by index.
- int getChildIndex (DisplayObject *child) const
  Returns a child's index.
- bool contains (const DisplayObject *child) const
  Returns true if the child DisplayObject exists as a child.
- int getNumChilds ()
  Returns the number of childs.
- void swapChildren (DisplayObject *child1, DisplayObject *child2)
  Swaps two childs indexes.
- virtual void logicUpdate ()
  This function is called every frame.
- virtual void logicTraversal ()
  Recursively calls logicUpdate in all child objects.
- virtual void render ()
  Renders the DisplayObject in the screen.
- virtual void renderTraversal (const Color &color)
  Recursively calls render in all childs objects.
- virtual void prerender ()
  Temporal transition to new automatic batch render.
- void setX (float x)
  Sets x position.
- float getX ()
  Returns x position.
- **void setY (float y)**
  
  Sets y position.

- **float getY ()**
  
  Returns y position.

- **void setPosition (float x, float y)**
  
  Sets x and y position.

- **void setPosition (glm::vec2 p)**
  
  Sets object position.

- **glm::vec2 getPosition () const**
  
  Returns object position.

- **void setPosition (glm::vec2 p)**
  
  Sets object position.

- **glm::vec2 getPosition () const**
  
  Returns object position.

- **void setScale (float scale)**
  
  Sets object scale.

- **void setScaleX (float scaleX)**
  
  Sets object x scale.

- **float getScaleX () const**
  
  Returns object x scale.

- **void setScaleY (float scaleY)**
  
  Sets object y scale.

- **float getScaleY () const**
  
  Returns object y scale.

- **void setAngle (float angle)**
  
  Sets object orientation angle.

- **float getAngle () const**
  
  Returns object orientation angle.

- **void setSkewX (float skewX)**
  
  Sets object x skew.

- **float getSkewX () const**
  
  Returns object x skew.

- **void setSkewY (float skewY)**
  
  Sets object y skew.

- **float getSkewY () const**
  
  Returns object y skew.

- **void setWidth (float width)**
  
  Sets object width.

- **float getWidth ()**
  
  Returns object width.

- **void setHeight (float height)**
  
  Sets object width.

- **float getHeight ()**
  
  Returns object width.

- **virtual bool hitTestPoint (int x, int y)**
  
  Returns true if the point lies inside the object boundary box.

- **glm::vec2 localToGlobal (const glm::vec2 &point)**
  
  Converts local coordinates to global coordinates.

- **glm::vec2 globalToLocal (const glm::vec2 &point)**
  
  Converts global coordinates to local coordinates.
• **void setClipRectangle** (const Rectangle &rect)
  This allows to render just a part of the bitmap.

• **Rectangle getBounds** (DisplayObject *targetCoordinateSpace=NULL)
  Returns the boundary box of the object.

• **void onNativeEvent** (events::TouchEvent &event)
  Notifies the object about an input event.

• **void setText** (rag::DisplayObject *displayObject, const std::string &text)
  Helper function to set a text.

• **void setText** (rag::DisplayObject *displayObject, int value)
  Helper function to set a text number.

• **void destroy** ()
  self-destroy the object and all its childs.

• **void updateMatrix** ()

**Static Public Member Functions**

• static **void deletePendentObjects** ()
  Deletes from memory all the nodes currently on the toDelete list.

• static **void showLivingObjects** ()
  Logs information about the current number of living DisplayObject.

**Public Attributes**

• std::string **name**
  Name of the object.

• DisplayObject * **parent**
  Parent DisplayObject.

• std::vector< DisplayObject *> * **childs**
  List of children.

• **Color color**
  Color of the object.

• **Material material**
  Material of the object.

• **Material * renderMaterial**
  The render material modified by the display list hierarchy.

• glm::mat4 **matrix**
  The object matrix.

• bool **visible**
  Determines object visibility.

• bool **autoScaleOnTouch**
  When true, the bounds scale when touched.

• bool **captureInput**
  When true, input events are captured and propagation stops.

• bool **checkHitPoint**
  When captureInput, checkHitPoint makes capture input only when hitTest is true. Defaults to false.

• std::string **soundName**

• std::string **script**

**Protected Member Functions**

• virtual **void updateBounds** (rag::DisplayObject *targetCoordinateSpace)
Updates the bounding box of the object according to childs bounds.

Protected Attributes

- float scaleX
- float scaleY
- float x
- float y
- float width
- float height
- float angle
- float skewX
- float skewY
- Rectangle bounds
- int numChilds
- bool dirty
- Color colorTransform
- Rectangle clipRect

Detailed Description

Core object used to display things in screen.

DisplayObject represents a node in a tree, and can have one or many DisplayObject children. Custom objects can inherit DisplayObject and override its basic functionality. EventDispatcher is extended for convenience, thus allowing to easily work with events.

Member Function Documentation

**void** DisplayObject::deleteChild (rag::DisplayObject * child)

Deletes a specific child from the DisplayList.

The child object is destroyed.

**void** DisplayObject::deletePendentObjects () [static]

Deletes from memory all the nodes currently on the toDelete list.

Don't call more than once per frame

**void** DisplayObject::destroy ()

self-destroy the object and all its childs.

Objects are usually deleted if they are part of the displaylist. If they're not, you can still delete their hierarchy by calling destroy.

**void** DisplayObject::logicTraversal () [virtual]
Recursively calls logicUpdate in all child objects.
This should be called once every frame in the root object.

```cpp
void DisplayObject::onNativeEvent (events::TouchEvent & event)
```

Notifies the object about an input event.

**Todo:**
This API should be moved to an input UI panel

```cpp
void DisplayObject::removeChild (DisplayObject * child)
```

Removes a child `DisplayObject`.
The child reference stops being child of the `DisplayObject`. If the child doesn’t exists then nothing is done.

**See also:**
`deleteChild()`

```cpp
void DisplayObject::updateBounds (rag::DisplayObject * targetCoordinateSpace)[protected], [virtual]
```

Updates the bounding box of the object according to childs bounds.
Is not required to call this function directly.
Reimplemented in spine::SkeletonDrawable (p.58).

```cpp
void DisplayObject::updateMatrix ()
```

Updates internal matrix from values such as parent matrix, position, scale and skew. Usually you don't need to call this directly.

---

**Member Data Documentation**

```cpp
bool rag::DisplayObject::autoScaleOnTouch
```

When true, the bounds scale when is touched.

**Todo:**
autoScale is weird, the only thing taht should autoscale should be buttons. Actually buttons should also by default captureInput and checkhitpoint.

```cpp
std::string rag::DisplayObject::soundName
```

**Todo:**
This should be part of a button.
The documentation for this class was generated from the following files:

- D:/prj/rag/include/rag/DisplayObject.h
- D:/prj/rag/include/rag/DisplayObject.cpp
rag::DropShadowFilter Class Reference

Shadow effect for TextField instances.
#include <TextField.h>

Public Member Functions
• DropShadowFilter (float angle, float distance, float strength, Color color)

Public Attributes
• Color color
• float x
• float y

Detailed Description
Shadow effect for TextField instances.

The documentation for this class was generated from the following files:
• D:/prj/rag/include/rag/TextField.h
• D:/prj/rag/include/rag/TextField.cpp
rag::Ease Class Reference

Collection of code-generated curves useful to create procedural tween animations.
#include <Ease.h>

Public Types
- enum EaseType { linear_01, quadIn_01, quadOut_01, quadInOut_01, cubicIn_01, cubicOut_01, cubicInOut_01, quartIn_01, quartOut_01, quartInOut_01, quintIn_01, quintOut_01, quintInOut_01, expoIn_01, expoOut_01, expoInOut_01, sineIn_01, sineOut_01, sineInOut_01, circIn_01, circOut_01, circInOut_01, backIn_01, backOut_01, backInOut_01, bounceIn_01, bounceOut_01, bounceInOut_01, elasticIn_01, elasticOut_01, elasticInOut_01, sinPi2_00, acelBreak_01, cos2Pi_01, sin2Pi_00, sinPi_00, sinPi2Pi_10, sin4Pi_00, sin3Pi4_00 }; The types of curve supported.

Public Member Functions
- Ease (EaseType myType=linear_01) Constructs an Ease curve.
- EaseType getType () Returns the current type.
- void setType (EaseType myType) Sets the current EaseType.
- float get (float t, float d) Returns the y coordinate of the curve for a given point t in a curve of length d.

Public Attributes
- EaseType type
- float PIF

Detailed Description
Collection of code-generated curves useful to create procedural tween animations.

The documentation for this class was generated from the following file:
- D:/prj/rag/include/rag/Ease.h
events::Event Class Reference

Base class for event system.
#include <Event.h>
Inheritance diagram for events::Event:

Public Member Functions
- **Event**(std::string type)
  Creates a new event of the given type.
- **virtual std::string toString()**
  string representing the event.

Public Attributes
- std::string **type**
  The type of the event. The string should be unique for this event.
- bool **captured**
  When an event is captured, it won't propagate anymore through the Display List.
- rag::DisplayObject * **target**
  Usually points to the dispatcher object. 'target' can be assigned to anything for custom events.

Detailed Description

Base class for event system.

When an event is triggered, it is processed this way: The display list is travelled from leaves to stage (AKA root). The propagation stops when the root is reached or when a node with the property 'captureInput' set to true is traversed.

See also:
- EventListener, EventDispatcher

The documentation for this class was generated from the following file:
- D:/prj/rag/include/rag/Event.h
events::EventDispatcher Class Reference

Base class used to dispatch events.
#include <EventDispatcher.h>
Inheritance diagram for events::EventDispatcher:

Public Member Functions

- void addEventListener (std::string type, EventListener *listener)
- void dispatchEvent (Event &event)
- bool hasEventListener (std::string type)
- void removeEventListener (std::string type, EventListener *listener)

Detailed Description

Base class used to dispatch events.

See also:

Event, EventListener

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

- D:/prj/rag/include/rag/EventDispatcher.h
- D:/prj/rag/include/rag/EventDispatcher.cpp
events::EventListener Class Reference

Interface that allows to listen events.
#include <EventListener.h>

Inheritance diagram for events::EventListener:

Public Member Functions

- virtual void onEvent (const std::string &type, events::Event &event)=0

Detailed Description

Interface that allows to listen events.

See also:
  Event, EventDispatcher

The documentation for this class was generated from the following file:
- D:/prj/rag/include/rag/EventListener.h
File multiplatform abstraction to read contents of a file.
#include <File.h>

Public Member Functions
- **File** (const std::string &path, bool bundle=true, bool logEnabled=true)
  Creates a File object.
- bool **open** (std::string mode="rb", bool showError=true)
  Open the file.
- void **close** ()
  Close the file.
- size_t **read** (void *buffer, size_t count)
  Read into buffer the number of 'count' bytes.
- long **getSize** ()
  Returns the size of the file.
- size_t **write** (const void *ptr, size_t size, size_t count)
  Writes into the file.
- bool **exists** ()
  Returns true if the file exists.
- const std::string & **getFullPath** ()
  Returns the full path of the file, may contain bundle folder.

Static Public Member Functions
- static std::string **load** (std::string filename, bool bundle=true, std::string mode="rb", bool showError=true)
  Convenient function to load files without dealing with low level api.
- static bool **existsPath** (const std::string &path)
  Returns true if the path exists.
- static bool **makePath** (const std::string &path)
  Creates a folder.
- static void **clearPatchFiles** ()
  Clean overridden files in bundle.
- static void **setPatchFile** (const std::string &filename, const std::string &filepath)
  Override files in bundle.
- static const std::map< std::string, std::string > & **getPatchFiles** ()
  Returns overridden files in bundle.

Protected Attributes
- FILE * **pFile**
- long **size**
- std::string **path**
- std::string **osPath**
- bool **bundle**
Static Protected Attributes

- static bool $\text{PatchFilesLoaded}$
- static std::map<std::string, std::string> $\text{patchFiles}$

Detailed Description

File multiplatform abstraction to read contents of a file.

Constructor & Destructor Documentation

File::File (const std::string & $\text{path}$, bool $\text{bundle} = \text{true}$, bool $\text{logEnabled} = \text{true}$)

Creates a File object.

Parameters:

<table>
<thead>
<tr>
<th>path</th>
<th>The path where the file can be found.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{bundle}$</td>
<td>If the file is inside the bundle. The bundle is the package created at build-time.</td>
</tr>
<tr>
<td>$\text{logEnabled}$</td>
<td></td>
</tr>
</tbody>
</table>

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

- D:/prj/rag/include/rag/File.h
- D:/prj/rag/include/rag/File.cpp
rag::MovieClip::Frame Struct Reference

Internal of MovieClip, represents a single frame.
#include <MovieClip.h>

Public Attributes
- int index
- int duration
- std::string label
- DisplayObject * bitmap

Detailed Description
Internal of MovieClip, represents a single frame.

The documentation for this struct was generated from the following file:
- D:/prj/rag/include/rag/MovieClip.h
rag::Image Class Reference

Image object.
#include <Image.h>

Inheritance diagram for rag::Image:

Public Member Functions
- **Image** (const std::string &path="", int textureWrapMode=GL_CLAMP_TO_EDGE, bool deleteImageData=true, bool downloaded=false)
  Returns a functional image with size, the image is loaded in background.
- virtual ~Image ()
  Default destructor.
- virtual void loadInBackground () override
  CPU intensive load goes here.
- virtual void loadSync () override
  The part of the loading that must be done in main thread.
- void reload ()
  on context lost, images can be reloaded.

Static Public Member Functions
- static **Image** * loadImage (const std::string &path="", int textureWrapMode=GL_CLAMP_TO_EDGE, bool deleteImageData=true, bool downloaded=false)
  Returns a functional image with size, the image is loaded in background.
- static void setCompressedFolder (std::string folder)
  Adds a compressed folder.
- static void clearCompressedFolders ()
  Clears all compressed folders.

Public Attributes
- int width
- int height
- int pixelFormat
- GLuint name
- GLubyte * bytes

Static Public Attributes
- static int s_memorySize = 0

Friends
- class ImageLoaderJPG
Additional Inherited Members

Detailed Description

Image object.

An image represents a 2D texture, usually readed from a specific file on disk. Images can be drawn by Bitmap instances. Image raw data can be read and/or manipulated.

Constructor & Destructor Documentation

Image::Image (const std::string & path = "", int textureWrapMode = GL_CLAMP_TO_EDGE, bool deleteImageData = true, bool downloaded = false)

Returns a functional image with size, the image is loaded in background.

If you want direct access to image in raw format, you need to specify deleteImageData = false, otherwise image data is deleted.

Member Function Documentation

Image * Image::loadImage (const std::string & path = "", int textureWrapMode = GL_CLAMP_TO_EDGE, bool deleteImageData = true, bool downloaded = false)[static]

Returns a functional image with size, the image is loaded in background.

Asks resource manager for the image, create it if not exists.

void Image::setCompressedFolder (std::string folder)[static]

Adds a compressed folder.

All pngs and jpps loaded inside a compressed folder are converted to 16bpp images in memory.

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

- D:/prj/rag/include/rag/Image.h
- D:/prj/rag/include/rag/Image.cpp
**rag::ImageLoader Class Reference**

Interface to load images.

```cpp
#include <ImageLoader.h>
```

Inheritance diagram for `rag::ImageLoader`:

```
  +--- ImageLoader
     |    
  |    +-- ImageLoaderJPG
  |         +-- ImageLoaderPNG
  |             +-- ImageLoaderPVR
```

### Public Member Functions

- virtual bool `loadInfo()` = 0  
  *Load header to know image size.*

- virtual bool `loadImage()` = 0  
  *Load image from a file.*

- virtual ~ImageLoader()  
  *Default destructor.*

### Detailed Description

Interface to load images.

The documentation for this class was generated from the following file:
- D:/prj/rag/include/rag/ImageLoader.h
rag::ImageLoaderJPG Class Reference

Loader for .jpg format.
#include <ImageLoaderJPG.h>
Inheritance diagram for rag::ImageLoaderJPG:

Public Member Functions
- ImageLoaderJPG (const std::string &name, rag::Image *image)
- virtual bool loadInfo () override
  Load header to know image size.
- virtual bool loadImage () override
  Load image from a file.

Detailed Description
Loader for .jpg format.

The documentation for this class was generated from the following files:
- D:/prj/rag/include/rag/ImageLoaderJPG.h
- D:/prj/rag/include/rag/ImageLoaderJPG.cpp
rag::ImageLoaderPNG Class Reference

Loader for .png format.
#include <ImageLoaderPNG.h>

Inheritance diagram for rag::ImageLoaderPNG:

- Public Member Functions
  - ImageLoaderPNG (const std::string &name, rag::Image *image)
  - bool loadInfo ()
    Load header to know image size.
  - bool loadImage ()
    Load image from a file.

- Public Attributes
  - size_t byte
  - unsigned char * buffer

Detailed Description

Loader for .png format.

The documentation for this class was generated from the following files:
- D:/prj/rag/include/rag/ImageLoaderPNG.h
- D:/prj/rag/include/rag/ImageLoaderPNG.cpp
rag::ImageLoaderPVR Class Reference

Loader for .pvr compressed format.
#include <ImageLoaderPVR.h>
Inheritance diagram for rag::ImageLoaderPVR:

Public Member Functions

- **ImageLoaderPVR** (const std::string &name, rag::Image *image)
- virtual bool **loadInfo** () override
  Load header to know image size.
- virtual bool **loadImage** () override
  Load image from a file.

Detailed Description

Loader for .pvr compressed format.

The documentation for this class was generated from the following files:
- D:/prj/rag/include/rag/ImageLoaderPVR.h
- D:/prj/rag/include/rag/ImageLoaderPVR.cpp
rag::InputManager Class Reference

Simple Input Manager.
#include <InputManager.h>

Public Member Functions
- void processInputEvent (events::TouchEvent &e)
  Introduces an input event in the system.
- void lock ()
  Prevents input to be processed.
- void unlock ()
  After lock(), returns to normal operation.
- bool isLocked ()
  Returns true if input is currently locked.
- void traverse (rag::DisplayObject *root)
  Traverses Display List recursively.

Static Public Member Functions
- static InputManager & getInstance ()
  Returns the shared instance of the InputManager.

Detailed Description
Simple Input Manager.

Member Function Documentation

bool rag::InputManager::isLocked ()

Returns true if input is currently locked.

See also:
  lock(), unlock()

The documentation for this class was generated from the following files:
- D:/prj/rag/include/rag/InputManager.h
- D:/prj/rag/include/rag/InputManager.cpp
rag::ITextFont Class Reference

Interface for text fonts.
#include <ITextFont.h>

Inheritance diagram for rag::ITextFont:

Public Member Functions

- virtual int getWidth (const std::string &text)=0
  Returns the width of a text.
- virtual void print (const std::string &text, const glm::mat4 &matrix)=0
  Renders text. Assumes ortho projection 1:1 screen pixel.
- virtual void setLetterSpacing (float value)
- virtual void reloadTexture ()
  Sets the extra space between characters.

Detailed Description

Interface for text fonts.

The documentation for this class was generated from the following file:
- D:/prj/rag/include/rag/ITextFont.h
rag::Keyboard Class Reference

Multiplatform keyboard abstraction.
#include <Keyboard.h>
Inheritance diagram for rag::Keyboard:

Public Types
- enum KeyboardType { KeyboardTypeDefault = 0, KeyboardTypeEmail }

Public Member Functions
- Keyboard (KeyboardType type=KeyboardTypeDefault)
  Default constructor.
- virtual ~Keyboard ()
  Default destructor.
- void show ()
  Shows the native keyboard.
- void hide ()
  Hides the native keyboard.
- virtual void onEvent (const std::string &type, events::Event &event)
  Keyboard implements EventListener. Here is where is listening native events.

Static Public Attributes
- static rag::Rectangle size
  The size of the native keyboard rectangle.

Additional Inherited Members

Detailed Description
Multiplatform keyboard abstraction.
Once created, a Keyboard instance will open a native keyboard in the device and will dispatch keyboard events.

See also:
  events::KeyboardEvent.
Constructor & Destructor Documentation

Keyboard::Keyboard (KeyboardType type = KeyboardTypeDefault)

Default constructor.
Creates the keyboard instance. You need to call show() to see the native keyboard.

See also:
    show()
events::KeyboardEvent Class Reference

Event to handle Keyboard actions.
#include <KeyboardEvent.h>
Inheritance diagram for events::KeyboardEvent:

Public Member Functions
- KeyboardEvent (std::string type)
- virtual std::string toString ()
  String representation of the event.

Public Attributes
- std::string key
  Key pressed, encoded in UTF-8.
- int charCode
  Character code.

Detailed Description
Event to handle Keyboard actions.

The documentation for this class was generated from the following file:
- D:/prj/rag/include/rag/KeyboardEvent.h
rag::KeyboardManager Class Reference

Singleton class that dispatches Keyboard events.
#include <KeyboardManager.h>

Inheritance diagram for rag::KeyboardManager:

```
+------------------+
| events::EventDispatcher |
|                   |
+------------------+
                  |
|                 |
|  |                  |
|  |                  |
+------------------+
                  |
|                 |
|  | rag::KeyboardManager |
+------------------+
```

Public Member Functions

- void nativeInsertText (std::string text)
  When called, will generate a standard multi-platform KeyboardEvent.
- void nativeDeleteBackward ()
  When called, will generate a standard multi-platform KeyboardEvent.

Static Public Member Functions

- static KeyboardManager & getInstance ()

Static Public Attributes

- static const int RETURN_KEYBOARD_CODE = 10

Detailed Description

Singleton class that dispatches Keyboard events.

Todo:
It's confusing to have a Keyboard and a KeyboardManager.

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

- D:/prj/rag/include/rag/KeyboardManager.h
- D:/prj/rag/include/rag/KeyboardManager.cpp
Material Class Reference

Represents a render material that would affect how objects will be rendered.
#include <Material.h>

Public Types
- enum EBlendMode { BLEND_NORMAL, BLEND_ADD, BLEND_PRE_ADD }

Public Member Functions
- Material (const std::string &vsh, const std::string &fsh)
  Default constructor using two paths for vertex and fragment shader respectively.
- virtual ~Material ()
  Default destructor.
- void setUniform (const std::string &name, float value)
  Set float shader uniform.
- void setUniform (const std::string &name, const glm::vec2 &value)
  Set vec2 shader uniform.
- void setUniform (const std::string &name, const glm::vec3 &value)
  Set vec3 shader uniform.
- void setUniform (const std::string &name, const glm::vec4 &value)
  Set vec4 shader uniform.
- void setUniform (const std::string &name, const glm::mat3 &value)
  Set mat3 shader uniform.
- void setUniform (const std::string &name, const glm::mat4 &value)
  Set mat4 shader uniform.
- const std::vector<TUniformVar> &getUniforms () const
  Returns the list of all uniforms related with the current shader program.

Public Attributes
- std::string vsh
  Path for vertex shader.
- std::string fsh
  Path for fragment shader.
- int textureId
  Texture id.
- EBlendMode blendMode
  Blending mode.
- int priority
  Materials with higher priority are propagated down in the Display List.
- bool stopsPropagation
  If true, no matter priorities, material won’t be affected by parent materials.

Friends
- class Renderer
Detailed Description

Represents a render material that would affect how objects will be rendered.

A Material is based in a vertex and fragment Shader. A texture is linked to the material, and a blend mode. Everything will contribute to the final result when the object is rendered.

Todo:

- Make a clearer design taking into account efficiency and naming convention. Refactor classes like Program, Shader. May make sense to allow change shader uniforms in Shader class instead of Material. DisplayObjects shold use Material references, and then have the ability to make copies if they need to modify something, (e.g., change shader uniforms).

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

- D:/prj/rag/include/rag/Material.h
- D:/prj/rag/include/rag/Material.cpp
rag::MovieClip Class Reference

Allows to use imported animations created by Flash CS tool.
#include <MovieClip.h>
Inheritance diagram for rag::MovieClip:

```
#include <MovieClip.h>

Inheritance diagram for rag::MovieClip:
```

Classes
- struct Frame

**Internal of MovieClip, represents a single frame.** Public Member Functions
- MovieClip (std::vector< Frame > > frames)
- int getCurrentFrame ()
  Specifies the number of the frame in which the playhead is located in the timeline of the MovieClip instance.
- std::string getCurrentFrameLabel ()
  Returns the label in the current frame. It may be empty.
- int getTotalFrames ()
  Returns the total number of frames in the MovieClip.
- void play ()
  Simple playback.
- void stop ()
  Stops the playhead in the movie clip.
- void gotoAndPlay (int frame)
  Goes to a specific frame, then starts playing from there.
- void gotoAndPlay (const std::string &frame, bool loop, bool forceFirstFrame=false)
  Goes to a specific frame, then starts playing from there.
- void gotoAndStop (int frame)
  Goes to a specific frame, then stops there.
- void gotoAndStop (const std::string &frame)
  Goes to a specific frame by name, then stops there.
- void nextFrame ()
  Sends the playhead to the next frame and stops it.
- void prevFrame ()
  Sends the playhead to the previous frame and stops it.
- Frame * getCurrentFrameNode ()
  Returns the current frame internals.
- virtual void logicUpdate () override
  Display object update.
- void setFPS (int fps)
Sets the speed in frames per second the MovieClip should use.

- void replace (const std::string &name, const std::string &library, const std::string &replacement)
  Replace an instance of a DisplayObject named "name" with a library item called "replacement".

- bool isPlaying ()
  Returns true if is currently playing.

- std::vector<Frame> & getFrames ()
  Returns all Frame instances.

Public Attributes

- std::string fileName
  MovieClip filename.

Additional Inherited Members

Detailed Description

Allows to use imported animations created by Flash CS tool.

Constructor & Destructor Documentation

MovieClip::MovieClip (std::vector< Frame > _frames)

TODO: MovieClip way of handle labels. Labels are used for loops, but should be used only as extra information. TODO: Imitate flash way of do stuff with Movieclips and add an extra layer (outside MovieClip) to handle animations and loops in a convenient way.

Member Function Documentation

void MovieClip::play ()

Simple playback.
Moves the playhead in the timeline of the movie clip.

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

- D:/prj/rag/include/rag/MovieClip.h
- D:/prj/rag/include/rag/MovieClip.cpp
rag::fs::path Class Reference

Mimics boost fs::path class with some limited functionality.
#include <File.h>

Public Member Functions
- path (const std::string &name="")
- path parent_path ()
- path extension () const
- bool has_parent_path ()
- bool has_extension () const
- path & replace_extension (const std::string extension)
- path filename () const
- std::string string () const
- const path operator/ (const path &rhs) const

Detailed Description
Mimics boost fs::path class with some limited functionality.

The documentation for this class was generated from the following file:
- D:/prj/rag/include/rag/File.h
rag::Program Struct Reference

Represents a Shader Program.
#include <Renderer.h>

Public Attributes
- std::string vsh
  *Path to the vertex shader.*
- std::string fsh
  *Path to the fragment shader.*
- GLuint handle
  *Internal GL handle.*
- bool usesVertexArray
- bool usesColorArray
- bool usesTexCoordArray

Detailed Description
Represents a Shader Program.

The documentation for this struct was generated from the following file:
- D:/prj/rag/include/rag/Renderer.h
rag::Rectangle Class Reference

Represents a Rectangle.
#include <Rectangle.h>

Public Member Functions

- **Rectangle** (float x=0, float y=0, float width=0, float height=0)
  Constructor with coordinates and size.
- **Rectangle rectUnion** (const Rectangle &toUnion) const
  Returns the union of the current instance with another Rectangle.
- **Rectangle intersection** (const Rectangle &toIntersect) const
  Returns the intersection of the current instance with another Rectangle.
- **bool intersects** (const Rectangle &toIntersect) const
  Returns true if the instance intersects with the other Rectangle.
- **bool contains** (float x, float y) const
  Checks if a point lies inside the Rectangle.
- **bool contains** (const glm::vec2 &point) const
  Checks if a point lies inside the Rectangle.
- **bool contains** (const Rectangle &rect) const
  Checks if a Rectangle lies inside the Rectangle.
- **bool operator==** (const Rectangle &other) const
- **bool operator!=** (const Rectangle &other) const
- **std::string toString ()**
  String representation of the Rectangle.

Public Attributes

- float x
- float y
- float width
- float height

Detailed Description

Represents a Rectangle.

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

- D:/prj/rag/include/rag/Rectangle.h
- D:/prj/rag/include/rag/Rectangle.cpp
rag::Renderer Class Reference

Contains methods to render objects.
#include <Renderer.h>

Public Types

- enum EMaterialAttributes { ATTRIB_VERTEX = 0, ATTRIB_TEXTURECOORD, ATTRIB_COLOR, ATTRIB_NORMAL, ATTRIB_TANGENT, ATTRIB_BONES_INDICES, ATTRIB_BONES_WEIGHTS, NUM_ATTRIBUTES }

Public Member Functions

- void init ()
- int loadProgram (const char *vsh, const char *fsh)
  Loads and starts using a program with the given shaders.
- void bindVertexArray (void *array, int size=2, int stride=0)
  Binds a vertex array that would be used for the next draw call.
- void bindTextureArray (void *array, int size=2, int stride=0)
  Binds a texture array that would be used for the next draw call.
- void bindColorArray (void *array, int channels=4, int stride=0)
  Binds a color array that would be used for the next draw call.
- int bindMaterial (rag::Material *material)
  Binds the Material to be used in the next draw call.
- void bindTexture (int textureName)
  Binds a texture by id.
- void setBlendFunc (GLenum source, GLenum dest)
  Sets blending function.
- void setClearColor (Color color)
  Set clear color.
- void bindBuffer (int target, int buffer)
- void createVertexBuffer (GLuint &vboID)
- void deleteVertexBuffer (GLuint &vboID)
- int getCurrentProgramHandle ()
  Returns the handle of the program used currently.
- void checkError ()
  Displays an error if something’s wrong.
- void precompileShader (const std::string &path)
  Precompiles a Shader, useful to call it in loading times.
- glm::mat4 getOrthoProjection ()
  Returns an orthographic projection.
- void clearShaders ()
  Forget shaders and programs loaded. Programs will be generated again as needed.
- rag::RenderTarget &getRenderTarget ()
  TODO Pass render target as parameter to render() method in DisplayObject - so that it becomes more obvious where are you supposed to draw in.
Static Public Member Functions

- static Renderer & getInstance ()

Detailed Description

Contains methods to render objects.

Most of the functionality in this class, is handled like a state machine, like OpenGL does. Setting states would left that states changed until other change is done. This applies for all the bind() functions.

Member Function Documentation

int rag::Renderer::bindMaterial (rag::Material * material)

Binds the Material to be used in the next draw call.

bindMaterial() changes internal states of the renderer and prepares a context to make the render draw call.

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

- D:/prj/rag/include/rag/Renderer.h
- D:/prj/rag/include/rag/Renderer.cpp
**rag::RenderTarget Class Reference**

Object where **DisplayObject** instances with render capability are supposed to render.

```
#include <RenderTarget.h>
```

**Public Member Functions**

- **RenderTarget ()**
  
  *Create a **RenderTarget** object.*

- virtual **~RenderTarget ()**
  
  *Default destructor.*

- void **draw** (const VertexArray &vertexArray, const **Material** &material)
  
  *Enqueues draw command.*

- void **flush ()**
  
  *Renders enqueued render commands.*

---

**Detailed Description**

Object where **DisplayObject** instances with render capability are supposed to render.

A **RenderTarget** is passed through the Display List and provides functionality to enqueue render commands. **RenderTarget** will group commands that use similar **Material** in order to reduce the number of draw calls.

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

- D:/prj/rag/include/rag/RenderTarget.h
- D:/prj/rag/include/rag/RenderTarget.cpp
rag::Resource Class Reference

Abstract class that represents a game Resource, typically something costly to load.
#include <ResourceManager.h>

Inheritance diagram for rag::Resource:

```
rag::Resource
  ↓
rag::Image
```

Public Types

- enum State { Enqueued = 0, LoadingInBackground, BackgroundLoaded, Ready } List of possible states for a Resource.

Public Member Functions

- virtual void loadInBackground ()=0
  
  CPU intensive load goes here.
- virtual void loadSync ()=0
  
  Load that must be synchronized with the main thread.
- void acquire ()
  
  Prevents the Resource to be deleted until it's release()'d.
- void release ()
  
  Releases the Resource, so it can be deleted.

Public Attributes

- State state
  
  Current state.
- std::string resourceName
  
  Resource unique name. Usually the file name.
- int memorySize
  
  Size of the Resource in memory measured in bytes.

Friends

- class ResourceManager

Detailed Description

Abstract class that represents a game Resource, typically something costly to load.

Resources are treated as if they were memory and CPU intensive. There are methods to load asynchronously a Resource.

See also:

- ResourceManager

The documentation for this class was generated from the following file:
rag::ResourceMgr Class Reference

Handles Resource management, including loading and unloading Resource instances.
#include <ResourceMgr.h>

Public Member Functions
- void update ()
  Do the tasks for this frame. Must be called each frame.
- Resource * getResource (const std::string &name)
  Returns a resource by name. Will return a null pointer if the resource doesn't exists.
- void loadResource (Resource *resource, bool inBackground=true)
  Starts loading a Resource.
- void unload (const std::string &resourceName)
  Free memory allocated by a given Resource by name.
- void reload ()
  On context lost.
- void dumpResources (const std::string &extension="*")
  Display the Resource instances currently loaded in memory.
- int numPendingResources ()
  Number of Resource instances waiting to be loaded.
- void clean ()
  Cleans all Resources with no users.

Static Public Member Functions
- static ResourceMgr & instance ()
  Returns the shared instance of the manager.

Public Attributes
- int memoryWarnings
  For iOS, the number of memory warnings given by the OS.

Detailed Description
Handles Resource management, including loading and unloading Resource instances.

Member Function Documentation
void ResourceMgr::loadResource (Resource * resource, bool inBackground = true)

Starts loading a Resource.
The Resource can be loaded synchronous or asynchronously.
Parameters:

| inBackground | If true the Resource will be loaded in a background thread. |

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

- D:/prj/rag/include/rag/ResourceMgr.h
- D:/prj/rag/include/rag/ResourceMgr.cpp
rag::Screen Class Reference

Contains information about the current device.
#include <Screen.h>

**Static Public Attributes**
- static int `width` = 0
  *Virtual width of the screen. Shared across all devices.*
- static int `height` = 0
  *Virtual height of the screen. Shared across all devices.*
- static glm::vec2 `center`
  *Virtual center of the screen. Shared across all devices.*
- static int `realWidth` = 0
  *Actual number of pixels of width in the screen.*
- static int `realHeight` = 0
  *Actual number of pixels of height in the screen.*
- static float `scale` = 0
  *Apple scale.*
- static float `scaleBest` = 0
  *Maximum scale where no deformation occurs.*
- static float `factorX` = 0
  *Factor used to maintain screen width coordinates independent from resolution or aspect ratio.*
- static float `factorY` = 0
  *Factor used to maintain screen height coordinates independent from resolution or aspect ratio.*
- static bool `isTablet` = false
  *True if the device is considered a table instead of a phone.*
- static bool `isLowPerformer` = false
  *True for low end devices.*
- static bool `isLowRes` = false
  *True for devices with low resolution screens.*

---

**Detailed Description**
Contains information about the current device.

The documentation for this class was generated from the following files:
- D:/prj/rag/include/rag/Screen.h
- D:/prj/rag/include/rag/Screen.cpp
rag::Shader Struct Reference

Represents a GL Shader.
#include <Renderer.h>

Public Attributes

- GLuint handle
- std::string file

Detailed Description

Represents a GL Shader.

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

- D:/prj/rag/include/rag/Renderer.h
**spine::SkeletonDrawable Class Reference**

Display 2D Skeletal Animations made with the 3rd party tool Spine.

`#include <Spine.h>`

Inheritance diagram for spine::SkeletonDrawable:

```
events::EventDispatcher

rag::DisplayObject

spine::SkeletonDrawable
```

**Public Member Functions**

- `SkeletonDrawable (spSkeletonData *skeleton, spAnimationStateData *stateData=0)`
- `virtual void logicUpdate () override`
  
  *This function is called every frame.*
- `virtual void render () override`
  
  *Renders the DisplayObject in the screen.*
- `virtual void updateBounds (rag::DisplayObject *targetCoordinateSpace) override`
  
  *Updates the bounding box of the object according to childs bounds.*

**Public Attributes**

- `rag::VertexArray vertexArray`
- `spSkeleton * skeleton`
- `spAnimationState * state`
- `spSkeletonBounds * skeletonBounds`
- `float timeScale`

**Additional Inherited Members**

---

**Detailed Description**

Display 2D Skeletal Animations made with the 3rd party tool Spine.

---

**Member Function Documentation**

```
void spine::SkeletonDrawable::updateBounds (rag::DisplayObject *targetCoordinateSpace)[override], [virtual]
```

Updates the bounding box of the object according to childs bounds.

Is not required to call this function directly.

Reimplemented from `rag::DisplayObject (p.20)`.
The documentation for this class was generated from the following files:

- D:/prj/rag/include/rag/Spine.h
- D:/prj/rag/include/rag/Spine.cpp
rag::TextEdit Class Reference

Creates a native window to edit a text.
#include <TextEdit.h>

Inheritance diagram for rag::TextEdit:

```
events::EventDispatcher

rag::DisplayObject

rag::TextEdit
```

Public Member Functions

- TextEdit (rag::DisplayObject *parent, const std::string &text)
- void close (bool cancel=false)
- const std::string &getValue ()

Static Public Member Functions

- static TextEdit *getInstance ()

Additional Inherited Members

---------------------------

Detailed Description

Creates a native window to edit a text.

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

- D:/prj/rag/include/rag/TextEdit.h
- D:/prj/rag/include/rag/TextEdit.cpp
Dispatched by InputText when user writes one character.

#include <TextEvent.h>

Inheritance diagram for events::TextEvent:

Public Member Functions
- TextEvent (std::string type)

Public Attributes
- std::string text
- std::string lastCharacter

Detailed Description
Dispatched by InputText when user writes one character.

The documentation for this class was generated from the following file:
- D:/prj/rag/include/rag/TextEvent.h
rag::TextField Class Reference

High level abstraction to render texts in display list.
#include <TextField.h>

Inheritance diagram for rag::TextField:

Public Types
- enum HorzAlignment { Left = 0, Center, Right }
- enum VertAlignment { Top = 0, Middle, Bottom }

Public Member Functions
- TextField (const std::string &folderPath, const std::string &descriptorFileName)
  Constructor with Bitmap fonts.
- TextField (const std::string &path, float pixelHeight=24, float letterSpacing=0)
  Constructor with truetype fonts.
- void addFilter (DropShadowFilter filter)
- std::vector<DropShadowFilter> getFilters ()
- int getLines ()
  Returns the number of lines used with the current text.
- int getLineHeight ()
  Returns the height of a line.
- int getTextWidth ()
  Returns the current length of the text, for single line.

Static Public Member Functions
- static void traceTextCache ()
  Debug function to know how many textures are cached by texts.
- static void reloadTextures ()
  When graphic context is missed (android) reloads fonts textures.

Public Attributes
- enum rag::TextField::HorzAlignment horzAlign
  The horizontal alignment of the text block.
- enum rag::TextField::VertAlignment vertAlign
  The vertical alignment of the text block.
- std::string text
  The text that should be rendered.
- bool multiline
Is it intended to be drawn in a single or multi-line fashion. False by default.

- bool showCursor
  When true, a cursor is shown right after the last letter. Note that text width remains the same with or without cursor.

- bool autotrim
  True by default. Trims single line text when longer than reserved dimensions.

- bool password
  Use the textfield to show a password. Wildcards would be printed instead of the actual text.

Protected Member Functions

- void init ()
- virtual void render () override
  Renders the DisplayObject in the screen.

- void printText (const std::string &text, const glm::mat4 matrix)
- std::vector< std::string > splitLines ()

Protected Attributes

- ITextFont * font
- int lineHeight
- int lineWidth
- std::vector< std::string > lines
- std::vector< int > lineWidths
- std::string lastText
- std::vector< DropShadowFilter > filters
- float letterSpacing

Detailed Description

High level abstraction to render texts in display list.

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

- D:/prj/rag/include/rag/TextField.h
- D:/prj/rag/include/rag/TextField.cpp
rag::TextInput Class Reference

Helper object to add input to a TextField.
#include <TextInput.h>
Inheritance diagram for rag::TextInput:

Public Member Functions
- TextInput (std::string defaultText="", int maxLines=0, int maxCharacters=0, bool useCaptureLayer=true, Keyboard::KeyboardType keyboardType=Keyboard::KeyboardTypeDefault)
- std::string getText ()
- void setText (const std::string &text)
- void openKeyboard ()
- void closeKeyboard ()
- void clearText ()

Additional Inherited Members

Detailed Description
Helper object to add input to a TextField.
System keyboard is shown when the user clicks the TextInput object.
You need to add a TextInput in a TextField to let user write on it, and TextField parent should have ButtonBehaviour.
The hierarchy for a button with text and textinput is like this:
button -> textfield -> textinput
where button is the grandfather of textinput.
CaptureLayer may be added, so when user clicks somewhere in the screen the keyboard is closed.

The documentation for this class was generated from the following files:
- D:/prj/rag/include/rag/TextInput.h
- D:/prj/rag/include/rag/TextInput.cpp
rag::Timer Class Reference

Provides time-related functionality.
#include <Timer.h>

Public Member Functions

- void start (float time, bool loop=false)
  Starts the Timer with a fixed amount of time.
- float getDelta (Ease::EaseType easetype=Ease::linear_01)
  Returns the time elapsed since the start interpolated between 0 and 1.
- bool finished ()
  True if the timer has been running for the time specified at the start or more. Only valid for non-loop operation mode.
- bool running ()
  True if the timer is running.
- void reset ()
  Reset stops the timer and puts it in the same state it was before start running.

Static Public Attributes

- static float deltaTime = 0
  Stores the elapsed time from frame to frame (use at your convenience).
- static float totalTime = 0
  Stores total time since app starts.
- static float timeFactor = 1.0f
  Factor shared by which all Timer instances.

Detailed Description
Provides time-related functionality.

Member Function Documentation

float Timer::getDelta (Ease::EaseType easetype = Ease::linear_01)

Returns the time elapsed since the start interpolated between 0 and 1.
The interpolated value can use any curve, for convenience exposed in this function.

The documentation for this class was generated from the following files:
- D:/prj/rag/include/rag/Timer.h
- D:/prj/rag/include/rag/Timer.cpp
events::TouchEvent Class Reference

Event for handle input from screen.
#include <TouchEvent.h>
Inheritance diagram for events::TouchEvent:

Public Member Functions
- TouchEvent (std::string type)
- virtual std::string toString ()
  string representing the event.

Public Attributes
- float localX
- float localY
- float stageX
- float stageY
- float movementX
- float movementY
- float pinch
- char touchId

Detailed Description
Event for handle input from screen.

The documentation for this class was generated from the following file:
- D:/prj/rag/include/rag/TouchEvent.h
rag::TTFFont Class Reference

Implementation of ITextFint based on TrueType or OpenType Fonts.

```cpp
#include <TTFFont.h>
```

Inheritance diagram for rag::TTFFont:

```
rag::ITextFont

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rag::TTFFont</td>
</tr>
<tr>
<td>----------------</td>
</tr>
</tbody>
</table>
```

Public Member Functions

- **TTFFont** (const std::string &path, float pixelHeight=24)
  
  Constructs a **TTFFont** using a path and a text size.

- virtual int **getWidth** (const std::string &text)
  
  Returns the width of a text.

- virtual void **print** (const std::string &text, const glm::mat4 &matrix)
  
  Renders text. Assumes ortho projection 1:1 screen pixel.

- virtual void **setLetterSpacing** (float value)
  
  Sets the extra space between characters.

- virtual void **reloadTexture** ()
  
  On context loss, reload textures. **ITextFont** should inherit **Resource**. Reload should be part of resource.

Static Public Member Functions

- static void **addFontAlias** (const std::string &alias, const std::string &fontPath)
  
  Allows to use a different name (or an 'alias') to refer to a font.

Detailed Description

Implementation of ITextFint based on TrueType or OpenType Fonts.

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

- D:/prj/rag/include/rag/TTFFont.h
- D:/prj/rag/include/rag/TTFFont.cpp
rag::TUniformVar Struct Reference

Public Attributes

- EUniformType type
- std::string name
- float value [16]

The documentation for this struct was generated from the following file:
- D:/prj/rag/include/rag/Material.h
rag::Vertex Struct Reference

Vertex representation.
#include <RenderTarget.h>

Public Attributes
- glm::vec2 position
- glm::vec2 texCoords
- Color4B color

Detailed Description
Vertex representation.

The documentation for this struct was generated from the following file:
- D:/prj/rag/include/rag/RenderTarget.h
rag::XFLBinaryParser Class Reference

Public Member Functions

- **rag::DisplayObject * load** (const std::string &rootFolder, const std::string &symbol)

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

- D:/prj/rag/include/rag/XFLParser.cpp
rag::XFLParser Class Reference

Parser of XFL documents generated by the 3rd party editor tool FlashCS.
#include <XFLParser.h>

Public Member Functions

- **XFLParser ()**
  
  Creates the parser object.

- **rag::DisplayObject * load (const std::string &rootFolder, const std::string &symbol)**
  
  Load a Flash CS symbol.

Detailed Description

Parser of XFL documents generated by the 3rd party editor tool FlashCS.

An XFLParser instance allows to read Flash symbols and creates the required DisplayObject instances to reproduce a particular symbol.

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

- D:/prj/rag/include/rag/XFLParser.h
- D:/prj/rag/include/rag/XFLParser.cpp
Index

autoScaleOnTouch
   rag::DisplayObject, 20
bindMaterial
   rag::Renderer, 50
Bitmap
   rag::Bitmap, 10
BMPFont
   rag::BMPFont, 11
Color
   rag::Color, 14
deleteChild
   rag::DisplayObject, 19
deletePendentObjects
   rag::DisplayObject, 19
destroy
   rag::DisplayObject, 19
events::Event, 24
events::EventDispatcher, 25
events::EventListener, 26
events::KeyboardEvent, 40
events::TextEvent, 61
events::TouchEvent, 66
File
   rag::File, 28
getDelta
   rag::Timer, 65
Image
   rag::Image, 31
isLocked
   rag::InputManager, 36
Keyboard
   rag::Keyboard, 39
loadImage
   rag::Image, 31
loadResource
   rag::ResourceMgr, 54
logicTraversal
   rag::DisplayObject, 19
MovieClip
   rag::MovieClip, 45
onNativeEvent
   rag::DisplayObject, 20
play
   rag::MovieClip, 45
rag::Bitmap, 9
   Bitmap, 10
   uv, 10
rag::BMPFont, 11
   BMPFont, 11
rag::Chrono, 12
rag::Color, 13
   Color, 14
rag::Color4B, 15
rag::DisplayObject, 16
   autoScaleOnTouch, 20
   deleteChild, 19
   deletePendentObjects, 19
   destroy, 19
   logicTraversal, 19
   onNativeEvent, 20
   removeChild, 20
   soundName, 20
   updateBounds, 20
   updateMatrix, 20
rag::DropShadowFilter, 22
rag::Ease, 23
rag::File, 27
   File, 28
rag::fs, 8
rag::fs::path, 46
rag::Image
   Image, 30
   Image, 31
   loadImage, 31
   setCompressedFolder, 31
rag::ImageLoader, 32
rag::ImageLoaderJPG, 33
rag::ImageLoaderPNG, 34
rag::ImageLoaderPVR, 35
rag::InputManager, 36
   isLocked, 36
rag::ITextFont, 37
rag::Keyboard
   Keyboard, 38
   Keyboard, 39
rag::KeyboardManager, 41
rag::Material, 42
rag::MovieClip, 44
   MovieClip, 45
   play, 45
rag::MovieClip::Frame, 29
rag::Program, 47
rag::Rectangle, 48
rag::Renderer, 49
   bindMaterial, 50
rag::RenderTarget, 51
rag::Resource, 52
rag::ResourceMgr, 54
   loadResource, 54
rag::Screen, 56
rag::Shader, 57
rag::TextEdit, 60
rag::TextField, 62
rag::TextInput, 64
rag::Timer, 65
   getDelta, 65
rag::TTFFont, 67
rag::TUniformVar, 68
rag::Vertex, 69
rag::XFLBinaryParser, 70
rag::XFLParser, 71
removeChild
  rag::DisplayObject, 20
setCompressedFolder
  rag::Image, 31
soundName
rag::DisplayObject, 20
spine::SkeletonDrawable, 58
  updateBounds, 58
updateBounds
  rag::DisplayObject, 20
spine::SkeletonDrawable, 58
updateMatrix
  rag::DisplayObject, 20
uv
  rag::Bitmap, 10