SDL Classes Nachschlagewerk

Erzeugt von Doxygen 1.3.9.1

Sun Apr 3 20:22:31 2005

Inhaltsverzeichnis

Kapitel 1

Documentation einiger Klassen

Diese Klassen sollen irgendwann soweit erweitert werden, dass aus ihnen ein kleines RPG entsteht.

1.1 Das erste Kapitel

Das ist das erste Kapitel, und darin geht es zm foobar asdas asd as asd dsad as dasd

1.2 Das zweite Kapitel

Das ist das zweite Kapitel und darin geht es um foo

Autor:
Bodo Akdeniz

Kapitel 2

SDL Classes Verzeichnishierarchie

2.1 SDL Classes Verzeichnisse

Diese Verzeichnishierarchie ist -mit Einschränkungen- alphabetisch sortiert:

Kapitel 3

SDL Classes Hierarchie-Verzeichnis

3.1 SDL Classes Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

Kapitel 4

SDL Classes Klassen-Verzeichnis

4.1 SDL Classes Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

Kapitel 5

SDL Classes Datei-Verzeichnis

5.1 SDL Classes Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

Kapitel 6

SDL Classes Seitenindex

6.1 SDL Classes Zusätzliche Informationen

Hier folgt eine Liste mit zusammengehörigen Themengebieten:

Kapitel 7

SDL Classes Verzeichnisdokumentation

7.1 versuch2/ Verzeichnisreferenz

Dateien

- Datei animation.cc

  In dieser Datei werden die Funktionen der Klasse CAnimation implementiert.

- Datei animation.h

  In dieser Datei wird die Klasse CAnimation definiert.

- Datei animationimage.cc

  In dieser Datei werden die Funktionen der Klasse CAnimationImage implementiert.

- Datei animationimage.h

  In dieser Datei wird die Klasse CAnimationImage definiert.

- Datei documentation_mainsite.h
- Datei example_canimation.cc

  In dieser Datei ist ein kleines Beispiel zur Benutzung von CAnimation.

- Datei example_clist_1.cc

  Diese Datei enthält ein Beispielprogramm für CList.

- Datei image.cc
- Datei image.h

  Diese Datei definiert die Klasse CImage, welche dazu dient, Bilder zu laden und zu zeichnen.

- Datei list.cc

  siehe list.h

- Datei list.h

  In dieser Datei wird die Klasse CList definiert **und** *implementiert.*

- Datei surfacechild.cc

- Datei surfacechild.h

  In dieser Datei wird die Klasse CSurfaceChild definiert.

Kapitel 8

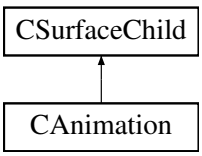
SDL Classes Klassen-Dokumentation

8.1 CAnimation Klassenreferenz

CAnimation ist eine Klasse um Animationen zu laden und zu zeichnen.

`#include <animation.h>`

Klassendiagramm für CAnimation::



Öffentliche Methoden

- CAnimation (SDL_Surface *dest)
  Constructor.
- virtual ~CAnimation ()
  Destructor.
- virtual void LoadFromFile (std::string file)
  Läd eine Animation.
- virtual void SetCurrentFrame (int i)
  Gibt den Frame an der gezeichnet werden soll.
- virtual void SetAnimationspeed (float speed)
  Setzt animationspeed.
- virtual int CalculateCurrentFrame ()
  Berechnet den aktuellen Frame.
- virtual void SetCurrentFrame ()

Berechnet den aktuellen Frame und setzt currentframe.

- virtual int CalculatePlaytime ()
  Ermittelt die gesamte Spielzeit der Animation in Milisekunden.
- virtual void SetPlaytime (int n)
  Setzt die Spielzeit.
- virtual void SetPlaytime ()
  Berechnet und setzt playtime.
- virtual int CalculateFullPlaytime ()
  Berechnet die volle Spielzeit der Animation.
- virtual void StartAnimation ()
  Startet die Animation.
- virtual void StopAnimation ()
  Stoppt die Animation.
- virtual bool IsRunning ()
  Gibt den Wert von running zurück.
- virtual void SetDefaultFrame (int n)
  Setzt die Nummer des Frames die gezeichnet werden soll, wenn die Animation steht.
- virtual int GetDefaultFrame ()
  Gibt defaultframe *zurück.*
- virtual void Draw ()
  Zeichnet den aktuellen Frame an die durch x *und* y *definierte Position.*
- virtual void Draw (int xpos, int ypos)
  Zeichnet den aktuellen Frame an durch xpos *und* ypos *definierte Position.*
- virtual CAnimationImage * GetCurrentImage ()
  Gibt das Bild des aktuellen Frames zurück.
- virtual float GetAnimationspeed ()
  Gibt den Wert von animationspeed *zurück.*
- virtual int GetCurrentFrame ()
  Gibt currentframe *zurück* **ohne** *es vorher neu zu berrechnen.*

Öffentliche Attribute

- int currentframe
  Die Nummer des gerade Frames, der als nächstes gezeichnet werden soll.

- CList< CAnimationImage * > ImageList

  Die Liste der Bilder die für die Animation benötigt wird.

- int playtime

  Die Zeit. die die Animation schon läuft in Milisekunden.

- int fullplaytime

  Die Zeit, die die Animation braucht um einmal komplett durchzulaufen in Milisekunden.

- long starttime

  Der Zeitpunkt an dem die Animation gestartet wurde als UNIX Timestamp.

- bool running

  Definiert ob die Animation läuft (true) oder nicht (false).

- float animationspeed

  Ein Wert für die Geschwindigkeit der Animation.

- int defaultframe

  Der Frame der gezeigt werden soll, wenn die Animation nicht läuft. Standardmäßig 0.

8.1.1 Ausführliche Beschreibung

CAnimation ist eine Klasse um Animationen zu laden und zu zeichnen.

Sie nutzt CAnimationImage für die Frames. Mehr Informationen zur Benutzung der Klasse finden sie auf der Seite Benutzung von CAnimation.

Noch zu erledigen

Noch nicht nutzbar!!!
Es muss noch was dazu dass man einstellen kann ob sich die Animation wiederholen soll oder nicht. Außerdem muss das dann noch irgendwo bearbeitet werden, DASS sie sich auch wiederholt, bzw. anhält, wenn sie fertig ist.
SCHEISSE!!!!!! Wir brauchen eine Klasse, die eine Position, und ein SDL_Surface besitzt!!!! davon können wir auch die ableiten. Die einzelnen Bilder der Animation bekommen dann eben als Parent, das Surface der Animation. D.h. ggf. CSurfaceChild erweitern und hier einiges ändern!!!
man könnte die Bilderliste auch noch in eine andere Klasse auslagern, von der wir diese dann ableiten.

Autor:
Bodo Akdeniz

Datum:
01.04.05

Version:
0.2

Definiert in Zeile 51 der Datei animation.h.

8.1.2 Beschreibung der Konstruktoren und Destruktoren

8.1.2.1 CAnimation::CAnimation (SDL_Surface * *dest*)

Constructor.

siehe CSurfaceChild::CSurfaceChild()

Definiert in Zeile 13 der Datei animation.cc.

Benutzt animationspeed, currentframe, defaultframe, fullplaytime und running.

8.1.2.2 CAnimation::~CAnimation () [virtual]

Destructor.

Definiert in Zeile 25 der Datei animation.cc.

8.1.3 Dokumentation der Elementfunktionen

8.1.3.1 int CAnimation::CalculateCurrentFrame () [virtual]

Berechnet den aktuellen Frame.

Die Funktion berechnet anhand der Zeitdaten der Frames und der vergangenen Laufzeit der Animation den aktuellen Frame.

Rückgabe:
Gibt die Nummer des ermittelten aktuellen Frames zurück. Dieser kann dann mit image[ermittelte_-nummer] verwendet werden.

Definiert in Zeile 46 der Datei animation.cc.

Benutzt CList< T >::Count(), GetAnimationspeed(), GetDefaultFrame(), ImageList, IsRunning() und SetPlaytime().

Wird benutzt von SetCurrentFrame().

8.1.3.2 int CAnimation::CalculateFullPlaytime () [virtual]

Berechnet die volle Spielzeit der Animation.

Die Funktion zählt die *DelayTime* aller Elemente von *ImageList* zusammen und gibt die errechnete Zeit zurück.

Rückgabe:
Es wird die volle Spielzeit der Animation zurückgegeben.

Definiert in Zeile 87 der Datei animation.cc.

Benutzt CList< T >::Count() und ImageList.

Wird benutzt von StartAnimation().

8.1.3.3 int CAnimation::CalculatePlaytime () [virtual]

Ermittelt die gesamte Spielzeit der Animation in Milisekunden.

Ermittelt die Spielzeit anhand der aktuellen Zeit und *starttime*

Definiert in Zeile 69 der Datei animation.cc.

Wird benutzt von SetPlaytime().

8.1.3.4 void CAnimation::Draw (int *xpos*, int *ypos*) [virtual]

Zeichnet den aktuellen Frame an durch *xpos* und *ypos* definierte Position.

Die Funktion macht das selbe wie Draw(), sie führt aber zuvor noch CSurfaceChild::SetPosition(xpos, ypos) aus.

Siehe auch:
Draw()

Definiert in Zeile 141 der Datei animation.cc.

Benutzt Draw() und CSurfaceChild::SetPosition().

8.1.3.5 void CAnimation::Draw () [virtual]

Zeichnet den aktuellen Frame an die durch *x* und *y* definierte Position.

Siehe auch:
Draw(int, int)

Definiert in Zeile 125 der Datei animation.cc.

Benutzt GetCurrentFrame(), CSurfaceChild::GetDestinationSurface(), CSurfaceChild::GetX(), CSurfaceChild::GetY(), ImageList und SetCurrentFrame().

Wird benutzt von Draw() und main().

8.1.3.6 float CAnimation::GetAnimationspeed () [virtual]

Gibt den Wert von *animationspeed* zurück.

Rückgabe:
Die Funktion gibt den Wert von animationspeed zurück.

Siehe auch:
animationspeed

Definiert in Zeile 153 der Datei animation.cc.

Wird benutzt von CalculateCurrentFrame().

8.1.3.7 int CAnimation::GetCurrentFrame () [virtual]

Gibt *currentframe* zurück **ohne** es vorher neu zu berrechnen.

Rückgabe:
Die Funktion gibt den Wert von *currentframe* zurück, **ohne** ihn vorher neu zu berechnen.

Definiert in Zeile 158 der Datei animation.cc.

Wird benutzt von Draw() und GetCurrentImage().

8.1.3.8 CAnimationImage * CAnimation::GetCurrentImage () [virtual]

Gibt das Bild des aktuellen Frames zurück.

Rückgabe:
Die Funktion gibt einen Zeiger auf *image*[currentframe] zurück.

Definiert in Zeile 147 der Datei animation.cc.

Benutzt GetCurrentFrame(), ImageList und SetCurrentFrame().

8.1.3.9 int CAnimation::GetDefaultFrame () [virtual]

Gibt *defaultframe* zurück.

Rückgabe:
Es wird der Wert von *defaultframe* zurückgegeben.

Siehe auch:
SetDefaultFrame()

Definiert in Zeile 120 der Datei animation.cc.

Wird benutzt von CalculateCurrentFrame().

8.1.3.10 bool CAnimation::IsRunning () [virtual]

Gibt den Wert von running zurück.

Rückgabe:
true, wenn die Animation gerade läuft, false wenn sie nicht läuft

Definiert in Zeile 110 der Datei animation.cc.

Wird benutzt von CalculateCurrentFrame() und main().

8.1.3.11 void CAnimation::LoadFromFile (std::string *file*) [virtual]

Läd eine Animation.

Noch zu erledigen
Muss noch implementiert werden. Ich weiß noch nicht genau wie das funktionieren soll.

Definiert in Zeile 29 der Datei animation.cc.

8.1.3.12 void CAnimation::SetAnimationspeed (float *speed*) [inline, virtual]

Setzt *animationspeed*.

Parameter:
speed ist der Faktor mit dem die *delaytime* des aktuellen Frame multipliziert wird.

Siehe auch:
animationspeed

Warnung:
Sollte nicht wärend einer laufenden Animation aufgerufen werden, da diese sonst "durcheinander" kommt.

Definiert in Zeile 41 der Datei animation.cc.

Benutzt animationspeed.

Wird benutzt von main().

8.1.3.13 void CAnimation::SetCurrentFrame () [virtual]

Berechnet den aktuellen Frame und setzt *currentframe*.

Berechnet die Nummer des aktuellen Frames mit CalculateCurrentFrame() und setzt *currentframe* entsprechend. Will man *currentframe* manuell bestimmen muss man SetCurrentFrame(int) benutzen.

Siehe auch:
SetCurrentFrame(int)

Definiert in Zeile 64 der Datei animation.cc.

Benutzt CalculateCurrentFrame() und currentframe.

Wird benutzt von Draw() und GetCurrentImage().

8.1.3.14 void CAnimation::SetCurrentFrame (int *i*) [inline, virtual]

Gibt den Frame an der gezeichnet werden soll.

Soll *currentframe* automatisch berechnet werden nutzen sie bitte SetCurrentFrame()

Parameter:
i ist dabei die Nummer, die der Frame in *images*[] hat.

Siehe auch:
SetCurrentFrame()

Warnung:
Diese Funktion sollte nicht benutzt werden, sie wird automatisch von SetCurrentFrame() aufgerufen und kann, sollte sie ohne die entsprechende Vorbereitung aufgerufen werden, zu Laufzeitfehlern führen.

Definiert in Zeile 36 der Datei animation.cc.

Benutzt currentframe.

8.1.3.15 void CAnimation::SetDefaultFrame (int *n*) [inline, virtual]

Setzt die Nummer des Frames die gezeichnet werden soll, wenn die Animation steht.

Parameter:
n ist der Wert auf den defaultframe gesetzt wird.

Siehe auch:
GetDefaultFrame()

Definiert in Zeile 115 der Datei animation.cc.

Benutzt defaultframe.

8.1.3.16 void CAnimation::SetPlaytime () [virtual]

Berechnet und setzt *playtime*.

Berechnet die Spielzeit mit CalculatePlaytime() und setzt *playtime* entsprechend.

Siehe auch:
SetPlaytime(int)

Achtung:
Diese Funktion wird automatisch aufgerufen. Es ist (außer vielleicht in seltenen Ausnahmefällen) nicht nötig sie manuell aufzurufen.

Definiert in Zeile 82 der Datei animation.cc.

Benutzt CalculatePlaytime().

Wird benutzt von CalculateCurrentFrame().

8.1.3.17 void CAnimation::SetPlaytime (int *n*) [inline, virtual]

Setzt die Spielzeit.

Setzt die Spielzeit auf *n* Milisekunden

Parameter:
n ist die Anzahl der Milisekunden, auf die *playtime* gesetzt werden soll.

Siehe auch:
SetPlaytime()

Warnung:
Diese Funktion sollte nicht benutzt werden, sie wird automatisch von SetPlaytime() aufgerufen und kann, sollte sie ohne die entsprechende Vorbereitung aufgerufen werden, zu Laufzeitfehlern führen.

Definiert in Zeile 77 der Datei animation.cc.

Benutzt playtime.

8.1.3.18 void CAnimation::StartAnimation () [virtual]

Startet die Animation.

Die Funktion bereitet alles für das laufen der Animation vor.

Siehe auch:
StopAnimation()

Definiert in Zeile 98 der Datei animation.cc.

Benutzt CalculateFullPlaytime(), fullplaytime, running und starttime.

Wird benutzt von main().

8.1.3.19 void CAnimation::StopAnimation () [inline, virtual]

Stoppt die Animation.

Siehe auch:
StartAnimation()

Definiert in Zeile 105 der Datei animation.cc.

Benutzt running.

Wird benutzt von main().

8.1.4 Dokumentation der Datenelemente

8.1.4.1 float CAnimation::animationspeed

Ein Wert für die Geschwindigkeit der Animation.

Dieser Wert wird mit image[i]->delaytime multipliziert, um die Geschwindigkeit der Animation verändern zu können. D.h. ist animationspeed 1, läuft die Animation in ihrem Originaltempo, ist es 2 mit halbem Tempo, 0.5 entspräche folglich dem doppelten Tempo.

Siehe auch:
SetAnimationspeed()

Definiert in Zeile 79 der Datei animation.h.

Wird benutzt von CAnimation() und SetAnimationspeed().

8.1.4.2 int CAnimation::currentframe

Die Nummer des gerade Frames, der als nächstes gezeichnet werden soll.

Definiert in Zeile 55 der Datei animation.h.

Wird benutzt von CAnimation() und SetCurrentFrame().

8.1.4.3 int CAnimation::defaultframe

Der Frame der gezeigt werden soll, wenn die Animation nicht läuft. Standardmäßig 0.

Definiert in Zeile 82 der Datei animation.h.

Wird benutzt von CAnimation() und SetDefaultFrame().

8.1.4.4 int CAnimation::fullplaytime

Die Zeit, die die Animation braucht um einmal komplett durchzulaufen in Milisekunden.

Definiert in Zeile 64 der Datei animation.h.

Wird benutzt von CAnimation() und StartAnimation().

8.1.4.5 CList<CAnimationImage*> CAnimation::ImageList

Die Liste der Bilder die für die Animation benötigt wird.

Definiert in Zeile 58 der Datei animation.h.

Wird benutzt von CalculateCurrentFrame(), CalculateFullPlaytime(), Draw(), GetCurrentImage() und main().

8.1.4.6 int CAnimation::playtime

Die Zeit. die die Animation schon läuft in Milisekunden.

Definiert in Zeile 61 der Datei animation.h.

Wird benutzt von SetPlaytime().

8.1.4.7 bool CAnimation::running

Definiert ob die Animation läuft (true) oder nicht (false).

Definiert in Zeile 70 der Datei animation.h.

Wird benutzt von CAnimation(), StartAnimation() und StopAnimation().

8.1.4.8 long CAnimation::starttime

Der Zeitpunkt an dem die Animation gestartet wurde als UNIX Timestamp.

Definiert in Zeile 67 der Datei animation.h.

Wird benutzt von StartAnimation().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

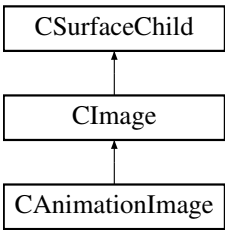
- versuch2/animation.h
- versuch2/animation.cc

8.2 CAnimationImage Klassenreferenz

Die Klasse dient als Frame von CAnimation.

`#include <animationimage.h>`

Klassendiagramm für CAnimationImage::



Öffentliche Methoden

- CAnimationImage (SDL_Surface *dest)

  Constructor.

- virtual ~CAnimationImage ()

  Destructor.

- virtual void SetDelayTime (int delaytime)

  Setzt DelayTime *auf* delaytime.

- virtual int GetDelayTime ()

  Gibt DelayTime *zurück.*

- virtual void SetTimestamp (int n)

  Setzt timestamp *auf* n.

- virtual int GetTimestamp ()

  Gibt timestamp *zurück.*

Öffentliche Attribute

- int DelayTime

  DelayTime *legt eine Zeit in Milisekunden fest, die gewartet werden soll, bis der nächste Frame von CAnimation angezeigt wird.*

- int timestamp

  Die Anzahl Milisekunden, die vom Beginn der Animation (CAnimation) an verstreichen, bis dieser Frame gezeichnet wird.

8.2.1 Ausführliche Beschreibung

Die Klasse dient als Frame von CAnimation.

Sie erweitert CImage um *DelayTime*, um sie ihre Eignung für Animationen zu bessern.

Autor:
Bodo Akdeniz

Datum:
01.04.05

Version:
0.2

Definiert in Zeile 26 der Datei animationimage.h.

8.2.2 Beschreibung der Konstruktoren und Destruktoren

8.2.2.1 CAnimationImage::CAnimationImage (SDL_Surface * *dest*)

Constructor.

Siehe CImage::CImage()

Definiert in Zeile 10 der Datei animationimage.cc.

8.2.2.2 CAnimationImage::~CAnimationImage () [virtual]

Destructor.

Definiert in Zeile 14 der Datei animationimage.cc.

8.2.3 Dokumentation der Elementfunktionen

8.2.3.1 int CAnimationImage::GetDelayTime () [virtual]

Gibt *DelayTime* zurück.

Rückgabe:
DelayTime.

Definiert in Zeile 23 der Datei animationimage.cc.

8.2.3.2 int CAnimationImage::GetTimestamp () [virtual]

Gibt *timestamp* zurück.

Rückgabe:
Die Funktion gibt den Wert von *timestamp* zurück.

Siehe auch:
SetTimestamp()

Definiert in Zeile 33 der Datei animationimage.cc.

8.2.3.3 void CAnimationImage::SetDelayTime (int *delaytime*) [virtual]

Setzt *DelayTime* auf *delaytime*.

Parameter:
delaytime ist die Zeit (in Milisekunden) auf die *DelayTime* gesetzt werden soll.

Definiert in Zeile 18 der Datei animationimage.cc.

Benutzt DelayTime.

Wird benutzt von main().

8.2.3.4 void CAnimationImage::SetTimestamp (int *n*) [virtual]

Setzt *timestamp* auf *n*.

Die Funktion setzt *timestamp* auf *n*. Diese Funktion muss nie von Hand aufgerufen werden. Sie wird von CAnimation benötigt und aufgerufen.

Parameter:
n ist der Wert auf den *timestamp* gesetzt wird.

Siehe auch:
GetTimestamp()

Definiert in Zeile 28 der Datei animationimage.cc.

Benutzt timestamp.

8.2.4 Dokumentation der Datenelemente

8.2.4.1 int CAnimationImage::DelayTime

DelayTime legt eine Zeit in Milisekunden fest, die gewartet werden soll, bis der nächste Frame von CAnimation angezeigt wird.

Definiert in Zeile 30 der Datei animationimage.h.

Wird benutzt von SetDelayTime().

8.2.4.2 int CAnimationImage::timestamp

Die Anzahl Milisekunden, die vom Beginn der Animation (CAnimation) an verstreichen, bis dieser Frame gezeichnet wird.

Definiert in Zeile 33 der Datei animationimage.h.

Wird benutzt von SetTimestamp().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

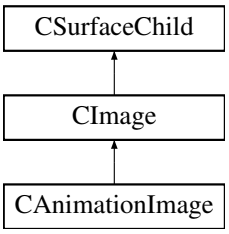
- versuch2/animationimage.h
- versuch2/animationimage.cc

8.3 CImage Klassenreferenz

Klasse zum laden/zeichnen von Bildern.

`#include <image.h>`

Klassendiagramm für CImage::



Öffentliche Methoden

- CImage (SDL_Surface *dest)
  Constructor.
- virtual ~CImage ()
  Destructor.
- virtual void LoadFromFile (std::string file)
  Läd ein Bild.
- virtual void SetColorKey (Uint8 r, Uint8 g, Uint8 b)
  Setzt die Farbe die transparent dargestellt werden soll.
- virtual void SetAlpha (Uint8 a)
  Setzt den Grad der Transparenz.
- virtual void SetImage (SDL_Surface *img)
  Setzt image.
- virtual void SetSourceRect (SDL_Rect *src)
  Gibt ein Rechteck von image *an, das gezeichnet werden soll.*
- virtual void SetSourceRect (int rx, int ry, int rw, int rh)
  Gibt ein Rechteck von image *an, das gezeichnet werden soll.*
- virtual void Draw ()
  Zeichnet image.
- virtual void Draw (int xpos, int ypos)
  Zeichnet image *an die (*xpos/*ypos*).
- virtual SDL_Surface * GetSurface ()
  Gibt image *zurück.*

Öffentliche Attribute

- SDL_Surface * image

  Zeigt auf das SDL_Surface des Bildes.

- SDL_Rect * sourcerect

  Definiert eine Rechteckige Fläche von image, *die gezeichnet werden soll. Ist* sourcerect *NULL wird, bei einem Aufruf von Draw(), image komplett gezeichnet.*

- std::string filename

  Der Dateiname des geladenen Bildes. Kann auch leer sein, wenn image *durch SetImage() gesetzt wurde.*

8.3.1 Ausführliche Beschreibung

Klasse zum laden/zeichnen von Bildern.

Eine Klasse um ein Bild (Bitmap) zu laden, speichern und zu zeichnen. Transparenz wird sowohl durch einen "Colorkey" als auch durch einen Alphawert unterstützt.

Autor:
Bodo Akdeniz

Datum:
01.04.05

Version:
0.2

Definiert in Zeile 29 der Datei image.h.

8.3.2 Beschreibung der Konstruktoren und Destruktoren

8.3.2.1 CImage::CImage (SDL_Surface * *dest*)

Constructor.

Wird beim initiieren der Klasse aufgerufen und setzt das Surface auf dem gezeichnet werden soll.

Parameter:
dest ist ein Zeiger auf das SDL_Surface, auf dem bei einem Aufruf von Draw(), gezeichnet werden soll.

Siehe auch:
SetDestinationSurface()

Definiert in Zeile 10 der Datei image.cc.

Benutzt SetSourceRect().

8.3.2.2 CImage::~CImage () [virtual]

Destructor.

Definiert in Zeile 15 der Datei image.cc.

8.3.3 Dokumentation der Elementfunktionen

8.3.3.1 void CImage::Draw (int *xpos*, int *ypos*) [virtual]

Zeichnet *image* an die (*xpos*/ypos*)*.

Genauso wie Draw(), nur dass davor SetPosition(xpos, ypos) aufgerufen wird.

Siehe auch:
Draw()
SetPositino()

Definiert in Zeile 66 der Datei image.cc.

Benutzt Draw() und CSurfaceChild::SetPosition().

8.3.3.2 void CImage::Draw () [virtual]

Zeichnet *image*.

Die Funktion zeichnet den, mit SetSourceRect() angegebenen Teil, von *image* an die mit SetPosition() festgelegte Position in das SDL_Surface, das beim initialisieren mit CImage() oder manuell mit SetDestinationSurface(), angegeben wurde.

Siehe auch:
SetSourceRect()
SetPosition
CImage()
SetDestinationSurface()
Draw(int, int)

Definiert in Zeile 58 der Datei image.cc.

Benutzt image und sourcerect.

Wird benutzt von Draw().

8.3.3.3 SDL_Surface * CImage::GetSurface () [virtual]

Gibt *image* zurück.

Gibt einen Zeiger auf das SDL_Surface zurück, das das zu zeichnende Bild enthält.

Rückgabe:
Zeiger auf das SDL_Surface, das das Bild enthält (*image*).

Definiert in Zeile 72 der Datei image.cc.

8.3.3.4 void CImage::LoadFromFile (std::string *file*) [virtual]

Läd ein Bild.

LoadFromFile() läd ein Bitmap in image, das dann verwendet werden kann. Tritt ein Fehler auf ist *image* NULL und *filename* leer.

Siehe auch:
SetImage()

Definiert in Zeile 19 der Datei image.cc.

Benutzt filename und image.

Wird benutzt von main().

8.3.3.5 void CImage::SetAlpha (Uint8 *a*) [virtual]

Setzt den Grad der Transparenz.

Parameter:
a bestimmt den Grad der Transparenz. (0=keine Transparenz bis 255=unsichtbar)

Siehe auch:
SetColorKey()

Definiert in Zeile 33 der Datei image.cc.

Benutzt image.

8.3.3.6 void CImage::SetColorKey (Uint8 *r*, Uint8 *g*, Uint8 *b*) [virtual]

Setzt die Farbe die transparent dargestellt werden soll.

Setzt die Farbe, die beim zeichnen mit Draw() nicht gezeichnet werden soll, also transparent dargestellt wird.

Parameter:
r bestimmt den Rotanteil der Farbe. (0=kein Rot bis 255=Sehr Rot)
g bestimmt den Grünanteil der Farbe. (0=kein Grün bis 255=Sehr Grün)
b bestimmt den Blauanteil der Farbe. (0=kein Blau bis 255=Sehr Blau)

Siehe auch:
SetAlpha()

Definiert in Zeile 28 der Datei image.cc.

Benutzt image.

8.3.3.7 void CImage::SetImage (SDL_Surface * *img*) [virtual]

Setzt *image*.

Die Funktion kann verwendet werden, wenn direkt ein SDL_Surface übergeben werden und kein Bild aus einer Datei geladen werden soll. Um ein Bild aus einer Datei zu laden verwenden sie bitte LoadFromFile().

Parameter:
img ist ein Zeiger auf das SDL_Surface, das als *image* verwendet werden soll.

Siehe auch:
LoadFromFile()

Definiert in Zeile 38 der Datei image.cc.

Benutzt image.

8.3.3.8 void CImage::SetSourceRect (int *rx*, int *ry*, int *rw*, int *rh*) [virtual]

Gibt ein Rechteck von *image* an, das gezeichnet werden soll.

Die Funktion hat die selbe Funktionalität wie SetSourceRect(SDL_Rect *), nur kann man hier die Koordinaten, Länge und Breite direkt übergeben.

Parameter:

rx ist die x-Koordinate des Rechtecks innerhalb von *image*.

ry ist die y-Koordinate des Rechtecks innerhalb von *image*.

rw ist die Breite des Rechtecks.

rh ist die Höhe des Rechtecks.

Siehe auch:

SetSourceRect(SDL_Rect *)

Definiert in Zeile 48 der Datei image.cc.

Benutzt SetSourceRect().

8.3.3.9 void CImage::SetSourceRect (SDL_Rect * *src*) [virtual]

Gibt ein Rechteck von *image* an, das gezeichnet werden soll.

Diese Funktion kann verwendet werden, wenn nur ein Teil, von *image* gezeichnet werden soll. Soll *image* komplett gezeichnet werden, kann für *src* NULL übergeben werden, das ist aber sofern es nicht manuell geändert wurde, sowiso so.

Parameter:

src ist ein Zeiger auf ein SDL_Rect, und bestimmt ein Rechteck von *image*.

Siehe auch:

SetSourceRect(int, int, int, int)

Definiert in Zeile 43 der Datei image.cc.

Benutzt sourcerect.

Wird benutzt von CImage() und SetSourceRect().

8.3.4 Dokumentation der Datenelemente

8.3.4.1 std::string CImage::filename

Der Dateiname des geladenen Bildes. Kann auch leer sein, wenn *image* durch SetImage() gesetzt wurde.

Definiert in Zeile 39 der Datei image.h.

Wird benutzt von LoadFromFile().

8.3.4.2 SDL_Surface* CImage::image

Zeigt auf das SDL_Surface des Bildes.

Definiert in Zeile 35 der Datei image.h.

Wird benutzt von Draw(), LoadFromFile(), SetAlpha(), SetColorKey() und SetImage().

8.3.4.3 SDL_Rect* CImage::sourcerect

Definiert eine Rechteckige Fläche von *image*, die gezeichnet werden soll. Ist *sourcerect* NULL wird, bei einem Aufruf von Draw(), image komplett gezeichnet.

Definiert in Zeile 37 der Datei image.h.

Wird benutzt von Draw() und SetSourceRect().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- versuch2/image.h
- versuch2/image.cc

8.4 CList< T > Template Klassenreferenz

`#include <list.h>`

Öffentliche Methoden

- void Add (T obj)

  Fügt ein Objekt am Ende der Liste hinzu.

- void Clear ()

  Löscht die gesamte Liste.

- int Count ()

  Gibt die Anzahl der Elemente zurück, die in der Liste gespeichert sind.

8.4.1 Ausführliche Beschreibung

template<class T> class CList< T >

Die Klasse ist von std::vector abgeleitet und bietet dieselbe Funktionalität. Sie wird nur definiert, um einen besser zu den anderen Klassen passenderen Klassennamen und passendere Funktionsnamen zu haben.

Noch zu erledigen
Fertigmachen!!

Autor:
Bodo Akdeniz

Datum:
02.04.05

Version:
0.1

Definiert in Zeile 32 der Datei list.h.

8.4.2 Dokumentation der Elementfunktionen

8.4.2.1 template<class T> void CList< T >::Add (T *obj*) [inline]

Fügt ein Objekt am Ende der Liste hinzu.

Parameter:
obj Ein Objekt vom Typ T

Definiert in Zeile 39 der Datei list.h.

Wird benutzt von main().

8.4.2.2 template<class T> void CList< T >::Clear () [inline]

Löscht die gesamte Liste.

Definiert in Zeile 46 der Datei list.h.

8.4.2.3 template<class T> int CList< T >::Count () [inline]

Gibt die Anzahl der Elemente zurück, die in der Liste gespeichert sind.

Rückgabe:
die Funktion gibt die Anzahl der Listenelemente zurück.

Definiert in Zeile 54 der Datei list.h.

Wird benutzt von CAnimation::CalculateCurrentFrame(), CAnimation::CalculateFullPlaytime() und main().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

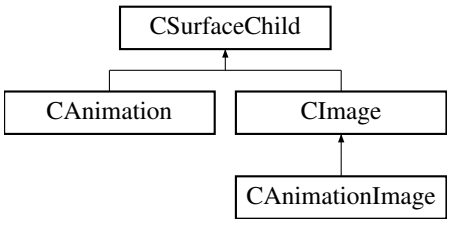
- versuch2/list.h

8.5 CSurfaceChild Klassenreferenz

CSurfaceChild ist ein, in einem SDL_Surface, positionnierbares Object.

`#include <surfacechild.h>`

Klassendiagramm für CSurfaceChild::



Öffentliche Methoden

- CSurfaceChild (SDL_Surface ∗surface)
  Constuctor.
- virtual ∼CSurfaceChild ()
  Destructor.
- virtual void SetDestinationSurface (SDL_Surface ∗dest)
  Setzt destination *auf* dest.
- virtual void SetPosition (int newx, int newy)
  Gibt die Position an, an der das Bild gespeichert werden soll.
- virtual int GetX ()
  Gibt x *zurück.*
- virtual int GetY ()
  Gibt y *zurück.*
- virtual SDL_Surface ∗ GetDestinationSurface ()
  Gibt destination *zurück.*

Öffentliche Attribute

- SDL_Surface ∗ destination
  Das SDL_Surface in dem die Ausgabe stattfinden bzw. in dem gezeichnet wird.
- int x
  Die x-Position des Objekts relativ zu destination.
- int y
  Die y-Position des Objekts relativ zu destination.

8.5.1 Ausführliche Beschreibung

CSurfaceChild ist ein, in einem SDL_Surface, positionnierbares Object.

Von dieser Klasse werden alle anderen Objekte abgeleitet, die in ein SDL_Surface (auch der Bildschirm ist in SDL ein SDL_Surface) gezeichnet werden und eine bestimmte Position haben. Das dürfte auf alle Objekte zutreffen, die irgendwann auf dem Bildschirm zu sehen sind.

Autor:
Bodo Akdeniz

Datum:
01.04.05

Version:
0.1

Definiert in Zeile 24 der Datei surfacechild.h.

8.5.2 Beschreibung der Konstruktoren und Destruktoren

8.5.2.1 CSurfaceChild::CSurfaceChild (SDL_Surface * *surface*)

Constuctor.

Initiert das Objekt und setzt *destination* auf *surface*.

Parameter:
surface ist ein Zeiger auf das SDL_Surface auf dem dann gezeichnet werden soll.

Definiert in Zeile 10 der Datei surfacechild.cc.

Benutzt SetDestinationSurface().

8.5.2.2 CSurfaceChild::~CSurfaceChild () [virtual]

Destructor.

Definiert in Zeile 15 der Datei surfacechild.cc.

8.5.3 Dokumentation der Elementfunktionen

8.5.3.1 SDL_Surface * CSurfaceChild::GetDestinationSurface () [virtual]

Gibt *destination* zurück.

Rückgabe:
Gibt einen Zeiger auf das SDL_Surface zurück auf dem gezeichnet werden soll.

Definiert in Zeile 40 der Datei surfacechild.cc.

Wird benutzt von CAnimation::Draw().

8.5.3.2 int CSurfaceChild::GetX () [virtual]

Gibt *x* zurück.

Rückgabe:
Gibt die x-Position des Objekts zurück.

Siehe auch:
GetY()

Definiert in Zeile 30 der Datei surfacechild.cc.

Wird benutzt von CAnimation::Draw().

8.5.3.3 int CSurfaceChild::GetY () [virtual]

Gibt *y* zurück.

Rückgabe:
Gibt die y-Position des Objekts zurück.

Siehe auch:
GetX()

Definiert in Zeile 35 der Datei surfacechild.cc.

Wird benutzt von CAnimation::Draw().

8.5.3.4 void CSurfaceChild::SetDestinationSurface (SDL_Surface * *dest*) [virtual]

Setzt *destination* auf *dest*.

Diese Funktion wird automatisch vom Constructor aufgerufen, und muss so, nur manuell aufgerufen werden, wenn *destination* nachträglich geändert werden soll.

Siehe auch:
CSurfaceChild()

Definiert in Zeile 19 der Datei surfacechild.cc.

Benutzt destination.

Wird benutzt von CSurfaceChild().

8.5.3.5 void CSurfaceChild::SetPosition (int *newx*, int *newy*) [virtual]

Gibt die Position an, an der das Bild gespeichert werden soll.

Die angegebene Position ist relativ zu der Position von *destination*. D.h, wenn *destination* auf dem Bildschirm die Position (10/20) hat, und SetPosition(5, 5) aufgerufen wird, wird auf dem Bildschirm (bei einem Aufruf von Draw()) an der Position (15/25) gezeichnet.

Parameter:
newx ist die relative x-Position.

newy ist die relative y-Position.

Siehe auch:
SetDestinationSurface()

Definiert in Zeile 24 der Datei surfacechild.cc.

Benutzt x und y.

Wird benutzt von CImage::Draw(), CAnimation::Draw() und main().

8.5.4 Dokumentation der Datenelemente

8.5.4.1 SDL_Surface* CSurfaceChild::destination

Das SDL_Surface in dem die Ausgabe stattfinden bzw. in dem gezeichnet wird.

Definiert in Zeile 28 der Datei surfacechild.h.

Wird benutzt von SetDestinationSurface().

8.5.4.2 int CSurfaceChild::x

Die x-Position des Objekts relativ zu *destination*.

Definiert in Zeile 31 der Datei surfacechild.h.

Wird benutzt von SetPosition().

8.5.4.3 int CSurfaceChild::y

Die y-Position des Objekts relativ zu *destination*.

Definiert in Zeile 34 der Datei surfacechild.h.

Wird benutzt von SetPosition().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- versuch2/surfacechild.h
- versuch2/surfacechild.cc

8.6 vector Klassenreferenz

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- versuch2/list.h

Kapitel 9

SDL Classes Datei-Dokumentation

9.1 versuch2/animation.cc-Dateireferenz

In dieser Datei werden die Funktionen der Klasse CAnimation implementiert.

```
#include "animation.h"
#include <iostream>
```

9.1.1 Ausführliche Beschreibung

In dieser Datei werden die Funktionen der Klasse CAnimation implementiert.

Autor:
Bodo Akdeniz

Datum:
02.04.05

Definiert in Datei animation.cc.

9.2 versuch2/animation.h-Dateireferenz

In dieser Datei wird die Klasse CAnimation definiert.

```
#include <SDL/SDL.h>
#include <string>
#include "animationimage.h"
#include "surfacechild.h"
#include "list.h"
```

Klassen

- class CAnimation

  CAnimation ist eine Klasse um Animationen zu laden und zu zeichnen.

9.2.1 Ausführliche Beschreibung

In dieser Datei wird die Klasse CAnimation definiert.

CAnimation kann dazu genutzt werden, Animationen zu laden und auf zu zeichnen.

Autor:
Bodo Akdeniz

Definiert in Datei animation.h.

9.3 versuch2/animationimage.cc-Dateireferenz

In dieser Datei werden die Funktionen der Klasse CAnimationImage implementiert.

```
#include "animationimage.h"
```

9.3.1 Ausführliche Beschreibung

In dieser Datei werden die Funktionen der Klasse CAnimationImage implementiert.

Autor:
Bodo Akdeniz

Datum:
01.04.05

Definiert in Datei animationimage.cc.

9.4 versuch2/animationimage.h-Dateireferenz

In dieser Datei wird die Klasse CAnimationImage definiert.

```
#include <SDL/SDL.h>
#include "image.h"
```

Klassen

- class CAnimationImage

  Die Klasse dient als Frame von CAnimation.

9.4.1 Ausführliche Beschreibung

In dieser Datei wird die Klasse CAnimationImage definiert.

CAnimationImage erweitert CImage um die Variable *DelayTime*, um die Klasse besser für die Verwendung in einer Animation von CAnimation zu machen.

Autor:
Bodo Akdeniz

Definiert in Datei animationimage.h.

9.5 versuch2/documentation_mainsite.h-Dateireferenz

9.6 versuch2/example_canimation.cc-Dateireferenz

In dieser Datei ist ein kleines Beispiel zur Benutzung von CAnimation.

```
#include <SDL/SDL.h>
#include <iostream>
#include "animation.h"
#include "animationimage.h"
```

Funktionen

- int main ()

9.6.1 Ausführliche Beschreibung

In dieser Datei ist ein kleines Beispiel zur Benutzung von CAnimation.

Es werden 2 Bilder geladen und der Animation hinzugefügt, und die Spielschleife wird gestartet. jetzt kann durch drücken der Leertaste die Animation gestartet und gestoppt werden. Beendet wird das Programm mit [ESC]. Es wird davon ausgegangen, dass der grundsätzliche Umgang mit SDL bekonnt ist.

Autor:
Bodo Akdeniz

Datum:
02.04.05

Version:
0.2

Definiert in Datei example_canimation.cc.

9.6.2 Dokumentation der Funktionen

9.6.2.1 int main ()

Definiert in Zeile 24 der Datei example_canimation.cc.

Benutzt CList< T >::Add(), CAnimation::Draw(), CAnimation::ImageList, CAnimation::IsRunning(), CImage::LoadFromFile(), CAnimation::SetAnimationspeed(), CAnimationImage::SetDelayTime(), CSurfaceChild::SetPosition(), CAnimation::StartAnimation() und CAnimation::StopAnimation().

9.7 versuch2/example_clist_1.cc-Dateireferenz

Diese Datei enthält ein Beispielprogramm für CList.

```
#include <string>
#include <iostream>
#include "list.h"
```

Funktionen

- int main ()

9.7.1 Ausführliche Beschreibung

Diese Datei enthält ein Beispielprogramm für CList.

Es werden Eingaben entgegen genommen, die in der Liste gespeichert werden. Bei der Eingabe eines Punktes wird die Liste ausgegeben und das Programm beendet.

Autor:
Bodo Akdeniz

Datum:
03.04.05

Version:
0.1

Definiert in Datei example_clist_1.cc.

9.7.2 Dokumentation der Funktionen

9.7.2.1 int main ()

Definiert in Zeile 18 der Datei example_clist_1.cc.

Benutzt CList< T >::Add() und CList< T >::Count().

9.8 versuch2/image.cc-Dateireferenz

```
#include "image.h"
```

9.9 versuch2/image.h-Dateireferenz

Diese Datei definiert die Klasse CImage, welche dazu dient, Bilder zu laden und zu zeichnen.

```
#include <SDL/SDL.h>
#include <string>
#include "surfacechild.h"
```

Klassen

- class CImage

  Klasse zum laden/zeichnen von Bildern.

9.9.1 Ausführliche Beschreibung

Diese Datei definiert die Klasse CImage, welche dazu dient, Bilder zu laden und zu zeichnen.

Um sie zu nutzen benötigt man SDL.

Autor:
Bodo Akdeniz

Definiert in Datei image.h.

9.10 versuch2/list.cc-Dateireferenz

siehe list.h

```
#include "list.h"
```

9.10.1 Ausführliche Beschreibung

siehe list.h

Dort befinden sich auch alle Implementationen.

Autor:
Bodo Akdeniz

Definiert in Datei list.cc.

9.11 versuch2/list.h-Dateireferenz

In dieser Datei wird die Klasse CList definiert **und** implementiert.

```
#include <vector>
```

Klassen

- class CList< T >

9.11.1 Ausführliche Beschreibung

In dieser Datei wird die Klasse CList definiert **und** implementiert.

Aus irgendwelchen mir bis jetzt nicht verständlichen Gründen, können Templateklassen wohl nicht (wie üblich) in zwei Dateien aufgeteilt werden. Aus diesem Grund werden die Funktionen der Klasse CList in dieser Datei nicht nur deklariert, sondern auch implementiert. Die Datei list.cc ist nur vorhanden, um die Struktur einheitlich zu gestalten, in ihr wird nur diese Datei includiert, dass man wie gewohnt kompilieren kann.

Autor:
Bodo Akdeniz

Definiert in Datei list.h.

9.12 versuch2/surfacechild.cc-Dateireferenz

```
#include "surfacechild.h"
```

9.13 versuch2/surfacechild.h-Dateireferenz

In dieser Datei wird die Klasse CSurfaceChild definiert.

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

Klassen

- class CSurfaceChild

  CSurfaceChild ist ein, in einem SDL_Surface, positionnierbares Object.

9.13.1 Ausführliche Beschreibung

In dieser Datei wird die Klasse CSurfaceChild definiert.

Autor:
Bodo Akdeniz

Definiert in Datei surfacechild.h.

Kapitel 10

SDL Classes Zusätzliche Informationen

10.1 Benutzung von CAnimation

bla bla bla

Noch zu erledigen

Sowas muss ich auch für CImage machen!!!!!!

10.2 Liste der zu erledigenden Dinge

Klasse CAnimation Noch nicht nutzbar!!!

Es muss noch was dazu dass man einstellen kann ob sich die Animation wiederholen soll oder nicht. Außerdem muss das dann noch irgendwo bearbeitet werden, DASS sie sich auch wiederholt, bzw. anhält, wenn sie fertig ist.

SCHEISSE!!!!!! Wir brauchen eine Klasse, die eine Position, und ein SDL_Surface besitzt!!!! davon können wir auch die ableiten. Die einzelnen Bilder der Animation bekommen dann eben als Parent, das Surface der Animation. D.h. ggf. CSurfaceChild erweitern und hier einiges ändern!!!

man könnte die Bilderliste auch noch in eine andere Klasse auslagern, von der wir diese dann ableiten.

Element CAnimation::CAnimation(SDL_Surface *dest) Es hat sich einiges geändert und die Funktion MUSS aktualisiert werden.

Element CAnimation::CalculateCurrentFrame() MUSS noch getestet und nochmal überdacht werden.

Element CAnimation::LoadFromFile(std::string file) Muss noch implementiert werden. Ich weiß noch nicht genau wie das funktionieren soll.

Element CAnimation::LoadFromFile(std::string file) muss noch implementiert werden.

Klasse CList< T > Fertigmachen!!

Seite Benutzung von CAnimation Sowas muss ich auch für CImage machen!!!!!!

Index