VQUARK

Generated by Doxygen 1.9.7

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 VQuark::ConnectBasic< ParameterType > Class Template Reference

Public Member Functions

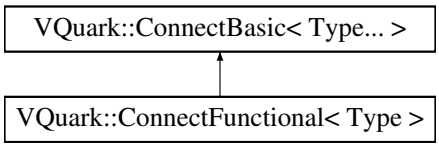
- **ConnectBasic** (std::function< void(ParameterType...)> Function)
- std::function< void(ParameterType...)> * **GetFunction** ()
- const bool **IsBlock** () const
- void **SetBlock** (const bool &Status)

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

- inc/VQuarkSignal.h

4.2 VQuark::ConnectFunctional< Type > Class Template Reference

Inheritance diagram for VQuark::ConnectFunctional< Type >:



Public Types

- using **FunctionPointer** = void(*)(Type...)

Public Member Functions

- **ConnectFunctional** (FunctionPointer Init)
- FunctionPointer **GetPointer** ()

Public Member Functions inherited from VQuark::ConnectBasic< Type... >

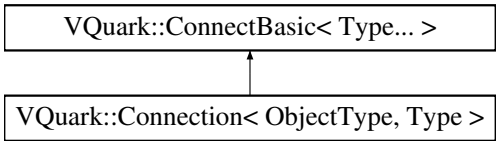
- **ConnectBasic** (std::function< void(ParameterType...)> Function)
- std::function< void(ParameterType...)> * **GetFunction** ()
- const bool **IsBlock** () const
- void **SetBlock** (const bool &Status)

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

- inc/VQuarkSignal.h

4.3 VQuark::Connection< ObjectType, Type > Class Template Reference

Inheritance diagram for VQuark::Connection< ObjectType, Type >:



Public Types

- using **ObjectPointer** = void(ObjectType::*)(Type...)

Public Member Functions

- **Connection** (ObjectType *ObjectPointer, ObjectPointer Function)
- void * **GetRawObject** ()
- ObjectPointer **GetRawFunction** ()

Public Member Functions inherited from VQuark::ConnectBasic< Type... >

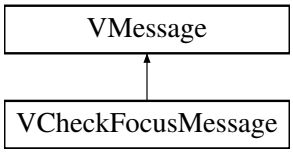
- **ConnectBasic** (std::function< void(ParameterType...)> Function)
- std::function< void(ParameterType...)> * **GetFunction** ()
- const bool **IsBlock** () const
- void **SetBlock** (const bool &Status)

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

- inc/VQuarkSignal.h

4.4 VCheckFocusMessage Class Reference

Inheritance diagram for VCheckFocusMessage:



Public Member Functions

- **VCheckFocusMessage** (HWND TriggerWidget, const VPoint &Point, void ∗MessageObject, const bool &MouseClick=false)

Public Member Functions inherited from VMessage

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Public Attributes

- VPoint **FocusPoint**
- void ∗ **Object**
- bool **Click**

Public Attributes inherited from VMessage

- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

- inc/VQuarkMessage.h

4.5 VQuark::VCorePoint< DataType > Class Template Reference

`#include <VQuarkData.h>`

Public Types

- using **Rect** = VCoreRect< DataType >
- using **ConstRect** = const Rect &
- using **Integer** = const DataType &
- using **Point** = const VCorePoint &

Public Member Functions

- const bool InsideRectangle (ConstRect Judgement) const

  : Judge the point inside the rectangle or not
- void Move (Integer X, Integer Y)

  : Move the point to specified position
- void Offset (Integer XOF, Integer YOF)

  : Offset the point
- template<class TupleType , class RawType = DataType>
  TupleType ToTwoTuple ()

  : Convert the point into two-tuple
- **VCorePoint** (Integer _IX, Integer _IY)
- **VCorePoint** (Point Point)

Public Attributes

- DataType **X**
- DataType **Y**

Friends

- bool **operator==** (ConstRect Left, ConstRect Right)
- bool **operator!=** (ConstRect Left, ConstRect Right)

4.5.1 Detailed Description

template<class DataType>
class VQuark::VCorePoint< DataType >

Template Parameters

DataType	

4.5.2 Member Function Documentation

4.5.2.1 InsideRectangle()

```
template<class DataType >
const bool VQuark::VCorePoint< DataType >::InsideRectangle (
            ConstRect Judgement ) const  [inline]
```

: Judge the point inside the rectangle or not

Parameters

Judgement	: Rectangle for judging

Returns

: If the point inside the rectangle, return true, nor return false

4.5.2.2 Move()

```
template<class DataType >
void VQuark::VCorePoint< DataType >::Move (
             Integer X,
             Integer Y )  [inline]
```

: Move the point to specified position

Parameters

X	: The new X
Y	: The new Y

4.5.2.3 Offset()

```
template<class DataType >
void VQuark::VCorePoint< DataType >::Offset (
             Integer XOF,
             Integer YOF )  [inline]
```

: Offset the point

Parameters

XOF	: The offset value of x
YOF	: The offset value of y

4.5.2.4 ToTwoTuple()

```
template<class DataType >
template<class TupleType , class RawType = DataType>
TupleType VQuark::VCorePoint< DataType >::ToTwoTuple ( )  [inline]
```

: Convert the point into two-tuple

Template Parameters

TupleType	: The two-tuple type
RawType	: The data type of two-tuple type

Returns

: The two-tuple

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

- inc/VQuarkData.h

4.6 VQuark::VCoreRect< DataType > Class Template Reference

: The abstract of the rectangle

`#include <VQuarkData.h>`

Public Types

- using **Integer** = const DataType &
  : The alias of ¨DataType¨
- using **Rect** = const VCoreRect &
  : The alias of ¨VCoreRect¨

Public Member Functions

- const DataType GetWidth () const
  : Get the width of the rectangle
- const DataType GetHeight () const
  : Get the height of the rectangle
- void Move (Integer X, Integer Y)
  : Move the rectangle to the target position
- VCoreRect * MoveChaining (Integer X, Integer Y)
  : Move the rectangle to the target position (Chaining method)
- void Extended (Integer LeftPanding, Integer TopPanding, Integer RightPanding, Integer BottomPanding)
  : Extended the rectangle
- void Resize (Integer Width, Integer Height)
  : Resize the rectangle
- void FusionRect (Rect Rectangle)
  : Mix the targeted rectangle with the rectangle
- bool Overlap (Rect JudgeRectangle)
  : Judge a target rectangle overlap with this rectangle
- bool Include (Rect Judgement)
  : Judge a target rectangle include with this rectangle
- template<class TupleType , class RawType = DataType>
  TupleType ToQuadruple ()
  : Convert this rectangle into a quadruple
- **VCoreRect** (Integer _ILeft, Integer _IRight, Integer _ITop, Integer _IBottom)
- **VCoreRect** (Rect Rectangle)

Public Attributes

- DataType **Left**
- DataType **Right**
- DataType **Top**
- DataType **Bottom**

4.6.1 Detailed Description

template<class DataType>
class VQuark::VCoreRect< DataType >

: The abstract of the rectangle

Template Parameters

DataType	: The data typew will using

4.6.2 Member Function Documentation

4.6.2.1 Extended()

```
template<class DataType >
void VQuark::VCoreRect< DataType >::Extended (
             Integer LeftPanding,
             Integer TopPanding,
             Integer RightPanding,
             Integer BottomPanding )  [inline]
```

: Extended the rectangle

Parameters

LeftPanding	: The left panding
TopPanding	: The top panding
RightPanding	: The right panding
BottomPanding	: The bottom panding

4.6.2.2 FusionRect()

```
template<class DataType >
void VQuark::VCoreRect< DataType >::FusionRect (
             Rect Rectangle )  [inline]
```

: Mix the targeted rectangle with the rectangle

Parameters

Rectangle	: The rectangle value

4.6.2.3 GetHeight()

```
template<class DataType >
const DataType VQuark::VCoreRect< DataType >::GetHeight ( ) const  [inline]
```

: Get the height of the rectangle

Returns

: The height value

4.6.2.4 GetWidth()

```
template<class DataType >
const DataType VQuark::VCoreRect< DataType >::GetWidth ( ) const  [inline]
```

: Get the width of the rectangle

Returns

: The width value

4.6.2.5 Include()

```
template<class DataType >
bool VQuark::VCoreRect< DataType >::Include (
            Rect Judgement )  [inline]
```

: Judge a target rectangle include with this rectangle

Parameters

Judgement	: The Judgement

Returns

: If it include with this rectangle, return true, nor false.

4.6.2.6 Move()

```
template<class DataType >
void VQuark::VCoreRect< DataType >::Move (
            Integer X,
            Integer Y )  [inline]
```

: Move the rectangle to the target position

Parameters

X	: The X
Y	: The Y

4.6.2.7 MoveChaining()

```
template<class DataType >
VCoreRect * VQuark::VCoreRect< DataType >::MoveChaining (
            Integer X,
            Integer Y )  [inline]
```

: Move the rectangle to the target position (Chaining method)

Parameters

X	: The X
Y	: The Y

Returns

: The self-pointer (For chaining method)

4.6.2.8 Overlap()

```
template<class DataType >
bool VQuark::VCoreRect< DataType >::Overlap (
            Rect JudgeRectangle )  [inline]
```

: Judge a target rectangle overlap with this rectangle

Parameters

JudgeRectangle	: The Judgement

Returns

: If it overalps with this rectangle, return true, nor false.

4.6.2.9 Resize()

```
template<class DataType >
void VQuark::VCoreRect< DataType >::Resize (
            Integer Width,
            Integer Height )  [inline]
```

: Resize the rectangle

Parameters

Width	: The width
Height	: The height

4.6.2.10 ToQuadruple()

```
template<class DataType >
template<class TupleType , class RawType = DataType>
TupleType VQuark::VCoreRect< DataType >::ToQuadruple ( )  [inline]
```

: Convert this rectangle into a quadruple

Template Parameters

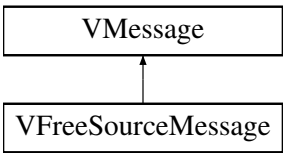
TupleType	: The quadruple

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

- inc/VQuarkData.h

4.7 VFreeSourceMessage Class Reference

Inheritance diagram for VFreeSourceMessage:



Public Member Functions

- **VFreeSourceMessage** (HWND TriggerWidget)

Public Member Functions inherited from VMessage

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Additional Inherited Members

Public Attributes inherited from VMessage

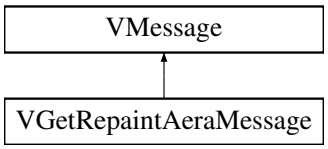
- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

- inc/VQuarkMessage.h

4.8 VGetRepaintAeraMessage Class Reference

Inheritance diagram for VGetRepaintAeraMessage:



Public Member Functions

- **VGetRepaintAeraMessage** (HWND TriggerWidget, VRect &RepaintRegion)

Public Member Functions inherited from VMessage

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Public Attributes

- VRect ∗ **RepaintAera**

Public Attributes inherited from VMessage

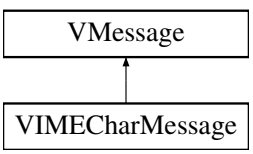
- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

- inc/VQuarkMessage.h

4.9 VIMECharMessage Class Reference

Inheritance diagram for VIMECharMessage:



Public Member Functions

- **VIMECharMessage** (HWND TriggerWidget, wchar_t CharInputed)

Public Member Functions inherited from VMessage

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Public Attributes

- wchar_t **IMEChar**

Public Attributes inherited from VMessage

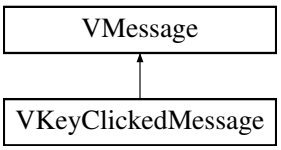
- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

- inc/VQuarkMessage.h

4.10 VKeyClickedMessage Class Reference

Inheritance diagram for VKeyClickedMessage:



Public Member Functions

- **VKeyClickedMessage** (HWND TriggerWidget, byte VKCode, bool PrevDown, bool Extened, VkeyClicked↩ Flag Stats)

Public Member Functions inherited from VMessage

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Public Attributes

- byte **KeyVKCode**
- bool **KeyPrevDown**
- bool **KeyExtened**
- VkeyClickedFlag **KeyStats**

Public Attributes inherited from VMessage

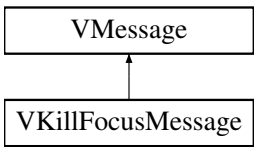
- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

- inc/VQuarkMessage.h

4.11 VKillFocusMessage Class Reference

Inheritance diagram for VKillFocusMessage:



Public Member Functions

- **VKillFocusMessage** (HWND TriggerWidget)

Public Member Functions inherited from VMessage

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Additional Inherited Members

Public Attributes inherited from VMessage

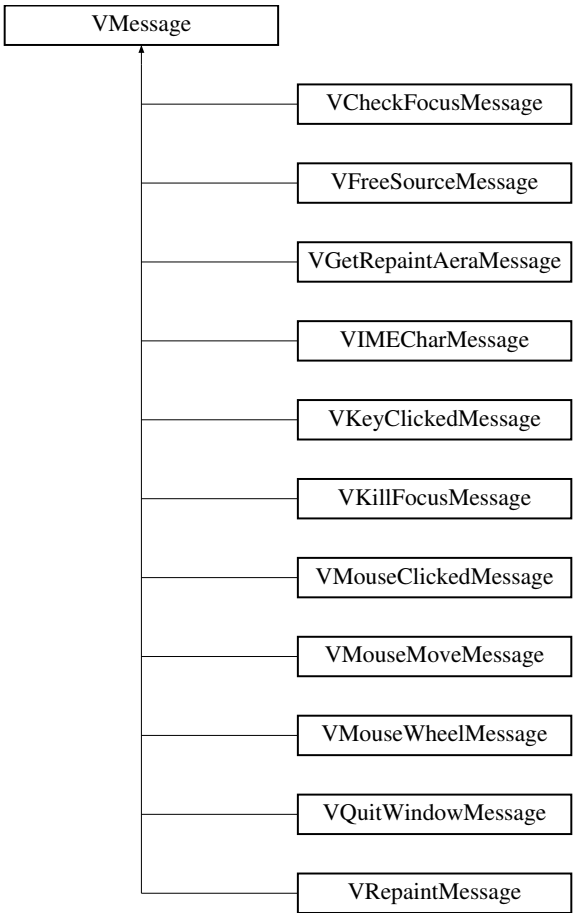
- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

- inc/VQuarkMessage.h

4.12 VMessage Class Reference

Inheritance diagram for VMessage:



Public Member Functions

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Public Attributes

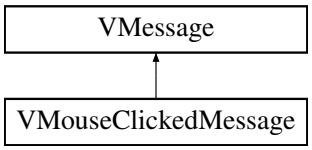
- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

- inc/VQuarkMessage.h

4.13 VMouseClickedMessage Class Reference

Inheritance diagram for VMouseClickedMessage:



Public Member Functions

- **VMouseClickedMessage** (HWND TriggerWidget, int X, int Y, VMouseClickedFlag ClickedFlag, VMouse↩ KeyFlag Key)

Public Member Functions inherited from VMessage

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Public Attributes

- VPoint **MousePosition**
- VMouseClickedFlag **ClickedMethod**
- VMouseKeyFlag **ClickedKey**

Public Attributes inherited from VMessage

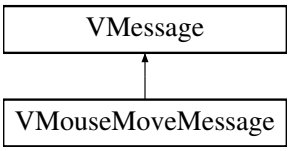
- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

- inc/VQuarkMessage.h

4.14 VMouseMoveMessage Class Reference

Inheritance diagram for VMouseMoveMessage:



Public Member Functions

- **VMouseMoveMessage** (HWND TriggerWidget, int X, int Y)

Public Member Functions inherited from VMessage

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Public Attributes

- VPoint **MousePosition**

Public Attributes inherited from VMessage

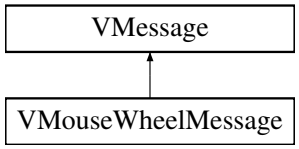
- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

- inc/VQuarkMessage.h

4.15 VMouseWheelMessage Class Reference

Inheritance diagram for VMouseWheelMessage:



Public Member Functions

- **VMouseWheelMessage** (HWND TriggerWidget, int X, int Y, short WheelParameter)

Public Member Functions inherited from VMessage

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Public Attributes

- VPoint **MousePosition**
- short **WheelValue**

Public Attributes inherited from VMessage

- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

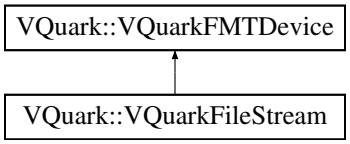
- inc/VQuarkMessage.h

4.16 VQuark::VQuarkFileStream Class Reference

: The file output stream

`#include <VQuarkFmt.h>`

Inheritance diagram for VQuark::VQuarkFileStream:



Public Member Functions

- **VQuarkFileStream** (const VString &Path)
- void Output (const VString &String) override
  : Print string
- void Open (const VString &Path)
  : Open a file
- void **Close** ()
  : Close the stream

- virtual void Output (const VString &String)=0
  : Print string

4.16.1 Detailed Description

: The file output stream

4.16.2 Member Function Documentation

4.16.2.1 Open()

```
void VQuark::VQuarkFileStream::Open (
            const VString & Path )
```

: Open a file

Parameters

Path	: The file path

4.16.2.2 Output()

```
void VQuark::VQuarkFileStream::Output (
            const VString & String )  [override], [virtual]
```

: Print string

Parameters

String	: The string

Implements VQuark::VQuarkFMTDevice.

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

- inc/VQuarkFmt.h
- src/VQuarkFmt.cpp

4.17 VQuark::VQuarkFMT Class Reference

Static Public Member Functions

- static void **Print** (const VString &String)
- template<class Input , class... Types>
  static void **Print** (const VString &String, Input _Input, Types... Args)
- template<class... Types>
  static void Print (const VString &String, Types... Args)

  : The format print (on std stream)
- static void **PrintDevice** (VQuarkFMTStream *Stream, const VString &String)
- template<class Input , class... Types>
  static void **PrintDevice** (VQuarkFMTStream *Stream, const VString &String, Input _Input, Types... Args)
- template<class... Types>
  static void PrintDevice (VQuarkFMTStream *Stream, const VString &String, Types... Args)

  : The format print (on specifed device)
- static VString **Format** (const VString &String)
- template<class Input , class... Types>
  static VString **Format** (const VString &String, Input _Input, Types... Args)
- template<class... Types>
  static VString Format (const VString &String, Types... Args)

  : Format string

4.17.1 Member Function Documentation

4.17.1.1 Format()

```
template<class...  Types>
static VString VQuark::VQuarkFMT::Format (
             const VString & String,
             Types...  Args )  [inline], [static]
```

: Format string

Template Parameters

...Types	: The agrument type

Parameters

String	: The format string
...Args	: The agrument

Returns

: The formated string

4.17.1.2 Print()

```
template<class...  Types>
static void VQuark::VQuarkFMT::Print (
             const VString & String,
             Types...  Args )  [inline], [static]
```

: The format print (on std stream)

Template Parameters

...Types	: The agrument type

Parameters

String	: The string
...Args	: The agrument

4.17.1.3 PrintDevice()

```
template<class...  Types>
static void VQuark::VQuarkFMT::PrintDevice (
             VQuarkFMTStream * Stream,
```

```
            const VString & String,
            Types... Args ) [inline], [static]
```

: The format print (on specifed device)

Template Parameters

...Types	: The agrument type

Parameters

String	: The string
...Args	: The agrument

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

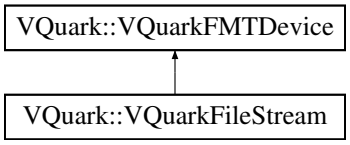
- inc/VQuarkFmt.h

4.18 VQuark::VQuarkFMTDevice Class Reference

: The FMT output device base class

`#include <VQuarkFmt.h>`

Inheritance diagram for VQuark::VQuarkFMTDevice:



Public Member Functions

- virtual void Output (const VString &String)=0
  : Print string

4.18.1 Detailed Description

: The FMT output device base class

4.18.2 Member Function Documentation

4.18.2.1 Output()

```
virtual void VQuark::VQuarkFMTDevice::Output (
            const VString & String ) [pure virtual]
```

: Print string

Parameters

String	: The string

Implemented in VQuark::VQuarkFileStream.

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

- inc/VQuarkFmt.h
- src/VQuarkFmt.cpp

4.19 VQuarkScreenDevice Class Reference

Static Public Member Functions

- static const unsigned short GetWidth ()
  : Get the screen's width
- static const unsigned short GetHeight ()
  : Get the screen's height
- static const unsigned short MurseClientWidth ()
  : Get the screen's width (Without the bar)
- static const unsigned short MurseClientHeight ()
  : Get the screen's height (Without the bar)

4.19.1 Member Function Documentation

4.19.1.1 GetHeight()

```
const unsigned short VQuarkScreenDevice::GetHeight ( )  [static]
```

: Get the screen's height

Returns

: The height

4.19.1.2 GetWidth()

```
VQUARK_SPACE_BEGIN const unsigned short VQuarkScreenDevice::GetWidth ( )  [static]
```

: Get the screen's width

Returns

: The width

4.19.1.3 MurseClientHeight()

```
const unsigned short VQuarkScreenDevice::MurseClientHeight ( )  [static]
```

: Get the screen's height (Without the bar)

Returns

: The height

4.19.1.4 MurseClientWidth()

```
const unsigned short VQuarkScreenDevice::MurseClientWidth ( )  [static]
```

: Get the screen's width (Without the bar)

Returns

: The width

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

- inc/VQuarkSys.h
- src/VQuarkSys.cpp

4.20 VQuarkWidget Class Reference

Public Member Functions

- VQuarkWidget (VWindowHandle WidgetHandle)
  : Build the Widget from the window handle
- void SetTitle (const VString &Title)
  : Set the title of the window
- void Resize (const size_t Width, const size_t Height)
  : Set the size of the window
- void Move (const size_t &X, const size_t &Y)
  : Move the widget into specified position
- void **Show** ()
  : Display the window
- void **Hide** ()
  : Hide the window
- void **MoveCenter** ()
  : Move the window into center place
- const size_t GetWidth () const noexcept
  : Get the window's width
- const size_t GetHeight () const noexcept
  : Get the window's height
- const VWindowHandle GetHandle () const noexcept
  : Get the handle of the window

Static Public Member Functions

- static VQuarkWidget * CreateWidget (const VString &Class, const VString &Title, const size_t &Width, const size_t &Height, const VWindowHandle &BelongTo=NULL)

  : Create a window

4.20.1 Constructor & Destructor Documentation

4.20.1.1 VQuarkWidget()

```
VQUARK_SPACE_BEGIN VQuarkWidget::VQuarkWidget (
            VWindowHandle WidgetHandle )
```

: Build the Widget from the window handle

Parameters

WidgetHandle	: The window handle

4.20.2 Member Function Documentation

4.20.2.1 CreateWidget()

```
static VQuarkWidget * VQuarkWidget::CreateWidget (
            const VString & Class,
            const VString & Title,
            const size_t & Width,
            const size_t & Height,
            const VWindowHandle & BelongTo = NULL )  [inline], [static]
```

: Create a window

Parameters

Class	: The window's titile
Title	: The class name of the window
Width	: The width of the window
Height	: The height of the window
BelongTo	: The parent of the window

Returns

: If a window was created successfully, return the VQuark widget instance

4.20.2.2 GetHandle()

```
const VWindowHandle VQuarkWidget::GetHandle ( ) const  [noexcept]
```

: Get the handle of the window

Returns

: The handle of the window

4.20.2.3 GetHeight()

```
const size_t VQuarkWidget::GetHeight ( ) const  [inline], [noexcept]
```

: Get the window's height

Returns

: The height

4.20.2.4 GetWidth()

```
const size_t VQuarkWidget::GetWidth ( ) const  [inline], [noexcept]
```

: Get the window's width

Returns

: The width

4.20.2.5 Move()

```
void VQuarkWidget::Move (
            const size_t & X,
            const size_t & Y )
```

: Move the widget into specified position

Parameters

X	: The x data
Y	: The y data

4.20.2.6 Resize()

```
void VQuarkWidget::Resize (
            const size_t Width,
            const size_t Height )
```

: Set the size of the window

Parameters

Width	: The width
Height	: The height

4.20.2.7 SetTitle()

```
void VQuarkWidget::SetTitle (
            const VString & Title )
```

: Set the title of the window

Parameters

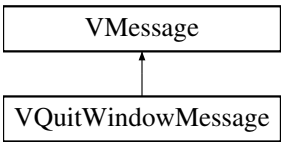
Title	: The title

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

- inc/VQuarkWidget.h
- src/VQuarkWidget.cpp

4.21 VQuitWindowMessage Class Reference

Inheritance diagram for VQuitWindowMessage:



Public Member Functions

- **VQuitWindowMessage** (HWND TriggerWidget)

Public Member Functions inherited from VMessage

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Additional Inherited Members

Public Attributes inherited from VMessage

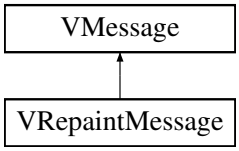
- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

- inc/VQuarkMessage.h

4.22 VRepaintMessage Class Reference

Inheritance diagram for VRepaintMessage:



Public Member Functions

- **VRepaintMessage** (HWND TriggerWidget, const VRect &RepaintRegion)

Public Member Functions inherited from VMessage

- **VMessage** (VMessageType Type=UnknowMessage)
- VMessageType **GetType** ()

Public Attributes

- VRect **DirtyRectangle**

Public Attributes inherited from VMessage

- UINT **Win32ID**
- HWND **MessageTriggerWidget** = NULL
- WPARAM **wParameter**
- LPARAM **lParameter**

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

- inc/VQuarkMessage.h

4.23 VQuark::VSignal< Type > Class Template Reference

: The signal in VQuark

`#include <VQuarkSignal.h>`

Public Member Functions

- void **Connect** (void(∗Function)(Type...))
- template<class ObjectType >
  void **Connect** (ObjectType ∗Object, void(ObjectType::∗Function)(Type...))
- template<class ObjectType >
  void **Disconnect** (ObjectType ∗Object, void(ObjectType::∗Function)(Type...))
- void **Block** (void(∗Function)(Type...), bool BlockStatus)
- template<class ObjectType >
  void **Block** (ObjectType ∗Object, void(∗Function)(Type...), bool BlockStatus)
- void **Emit** (Type... Agruments)

4.23.1 Detailed Description

template<class... Type>
class VQuark::VSignal< Type >

: The signal in VQuark

Template Parameters

...Type	: The agrument type list

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

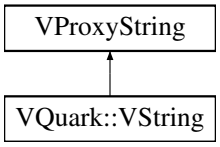
- inc/VQuarkSignal.h

4.24 VQuark::VString Class Reference

: The wrapper of the STL string

`#include <VQuarkString.h>`

Inheritance diagram for VQuark::VString:



Public Types

- using **Iterator** = VProxyString::iterator
  : The iterator
- using **ConstIterator** = VProxyString::const_iterator
  : The const iterator
- using **ReverseIterator** = VProxyString::reverse_iterator
  : The Reverse iterator

Public Member Functions

- **VString** (const std::wstring &String)
- **VString** (const std::string &String)
- **VString** (const wchar_t *String)
- **VString** (wchar_t *String)
- **VString** (const char *String)
- **VString** (char *String)
- VString Split (const size_t &Begin, const size_t &SplitCount)
  : Split the string

- VString Split (const size_t &Begin, const size_t &SplitCount) const
  : Split the string
- VString SplitRange (const size_t &Begin, const size_t &End)
  : Split the string
- void Append (const VString &AppendString)
  : Append string
- bool StartWith (const VString &JudgeString, const size_t &StartOn=0)
  : Is the string start with the specified string
- bool EndWith (const VString &JudgeString)
  : Is the string end with the specified string
- VChar & At (const size_t &Position)
  : Get the characeter from specified position
- bool IsEmpty ()
  : Is the string empty
- void Fill (const VChar &Character, size_t FillSize=0)
  : Fill the string
- void Set (const VString &String)
  : Set the string with specified string
- void Erase (const size_t &Begin, const size_t &Count)
  : Delete the characeters from the
- void Erase (ConstIterator Iterator)
  : Erase the character by iterator
- void EraseRange (const size_t &Begin, const size_t &End)
  : Erase by range
- void Insert (const size_t &Position, const VString &String)
  : Insert a specified string
- void Insert (const size_t &Position, const VString &String, const size_t &Count)
  : Insert a specified string by the specified times
- void Insert (ConstIterator Iterator, const VChar &Character)
  : Insert a specified character by the iterator
- size_t IndexOf (const VString &String, const size_t &StartAt=0) const
  : Find the specified string
- size_t IndexLastOf (const VString &String, const size_t &StartAt=0)
  : Find the one last of the specified string in this string
- Iterator Begin ()
  : Get the begin iterator
- Iterator End ()
  : Get the end iterator
- ReverseIterator ReverseBegin ()
  : Get the begin of the reverse iterator
- ReverseIterator ReverseEnd ()
  : Get the end of the reverse iterator
- VString Args (VString FormatInstance) const
  : Args format with string
- VString Args (int IntFormat) const
  : Args format with int
- VString Args (const long long IntFormat) const
  : Args format with long long
- VString Args (const long IntFormat) const
  : Args format with long
- VString Args (const unsigned int IntFormat) const

: Args format with unsigned int

- VString Args (const unsigned long IntFormat) const

  : Args format with unsgiend long
- VString Args (const unsigned long long IntFormat) const

  : Args format with unsigned long long
- VString Args (const VPoint Format) const

  : Args format with VPoint
- VString Args (const VRect Format) const

  : Args format with VRect
- const VChar * CStyleString () const

  : Get the C Style String
- size_t Size () const

  : Get the string size
- size_t Length () const

  : Get the length of the string

Static Public Member Functions

- static VString FromNumber (const int &NumberConvert)

  : Build from the number
- static VString FromNumber (const long long &NumberConvert)

  : Build from the number
- static VString FromNumber (const long &NumberConvert)

  : Build from the number
- static VString FromNumber (const unsigned int &NumberConvert)

  : Build from the number
- static VString FromNumber (const unsigned long &NumberConvert)

  : Build from the number
- static VString FromNumber (const unsigned long long &NumberConvert)

  : Build from the number
- static VString FromString (const std::string &String)

  : Build from the low byte string
- static VString FromWideString (const std::wstring &String)

  : Build from the wide byte string

Static Public Attributes

- static constexpr auto **NoPosition** { static_cast<size_type>(-1) }

  : If the IndexOf function dosen't find anything, return this

4.24.1 Detailed Description

: The wrapper of the STL string

4.24.2 Member Function Documentation

4.24.2.1 Append()

```
void VQuark::VString::Append (
            const VString & AppendString )
```

: Append string

Parameters

AppendString	: The string be appended

4.24.2.2 Args() [1/9]

```
VString VQuark::VString::Args (
            const long IntFormat ) const
```

: Args format with long

Parameters

IntFormat	: Format agrument

Returns

: Formated string

4.24.2.3 Args() [2/9]

```
VString VQuark::VString::Args (
            const long long IntFormat ) const
```

: Args format with long long

Parameters

IntFormat	: Format agrument

Returns

: Formated string

4.24.2.4 Args() [3/9]

```
VString VQuark::VString::Args (
            const unsigned int IntFormat ) const
```

: Args format with unsigned int

Parameters

IntFormat	: Format agrument

Returns

: Formated string

4.24.2.5 Args() [4/9]

```
VString VQuark::VString::Args (
            const unsigned long IntFormat ) const
```

: Args format with unsgiend long

Parameters

IntFormat	: Format agrument

Returns

: Formated string

4.24.2.6 Args() [5/9]

```
VString VQuark::VString::Args (
            const unsigned long long IntFormat ) const
```

: Args format with unsigned long long

Parameters

IntFormat	: Format agrument

Returns

: Formated string

4.24.2.7 Args() [6/9]

```
VString VQuark::VString::Args (
            const VPoint Format ) const
```

: Args format with VPoint

Parameters

IntFormat	: Format agrument

Returns

: Formated string

4.24.2.8 Args() [7/9]

```
VString VQuark::VString::Args (
            const VRect Format ) const
```

: Args format with VRect

Parameters

IntFormat	: Format agrument

Returns

: Formated string

4.24.2.9 Args() [8/9]

```
VString VQuark::VString::Args (
            int IntFormat ) const
```

: Args format with int

Parameters

IntFormat	: Format agrument

Returns

: Formated string

4.24.2.10 Args() [9/9]

```
VString VQuark::VString::Args (
            VString FormatInstance ) const
```

: Args format with string

Parameters

FormatInstance	: Format agrument

Returns

: Formated string

4.24.2.11 At()

```
VChar & VQuark::VString::At (
            const size_t & Position )  [inline]
```

: Get the characeter from specified position

Parameters

Position	: The position

Returns

: The characeter

4.24.2.12 Begin()

```
VString::Iterator VQuark::VString::Begin ( )
```

: Get the begin iterator

Returns

: The iterator

4.24.2.13 CStyleString()

```
const VChar * VQuark::VString::CStyleString ( ) const
```

: Get the C Style String

Returns

: The C Style String

4.24.2.14 End()

```
VString::Iterator VQuark::VString::End ( )
```

: Get the end iterator

Returns

: The iterator

4.24.2.15 EndWith()

```
bool VQuark::VString::EndWith (
            const VString & JudgeString )
```

: Is the string end with the specified string

Returns

: If this string end with the specified string, return true, nor return false

4.24.2.16 Erase() [1/2]

```
void VQuark::VString::Erase (
            const size_t & Begin,
            const size_t & Count )
```

: Delete the characeters from the

Parameters

Begin	: The begin position
Count	: The end position

4.24.2.17 Erase() [2/2]

```
void VQuark::VString::Erase (
            ConstIterator Iterator )
```

: Erase the character by iterator

Parameters

Iterator	: The iterator

4.24.2.18 EraseRange()

```
void VQuark::VString::EraseRange (
            const size_t & Begin,
            const size_t & End )
```

: Erase by range

Parameters

Begin	: Range begin
End	: Range end

4.24.2.19 Fill()

```
void VQuark::VString::Fill (
            const VChar & Character,
            size_t FillSize = 0 )
```

: Fill the string

Parameters

Character	: The specified characeter
FillSize	: The count of the characters

4.24.2.20 FromNumber() [1/6]

```
VString VQuark::VString::FromNumber (
            const int & NumberConvert )  [static]
```

: Build from the number

Parameters

NumberConvert	: The number

Returns

: A string that was converted from the number

4.24.2.21 FromNumber() [2/6]

```
VString VQuark::VString::FromNumber (
            const long & NumberConvert )  [static]
```

: Build from the number

Parameters

NumberConvert	: The number

Returns

: A string that was converted from the number

4.24.2.22 FromNumber() [3/6]

```
VString VQuark::VString::FromNumber (
            const long long & NumberConvert )  [static]
```

: Build from the number

Parameters

NumberConvert	: The number

Returns

: A string that was converted from the number

4.24.2.23 FromNumber() [4/6]

```
VString VQuark::VString::FromNumber (
            const unsigned int & NumberConvert )  [static]
```

: Build from the number

Parameters

NumberConvert	: The number

Returns

: A string that was converted from the number

4.24.2.24 FromNumber() [5/6]

```
VString VQuark::VString::FromNumber (
            const unsigned long & NumberConvert )  [static]
```

: Build from the number

Parameters

NumberConvert	: The number

Returns

: A string that was converted from the number

4.24.2.25 FromNumber() [6/6]

```
VString VQuark::VString::FromNumber (
            const unsigned long long & NumberConvert )  [static]
```

: Build from the number

Parameters

NumberConvert	: The number

Returns

: A string that was converted from the number

4.24.2.26 FromString()

```
VString VQuark::VString::FromString (
            const std::string & String )  [static]
```

: Build from the low byte string

Parameters

String	: The string

Returns

: A string that was converted from the low byte string

4.24.2.27 FromWideString()

```
VString VQuark::VString::FromWideString (
            const std::wstring & String )  [static]
```

: Build from the wide byte string

Parameters

String	: The string

Returns

: A string that was converted from the wide byte string

4.24.2.28 IndexLastOf()

```
size_t VQuark::VString::IndexLastOf (
            const VString & String,
            const size_t & StartAt = 0 )
```

: Find the one last of the specified string in this string

Parameters

String	: The string
StartAt	: Where to start

Returns

: If there exists the string, return the position of the string, nor return npos

4.24.2.29 IndexOf()

```
size_t VQuark::VString::IndexOf (
            const VString & String,
            const size_t & StartAt = 0 ) const
```

: Find the specified string

Parameters

String	: The string
StartAt	: Where to start

Returns

: If there exists the string, return the position of the string, nor return npos

4.24.2.30 Insert() [1/3]

```
void VQuark::VString::Insert (
            const size_t & Position,
            const VString & String )
```

: Insert a specified string

Parameters

Position	: The position
String	: The string

4.24.2.31 Insert() [2/3]

```
void VQuark::VString::Insert (
            const size_t & Position,
            const VString & String,
            const size_t & Count )
```

: Insert a specified string by the specified times

Parameters

Position	: The position
String	: The string
Count	: Count of the string

4.24.2.32 Insert() [3/3]

```
void VQuark::VString::Insert (
            ConstIterator Iterator,
            const VChar & Character )
```

: Insert a specified character by the iterator

Parameters

Iterator	: The iterator
Character	: The character

4.24.2.33 IsEmpty()

```
bool VQuark::VString::IsEmpty ( )
```

: Is the string empty

Returns

: If it is empty, return true

4.24.2.34 Length()

```
size_t VQuark::VString::Length ( ) const
```

: Get the length of the string

Returns

: The string's length

4.24.2.35 ReverseBegin()

```
VString::ReverseIterator VQuark::VString::ReverseBegin ( )
```

: Get the begin of the reverse iterator

Returns

: The iterator

4.24.2.36 ReverseEnd()

```
VString::ReverseIterator VQuark::VString::ReverseEnd ( )
```

: Get the end of the reverse iterator

Returns

: The iterator

4.24.2.37 Set()

```
void VQuark::VString::Set (
            const VString & String )
```

: Set the string with specified string

Parameters

String	: The string

4.24.2.38 Size()

```
size_t VQuark::VString::Size ( ) const
```

: Get the string size

Returns

: The size of the string

4.24.2.39 Split() [1/2]

```
VString VQuark::VString::Split (
            const size_t & Begin,
            const size_t & SplitCount )
```

: Split the string

Parameters

Begin	: The begin of the split string
SplitCount	: The characters count of the split string

Returns

: The string was splited

4.24.2.40 Split() [2/2]

```
VString VQuark::VString::Split (
            const size_t & Begin,
            const size_t & SplitCount ) const
```

: Split the string

Parameters

Begin	: The begin of the split string
SplitCount	: The characters count of the split string

Returns

: The string was splited

4.24.2.41 SplitRange()

```
VString VQuark::VString::SplitRange (
            const size_t & Begin,
            const size_t & End )
```

: Split the string

Parameters

Begin	: The begin of the split string
SplitCount	: The end of the split string

Returns

: The string was splited

4.24.2.42 StartWith()

```
bool VQuark::VString::StartWith (
            const VString & JudgeString,
            const size_t & StartOn = 0 )
```

: Is the string start with the specified string

Parameters

JudgeString	: The string will be judged
StartOn	: Where to start

Returns

: If this string start with the specified string, return true, nor return false

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

- inc/VQuarkString.h
- src/VQuarkString.cpp

Chapter 5

File Documentation

5.1 VQuarkBase.h

```
00001 /*
00002  * VQuarkBase.h.h (2023/5.20)
00003  *   The basic defition of the VQuark (Window handle, Basic type, e.t.c)
00004  *
00005  *
00006  * Copyright (C) 2023~now Margoo
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining a copy of this softwareand
       associated documentation files(the "Software"),
00009  * to deal in the Software without restriction, including without limitation the rights to use, copy,
       modify, merge, publish, distribute, sublicense, and /or sell copies of the Software,
00010  * and to permit persons to whom the Software is furnished to do so, subject to the following
       conditions :
00011  *
00012  * The above copyright noticeand this permission notice shall be included in all copies or substantial
       portions of the Software.
00013  *
00014  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
       NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00015  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
       HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00016  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
       DEALINGS IN THE SOFTWARE.
00017 */
00018 #pragma once
00019
00020 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) || defined(__NT__)
00021
00022 /*
00023  * This API is only for the specified platform, it can't work on the other
00024  * platform.
00025 */
00026 #define PLANTFORM_INCOMPATIBLE_API
00027 #define VQUARK_SPACE_BEGIN namespace VQuark {
00028 #define VQUARK_SPACE_END };
00029
00030 #include "VQuarkString.h"
00031
00032 #include <tuple>
00033 #include <functional>
00034 #include <vector>
00035 #include <map>
00036
00037 #ifdef _WIN64
00038 #include <Windows.h>
00039 #include <assert.h>
00040 #include <string>
00041
00042 #pragma comment(lib, "IMM32.LIB")
00043
00044 namespace VQuark {
00045     /* The Window handle define */
00046     class VWindowHandle {
00047     public:
00048         VWindowHandle();
00053         VWindowHandle(const HWND& Handle) PLANTFORM_INCOMPATIBLE_API;
00058         VWindowHandle(const VWindowHandle& Data);
00059
```

```
00060     public:
00065         const bool IsNull() const;
00066
00067     public:
00071         const HWND ToWinId() const PLANTFORM_INCOMPATIBLE_API;
00072
00073     public:
00074         friend const bool operator==(const VWindowHandle& Left, const VWindowHandle& Right);
00075         friend const bool operator!=(const VWindowHandle& Left, const VWindowHandle& Right);
00076
00077     public:
00081         HWND Handle;
00082     };
00083
00093     VWindowHandle   VQuarkCreateWindow(const VString& Title, const VString& ClassName, const size_t&
      Width, const size_t& Height,
00094         const VWindowHandle Parent = NULL);
00100     size_t          VQuarkGetWindowWidth(const VWindowHandle& Window);
00106     size_t          VQuarkGetWindowHeight(const VWindowHandle& Window);
00113     void            VQuarkMoveWindow(const VWindowHandle& Window, const size_t& X, const size_t& Y);
00119     void            VQuarkRenameWindow(const VWindowHandle& Window, const VString& Title);
00125     std::tuple<size_t, size_t>
00126                     VQuarkMurseWindow(const VWindowHandle& Window);
00132     void            VQuarkResizeWindow(const VWindowHandle& Window,
00133         const std::tuple<size_t, size_t>& Geomtery);
00138     void            VQuarkShowWindow(const VWindowHandle& Window);
00143     void            VQuarkHideWindow(const VWindowHandle& Window);
00144
00149     void            VAssert(const bool& Expression);
00150
00151     using VRawFont = LOGFONT;
00152
00156     struct VQuarkWinProcessPipeObject {
00157         std::tuple<size_t, size_t>  IMEPosition;
00158         VRawFont                    IMEFontStyle;
00159         bool                        IMEOperating = false;
00160
00161         bool Frameless  = false;
00162         bool MaxMinSize = false;
00163         bool Sizable    = false;
00164         bool Borderless = false;
00165
00166         std::tuple<size_t, size_t> MaxSize;
00167         std::tuple<size_t, size_t> MinSize;
00168
00169         std::function<void()>                       Repaint;
00170         std::function<void()>                       StartInput;
00171         std::function<void()>                       EndInput;
00172         std::function<void()>                       LoseFocus;
00173         std::function<bool()>                       Quit;
00174         std::function<void(std::vector<VString>)>   FileOnDrag;
00175         std::function<void(int, int)>               OnResize;
00176     };
00177
00186     LRESULT _VQuarkMsgDealy(HWND Handle, UINT Message, WPARAM WideParameter, LPARAM LowParameter);
00187
00196     template<class TypeLeft, class TypeRight>
00197     decltype(TypeLeft(0) + TypeRight(0)) VQuarkMurseCenter(const TypeLeft& Left, const TypeRight&
      Right) {
00198         return max(Left, Right) / 2 - min(Left, Right) / 2;
00199     }
00200
00201 #define VQUARK_WINDOWS
00202 #endif
00203 #elif __APPLE__
00204 #include <TargetConditionals.h>
00205 #if TARGET_IPHONE_SIMULATOR
00206 #   error "VQuark Error : Unsupported platform"
00207 #elif TARGET_OS_IPHONE
00208 #   error "VQuark Error : Unsupported platform"
00209 #elif TARGET_OS_MAC
00210 #   error "VQuark Error : Unsupported platform"
00211 #else
00212 #   error "VQuark Error : Unknown Apple platform"
00213 #endif
00214
00215 #   define VQUARK_APPLE
00216 #elif __linux__
00217 #   error "VQuark Error : Unsupported platform"
00218 #   define VQUARK_LINUX
00219 #elif __unix__
00220 #   error "VQuark Error : Unsupported platform"
00221 #   define VQUARK_UNIX
00222 #elif defined(_POSIX_VERSION)
00223 #   error "VQuark Error : Unsupported platform"
00224 #   define VQUARK_POSIX_LINX
00225 #else
```

```
00226 #   error "VQuark Error : Unsupported platform"
00227 #   define VQUARK_UNKNOWN
00228 #endif
00229 };
```

5.2 VQuarkData.h

```
00001 /*
00002  * VQuarkData.h (2023/5.26)
00003  *   This file defines some data types in VQuark (Point, Rectangle, e.t.c)
00004  *
00005  *
00006  * Copyright (C) 2023~now Margoo
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining a copy of this softwareand
      associated documentation files(the "Software"),
00009  * to deal in the Software without restriction, including without limitation the rights to use, copy,
      modify, merge, publish, distribute, sublicense, and /or sell copies of the Software,
00010  * and to permit persons to whom the Software is furnished to do so, subject to the following
      conditions :
00011  *
00012  * The above copyright noticeand this permission notice shall be included in all copies or substantial
      portions of the Software.
00013  *
00014  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
      NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00015  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
      HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00016  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
      DEALINGS IN THE SOFTWARE.
00017 */
00018
00019 #pragma once
00020
00021 #include <utility>
00022
00023 namespace VQuark {
00024
00029 template <class DataType> class VCoreRect {
00030 public:
00034     using Integer    =   const DataType&;
00038     using Rect       =   const VCoreRect&;
00039
00040 public:
00045     const DataType GetWidth() const {
00046         return Right - Left;
00047     }
00052     const DataType GetHeight() const {
00053         return Bottom - Top;
00054     }
00055
00056 public:
00062     void Move(Integer X, Integer Y) {
00063         Integer Width    =   GetWidth();
00064         Integer Height   =   GetHeight();
00065
00066         Left     =   X;
00067         Top      =   Y;
00068         Right    =   Y + Bottom;
00069         Bottom   =   Y + Height;
00070     }
00077     VCoreRect* MoveChaining(Integer X, Integer Y) {
00078         Move(X, Y);
00079
00080         return this;
00081     }
00089     void Extended(Integer LeftPanding, Integer TopPanding,
00090         Integer RightPanding, Integer BottomPanding) {
00091         Right    +=  RightPanding;
00092         Bottom   +=  BottomPanding;
00093         Left     -=  LeftPanding;
00094         Top      -=  TopPanding;
00095     }
00101     void Resize(Integer Width, Integer Height) {
00102         Right    =   Left     +   Width;
00103         Bottom   =   Top      +   Height;
00104     }
00109     void FusionRect(Rect Rectangle) {
00110         Left     = min(Rectangle.Left, Left);
00111         Right    = max(Rectangle.Right, Right);
00112         Top      = min(Rectangle.Top, Top);
00113         Bottom   = max(Rectangle.Bottom, Bottom);
00114     }
```

```
00120     bool Overlap(Rect JudgeRectangle) {
00121         return  max(Left, JudgeRectangle.Left)  <   min(Right, JudgeRectangle.Right) &&
00122                 max(Top, JudgeRectangle.Top)    <   min(Bottom, JudgeRectangle.Bottom);
00123     }
00129     bool Include(Rect Judgement) {
00130         return  Left <= Judgement.Left && Top <= Judgement.Top &&
00131                 Right <= Judgement.Right && Bottom >= Judgement.Bottom;
00132     }
00137     template <class TupleType, class RawType = DataType> TupleType ToQuadruple() {
00138         return TupleType{ static_cast<RawType>(Left), static_cast<RawType>(Top),
00139             static_cast<RawType>(Right), static_cast<RawType>(Bottom) };
00140     }
00141
00142 public:
00143     VCoreRect(Integer _ILeft, Integer _IRight, Integer _ITop, Integer _IBottom)
00144         : Left(_ILeft), Right(_IRight), Top(_ITop), Bottom(_IBottom) {
00145
00146     }
00147     VCoreRect(Rect Rectangle)
00148         : Left(Rectangle.Left), Right(Rectangle.Right), Top(Rectangle.Top), Bottom(Rectangle.Bottom) {
00149
00150     }
00151     VCoreRect()
00152         :   Left(reinterpret_cast<Integer>(0)), Right(reinterpret_cast<Integer>(0)),
00153             Top(reinterpret_cast<Integer>(0)), Bottom(reinterpret_cast<Integer>(0)) {
00154
00155     }
00156
00157 public:
00158     DataType Left;
00159     DataType Right;
00160     DataType Top;
00161     DataType Bottom;
00162 };
00167 template <class DataType> class VCorePoint {
00168 public:
00169     using Rect      = VCoreRect<DataType>;
00170     using ConstRect = const Rect&;
00171     using Integer   = const DataType&;
00172     using Point        = const VCorePoint&;
00173
00174 public:
00180     const bool InsideRectangle(ConstRect Judgement) const {
00181         return  Judgement.Left  >=  X && X <= Judgement.Right &&
00182                 Judgement.Top   >=  Y && Y <= Judgement.Bottom;
00183     }
00189     void Move(Integer X, Integer Y) {
00190         X = X;
00191         Y = Y;
00192     }
00198     void Offset(Integer XOF, Integer YOF) {
00199         X += XOF;
00200         Y += YOF;
00201     }
00202
00203 public:
00210     template<class TupleType, class RawType = DataType> TupleType ToTwoTuple() {
00211         return TupleType{ reinterpret_cast<RawType>(X), RawType<RawType>(Y) };
00212     }
00213
00214 public:
00215     friend bool operator==(ConstRect Left, ConstRect Right) {
00216         return Left.X == Right.X && Left.Y == Right.Y;
00217     }
00218     friend bool operator!=(ConstRect Left, ConstRect Right) {
00219         return Left.X != Right.X || Left.Y != Right.Y;
00220     }
00221
00222
00223 public:
00224     VCorePoint(Integer _IX, Integer _IY)
00225         : X(_IX), Y(_IY) {
00226
00227     }
00228     VCorePoint(Point Point) : X(Point.X), Y(Point.Y) {
00229
00230     }
00231     VCorePoint() : X(reinterpret_cast<Integer>(0)), Y(reinterpret_cast<Integer>(0)) {
00232
00233     }
00234
00235 public:
00236     DataType X;
00237     DataType Y;
00238 };
00239
00240 using VRect         =   VCoreRect<long>;
```

```
00241 using VRectFloat    =   VCoreRect<double>;
00242 using VPoint        =   VCorePoint<long>;
00243 using VPointF       =   VCorePoint<double>;
00244
00245 };
```

5.3 VQuarkFmt.h

```
00001 /*
00002  * VQuarkFmt.h (2023/5.20)
00003  *   The fmt lib in the VQuark
00004  *
00005  *
00006  * Copyright (C) 2023~now Margoo
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining a copy of this softwareand
      associated documentation files(the "Software"),
00009  * to deal in the Software without restriction, including without limitation the rights to use, copy,
      modify, merge, publish, distribute, sublicense, and /or sell copies of the Software,
00010  * and to permit persons to whom the Software is furnished to do so, subject to the following
      conditions :
00011  *
00012  * The above copyright noticeand this permission notice shall be included in all copies or substantial
      portions of the Software.
00013  *
00014  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
      NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00015  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
      HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00016  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
      DEALINGS IN THE SOFTWARE.
00017 */
00018 #pragma once
00019
00020 #include "VQuarkString.h"
00021
00022 #include <fstream>
00023 #include <tchar.h>
00024
00025 namespace VQuark {
00029     class VQuarkFMTDevice {
00030     public:
00031         VQuarkFMTDevice();
00032
00033     public:
00038         virtual void Output(const VString& String) = 0;
00039     };
00040
00044     class VQuarkFileStream : public VQuarkFMTDevice {
00045     public:
00046         VQuarkFileStream(const VString& Path);
00047
00048         void Output(const VString& String) override;
00049
00050     public:
00055         void Open(const VString& Path);
00059         void Close();
00060
00061     private:
00062         std::basic_ofstream<VChar, std::char_traits<VChar» Stream;
00063     };
00064
00065     using VQuarkFMTStream = VQuarkFMTDevice;
00066
00067     class VQuarkFMT {
00068     public:
00069         static void Print(const VString& String) {
00070             _tprintf(VStr("%s"), String.CStyleString());
00071         }
00072         template <class Input, class... Types>
00073         static void Print(const VString& String, Input _Input, Types... Args) {
00074             Print(String.Args(_Input), Args...);
00075         }
00082         template <class... Types>
00083         static void Print(const VString& String, Types... Args) {
00084             Print(String, Args...);
00085         }
00086
00087         static void PrintDevice(VQuarkFMTStream* Stream, const VString& String) {
00088             Stream->Output(String.CStyleString());
00089         }
00090         template <class Input, class... Types>
00091         static void PrintDevice(VQuarkFMTStream* Stream, const VString& String, Input _Input, Types...
      Args) {
```

```
00092             PrintDevice(Stream, String.Args(_Input), Args...);
00093         }
00100         template <class... Types>
00101         static void PrintDevice(VQuarkFMTStream* Stream, const VString& String, Types... Args) {
00102             PrintDevice(Stream, String, Args...);
00103         }
00104
00105         static VString Format(const VString& String) {
00106             return String;
00107         }
00108         template <class Input, class... Types>
00109         static VString Format(const VString& String, Input _Input, Types... Args) {
00110             return Format(String.Args(_Input), Args...);
00111         }
00119         template <class... Types>
00120         static VString Format(const VString& String, Types... Args) {
00121             return Format(String, Args...);
00122         }
00123     };
00124 }
```

5.4 VQuarkMessage.h

```
00001 /*
00002  * VQuarkMessage.h (2023/5.21)
00003  *   The message system in the VQuark
00004  *
00005  *
00006  * Copyright (C) 2023~now Margoo
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining a copy of this softwareand
      associated documentation files(the "Software"),
00009  * to deal in the Software without restriction, including without limitation the rights to use, copy,
      modify, merge, publish, distribute, sublicense, and /or sell copies of the Software,
00010  * and to permit persons to whom the Software is furnished to do so, subject to the following
      conditions :
00011  *
00012  * The above copyright noticeand this permission notice shall be included in all copies or substantial
      portions of the Software.
00013  *
00014  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
      NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00015  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
      HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00016  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
      DEALINGS IN THE SOFTWARE.
00017 */
00018
00019 #pragma once
00020
00021 #include "VQuarkBase.h"
00022
00023 VQUARK_SPACE_BEGIN
00024
00025 enum VMessageType
00026 {
00027     UnknowMessage,
00028     MouseMoveMessage,
00029     MouseClickedMessage,
00030     KeyClickedMessage,
00031     RepaintMessage,
00032     GetRepaintAeraMessage,
00033     IMECharMessage,
00034     MouseWheelMessage,
00035     FreeResourceMessage,
00036     CheckLocalFocusMessage,
00037     KillFocusMessage,
00038     QuitWindowMessage,
00039     BlurCombinationOnRend
00040 };
00041
00042 typedef class VMessage
00043 {
00044 private:
00045     VMessageType MessageType;
00046
00047 public:
00048     UINT   Win32ID;
00049     HWND   MessageTriggerWidget = NULL;
00050     WPARAM wParameter;
00051     LPARAM lParameter;
00052
00053 public:
```

```
00054     explicit VMessage(VMessageType Type = UnknowMessage);
00055
00056     VMessageType GetType()
00057     {
00058         return MessageType;
00059     }
00060 } VBasicMessage;
00061
00062 class VFreeSourceMessage : public VMessage
00063 {
00064 public:
00065     VFreeSourceMessage(HWND TriggerWidget);
00066 };
00067
00068 class VMouseMoveMessage : public VMessage
00069 {
00070 public:
00071     VPoint MousePosition;
00072
00073 public:
00074     VMouseMoveMessage(HWND TriggerWidget, int X, int Y);
00075 };
00076
00077 class VMouseWheelMessage : public VMessage
00078 {
00079 public:
00080     VPoint MousePosition;
00081
00082     short WheelValue;
00083
00084 public:
00085     VMouseWheelMessage(HWND TriggerWidget, int X, int Y, short WheelParameter);
00086 };
00087
00088 typedef enum VMouseClickedFlag
00089 {
00090     Down,
00091     Up
00092 } VkeyClickedFlag;
00093 enum VMouseKeyFlag
00094 {
00095     Middle,
00096     Left,
00097     Right
00098 };
00099
00100 class VMouseClickedMessage : public VMessage
00101 {
00102 public:
00103     VPoint MousePosition;
00104
00105     VMouseClickedFlag ClickedMethod;
00106     VMouseKeyFlag     ClickedKey;
00107
00108 public:
00109     VMouseClickedMessage(HWND TriggerWidget, int X, int Y, VMouseClickedFlag ClickedFlag,
    VMouseKeyFlag Key);
00110 };
00111
00112 class VKeyClickedMessage : public VMessage
00113 {
00114 public:
00115     byte            KeyVKCode;
00116     bool            KeyPrevDown;
00117     bool            KeyExtened;
00118     VkeyClickedFlag KeyStats;
00119
00120 public:
00121     VKeyClickedMessage(HWND TriggerWidget, byte VKCode, bool PrevDown, bool Extened, VkeyClickedFlag
    Stats);
00122 };
00123
00124 class VRepaintMessage : public VMessage
00125 {
00126 public:
00127     VRect DirtyRectangle;
00128
00129 public:
00130     explicit VRepaintMessage(HWND TriggerWidget, const VRect& RepaintRegion);
00131 };
00132
00133 class VGetRepaintAeraMessage : public VMessage
00134 {
00135 public:
00136     VRect* RepaintAera;
00137
00138 public:
```

```
00139     explicit VGetRepaintAeraMessage(HWND TriggerWidget, VRect& RepaintRegion);
00140     ~VGetRepaintAeraMessage();
00141 };
00142
00143 class VIMECharMessage : public VMessage
00144 {
00145 public:
00146     wchar_t IMEChar;
00147
00148 public:
00149     explicit VIMECharMessage(HWND TriggerWidget, wchar_t CharInputed);
00150 };
00151
00152 class VCheckFocusMessage : public VMessage
00153 {
00154 public:
00155     VPoint FocusPoint;
00156     void* Object;
00157     bool   Click;
00158
00159 public:
00160     explicit VCheckFocusMessage(HWND TriggerWidget, const VPoint& Point, void* MessageObject,
00161         const bool& MouseClick = false);
00162 };
00163
00164 class VKillFocusMessage : public VMessage
00165 {
00166 public:
00167     VKillFocusMessage(HWND TriggerWidget);
00168 };
00169
00170 class VQuitWindowMessage : public VMessage
00171 {
00172 public:
00173     VQuitWindowMessage(HWND TriggerWidget);
00174 };
00175
00176 VQUARK_SPACE_END
```

5.5 VQuarkSignal.h

```
00001 /*
00002  * VQuarkSignal.h (2023/5.27)
00003  *   The siganl slot in VQuark
00004  *
00005  *
00006  * Copyright (C) 2023~now Margoo
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining a copy of this softwareand
      associated documentation files(the "Software"),
00009  * to deal in the Software without restriction, including without limitation the rights to use, copy,
      modify, merge, publish, distribute, sublicense, and /or sell copies of the Software,
00010  * and to permit persons to whom the Software is furnished to do so, subject to the following
      conditions :
00011  *
00012  * The above copyright noticeand this permission notice shall be included in all copies or substantial
      portions of the Software.
00013  *
00014  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
      NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00015  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
      HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00016  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
      DEALINGS IN THE SOFTWARE.
00017 */
00018
00019 #pragma once
00020
00021 #include <functional>
00022 #include <list>
00023 #include <memory>
00024
00025 namespace VQuark {
00026 template <class... ParameterType> class ConnectBasic {
00027 public:
00028     explicit ConnectBasic(std::function<void(ParameterType...)> Function) {
00029         CallFunction = Function;
00030     }
00031     inline std::function<void(ParameterType...)>* GetFunction() {
00032         return &CallFunction;
00033     }
00034
00035     const bool IsBlock() const {
```

```
00036         return Blocked;
00037     }
00038     void SetBlock(const bool& Status) {
00039         Blocked = Status;
00040     }
00041
00042 private:
00043     std::function<void(ParameterType...)>   CallFunction;
00044     bool                                    Blocked = false;
00045 };
00046
00047 template <class... Type> class ConnectFunctional : public ConnectBasic<Type...> {
00048 public:
00049     using FunctionPointer = void (*)(Type...);
00050
00051 public:
00052     explicit ConnectFunctional(FunctionPointer Init)
00053         : ConnectBasic<Type...>(std::function<void(Type...)>(Init)) {
00054         Function = Init;
00055     }
00056     inline FunctionPointer GetPointer() {
00057         return Function;
00058     }
00059
00060 private:
00061     FunctionPointer Function;
00062 };
00063 template <class ObjectType, class... Type> class Connection : public ConnectBasic<Type... > {
00064 public:
00065     using ObjectPointer = void (ObjectType::*)(Type...);
00066
00067 public:
00068     Connection(ObjectType* ObjectPointer, ObjectPointer Function)
00069         : ConnectBasic<Type...>([ObjectPointer, Function](Type... Agrument) {
      (*ObjectPointer.*Function)(Agrument...); }) {
00070         Object = ObjectPointer;
00071     }
00072
00073     inline void* GetRawObject() {
00074         return Object;
00075     }
00076     inline ObjectPointer GetRawFunction() {
00077         return ObjectFunction;
00078     }
00079
00080 private:
00081     ObjectType*     Object;
00082     ObjectPointer   ObjectFunction;
00083 };
00084
00088 enum class VQuarkSignalOperation {
00089     Del, Block, Disblock
00090 };
00091
00096 template <class... Type> class VSignal {
00097 private:
00098     void _Operator(void (*Function)(Type...), VQuarkSignalOperation OperatorStage) {
00099         for (auto Iterator = Slots->begin; Iterator != Slots->end();) {
00100             ConnectBasic<Type...>* ConnectFunction = static_cast<ConnectBasic<Type...>
      *>((*Iterator));
00101             if (ConnectFunction->GetFunction() == Function) {
00102                 switch (OperatorStage) {
00103                 case VQuarkSignalOperation::Del: {
00104                     Slots->erase(Iterator++);
00105
00106                     break;
00107                 }
00108                 case VQuarkSignalOperation::Block: {
00109                     ConnectFunction->SetBlock(true);
00110
00111                     ++Iterator;
00112
00113                     break;
00114                 }
00115                 case VQuarkSignalOperation::Disblock: {
00116                     ConnectFunction->SetBlock(false);
00117
00118                     ++Iterator;
00119
00120                     break;
00121                 }
00122                 }
00123             }
00124             else {
00125                 ++Iterator;
00126             }
00127         }
```

```
00128     }
00129
00130     template <class ObjectType> void _Operator(ObjectType* Object,
00131         void (ObjectType::*ObjectFunction)(Type...), VQuarkSignalOperation OperatorStage) {
00132         for (auto Iterator = Slots->begin; Iterator != Slots->end();) {
00133             Connection<ObjectType, Type...>* ConnectFunction = static_cast<Connection<ObjectType,
      Type...> *>((*Iterator));
00134             if (ConnectFunction->GetRawFunction() == ObjectFunction &&
      ConnectFunction->GetRawFunction() == Object) {
00135                 switch (OperatorStage) {
00136                 case VQuarkSignalOperation::Del: {
00137                     Slots->erase(Iterator++);
00138
00139                     break;
00140                 }
00141                 case VQuarkSignalOperation::Block: {
00142                     ConnectFunction->SetBlock(true);
00143
00144                     ++Iterator;
00145
00146                     break;
00147                 }
00148                 case VQuarkSignalOperation::Disblock: {
00149                     ConnectFunction->SetBlock(false);
00150
00151                     ++Iterator;
00152
00153                     break;
00154                 }
00155                 }
00156             }
00157             else {
00158                 ++Iterator;
00159             }
00160         }
00161     }
00162
00163 public:
00164     inline void Connect(void (*Function)(Type...)) {
00165         std::shared_ptr<ConnectFunctional<Type...>> FunctionPointer(new
      ConnectFunctional<Type...>(Function));
00166         Slots->push_back(FunctionPointer);
00167     }
00168     template <class ObjectType> inline void Connect(ObjectType* Object, void
      (ObjectType::*Function)(Type...)) {
00169         std::shared_ptr<Connection<ObjectType, Type...>> FunctionPointer(new
      Connection<ObjectType, Type...>(Object, Function));
00170         Slots->push_back(FunctionPointer);
00171     }
00172     template <class ObjectType> inline void Disconnect(ObjectType* Object, void (ObjectType::*
      Function)(Type...)) {
00173         _Operator(Object, Function, VQuarkSignalOperation::Del);
00174     }
00175     void Block(void (*Function)(Type...), bool BlockStatus) {
00176         _Operator(Function, BlockStatus ? VQuarkSignalOperation::Block :
      VQuarkSignalOperation::Disblock);
00177     }
00178     template <class ObjectType> void Block(ObjectType* Object, void (*Function)(Type...), bool
      BlockStatus) {
00179         _Operator(Function, Object, BlockStatus ? VQuarkSignalOperation::Block :
      VQuarkSignalOperation::Disblock);
00180     }
00181     void Emit(Type... Agruments) {
00182         for (auto Iterator = Slots->begin(); Iterator != Slots->end(); ++Iterator) {
00183             if ((*Iterator)->IsBlock() == true) {
00184                 continue;
00185             }
00186
00187             auto Function = (*Iterator)->GetFunction();
00188
00189             (*Function)(Agruments...);
00190         }
00191     }
00192
00193 public:
00194     VSignal() {
00195         Slots = new std::list<std::shared_ptr<ConnectBasic<Type...>>>;
00196     }
00197     ~VSignal() {
00198         delete Slots;
00199     }
00200
00201 private:
00202     std::list<std::shared_ptr<ConnectBasic<Type...>>* Slots;
00203 };
00204
00205 }
```

5.6 VQuarkString.h

```
00001 /*
00002  * VQuarkString.h (2023/5.20)
00003  *   The wrapper of the STL string
00004  *
00005  *
00006  * Copyright (C) 2023~now Margoo
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining a copy of this softwareand
      associated documentation files(the "Software"),
00009  * to deal in the Software without restriction, including without limitation the rights to use, copy,
      modify, merge, publish, distribute, sublicense, and /or sell copies of the Software,
00010  * and to permit persons to whom the Software is furnished to do so, subject to the following
      conditions :
00011  *
00012  * The above copyright noticeand this permission notice shall be included in all copies or substantial
      portions of the Software.
00013  *
00014  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
      NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00015  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
      HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00016  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
      DEALINGS IN THE SOFTWARE.
00017 */
00018 #pragma once
00019
00020 #include <comdef.h>
00021 #include <string>
00022
00023 #include "VQuarkData.h"
00024
00025 #pragma warning(disable : 4455)
00026
00027 namespace VQuark {
00028
00029 #pragma comment(lib, "comsuppw.lib")
00030 #ifdef _UNICODE
00031 #define VStr(Str) L##Str
00032 #define VSTCStr   std::to_wstring
00033
00034 using VChar = wchar_t;
00035 using VProxyString = std::wstring;
00036
00037 const wchar_t* operator"" vs(const wchar_t* OriginString, size_t StringLength);
00038 const wchar_t* operator"" vs(const char* OriginString, size_t StringLength);
00039
00040 const wchar_t* vstring_convert(const wchar_t* OriginString);
00041 const wchar_t* vstring_convert(const char* OriginString);
00042 #else
00043 #define VStr(Str) ##Str
00044 #define VSTCStr   std::to_string
00045
00046 #pragma comment(lib, "comsupp.lib")
00047 using VChar = char;
00048 using VProxyString = std : string;
00049
00050 const char* operator"" vs(const wchar_t* OriginString, size_t StringLength);
00051 const char* operator"" vs(const char* OriginString, size_t StringLength);
00052
00053 const wchar_t* vstring_convert(const wchar_t* OriginString);
00054 const wchar_t* vstring_convert(const char* OriginString);
00055 #endif
00056
00060 class VString : public VProxyString {
00061 public:
00065     using Iterator = VProxyString::iterator;
00069     using ConstIterator = VProxyString::const_iterator;
00073     using ReverseIterator = VProxyString::reverse_iterator;
00074
00075 public:
00079     static constexpr auto NoPosition{ static_cast<size_type>(-1) };
00080
00081 public:
00082     VString();
00083     VString(const std::wstring& String);
00084     VString(const std::string& String);
00085     VString(const wchar_t* String);
00086     VString(wchar_t* String);
00087     VString(const char* String);
00088     VString(char* String);
00089
00090 public:
00096     static VString FromNumber(const int& NumberConvert);
00102     static VString FromNumber(const long long& NumberConvert);
00108     static VString FromNumber(const long& NumberConvert);
```

```
00114     static VString FromNumber(const unsigned int& NumberConvert);
00120     static VString FromNumber(const unsigned long& NumberConvert);
00126     static VString FromNumber(const unsigned long long& NumberConvert);
00132     static VString FromString(const std::string& String);
00138     static VString FromWideString(const std::wstring& String);
00139
00140 public:
00147     VString Split(const size_t& Begin, const size_t& SplitCount);
00154     VString Split(const size_t& Begin, const size_t& SplitCount) const;
00161     VString SplitRange(const size_t& Begin, const size_t& End);
00162
00167     void Append(const VString& AppendString);
00168
00175     bool StartWith(const VString& JudgeString, const size_t& StartOn = 0);
00180     bool EndWith(const VString& JudgeString);
00181
00187     inline VChar& At(const size_t& Position);
00188
00193     bool IsEmpty();
00194
00200     void Fill(const VChar& Character, size_t FillSize = 0);
00205     void Set(const VString& String);
00211     void Erase(const size_t& Begin, const size_t& Count);
00216     void Erase(ConstIterator Iterator);
00222     void EraseRange(const size_t& Begin, const size_t& End);
00228     void Insert(const size_t& Position, const VString& String);
00235     void Insert(const size_t& Position, const VString& String, const size_t& Count);
00241     void Insert(ConstIterator Iterator, const VChar& Character);
00242
00249     size_t IndexOf(const VString& String, const size_t& StartAt = 0) const;
00256     size_t IndexLastOf(const VString& String, const size_t& StartAt = 0);
00257
00262     Iterator        Begin();
00267     Iterator        End();
00272     ReverseIterator ReverseBegin();
00277     ReverseIterator ReverseEnd();
00278
00284     VString Args(VString FormatInstance) const;
00290     VString Args(int IntFormat) const;
00296     VString Args(const long long IntFormat) const;
00302     VString Args(const long IntFormat) const;
00308     VString Args(const unsigned int IntFormat) const;
00314     VString Args(const unsigned long IntFormat) const;
00320     VString Args(const unsigned long long IntFormat) const;
00326     VString Args(const VPoint Format) const;
00332     VString Args(const VRect Format) const;
00333
00338     const VChar* CStyleString() const;
00339
00344     size_t Size() const;
00349     size_t Length() const;
00350 };
00351
00352 }
```

5.7 VQuarkSys.h

```
00001 /*
00002  * VQuarkSys.h (2023/5.27)
00003  *   Some system API operation's wrapper
00004  *
00005  *
00006  * Copyright (C) 2023~now Margoo
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining a copy of this softwareand
      associated documentation files(the "Software"),
00009  * to deal in the Software without restriction, including without limitation the rights to use, copy,
      modify, merge, publish, distribute, sublicense, and /or sell copies of the Software,
00010  * and to permit persons to whom the Software is furnished to do so, subject to the following
      conditions :
00011  *
00012  * The above copyright noticeand this permission notice shall be included in all copies or substantial
      portions of the Software.
00013  *
00014  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
      NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00015  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
      HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00016  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
      DEALINGS IN THE SOFTWARE.
00017 */
00018
00019 #pragma once
```

```
00020
00021 #include "VQuarkData.h"
00022 #include "VQuarkBase.h"
00023
00024 VQUARK_SPACE_BEGIN
00025
00026 class VQuarkScreenDevice {
00027 public:
00032     static const unsigned short GetWidth();
00037     static const unsigned short GetHeight();
00042     static const unsigned short MurseClientWidth();
00047     static const unsigned short MurseClientHeight();
00048 };
00049
00050 VQUARK_SPACE_END
```

5.8 VQuarkWidget.h

```
00001 /*
00002  * VQuarkWidget.h (2023/5.20)
00003  *   The wrapper of the widget operation
00004  *
00005  *
00006  * Copyright (C) 2023~now Margoo
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining a copy of this softwareand
      associated documentation files(the "Software"),
00009  * to deal in the Software without restriction, including without limitation the rights to use, copy,
      modify, merge, publish, distribute, sublicense, and /or sell copies of the Software,
00010  * and to permit persons to whom the Software is furnished to do so, subject to the following
      conditions :
00011  *
00012  * The above copyright noticeand this permission notice shall be included in all copies or substantial
      portions of the Software.
00013  *
00014  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
      NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00015  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
      HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00016  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
      DEALINGS IN THE SOFTWARE.
00017 */
00018 #pragma once
00019
00020 #include "VQuarkBase.h"
00021 #include "VQuarkSys.h"
00022 #include "VQuarkString.h"
00023
00024 VQUARK_SPACE_BEGIN
00025
00026 class VQuarkWidget {
00027 public:
00032     VQuarkWidget(VWindowHandle WidgetHandle);
00033
00034 public:
00044     static VQuarkWidget* CreateWidget(const VString& Class, const VString& Title, const size_t& Width,
      const size_t& Height,
00045         const VWindowHandle& BelongTo = NULL) {
00046         return new VQuarkWidget(VQuarkCreateWindow(Title, Class, Width, Height, BelongTo));
00047     }
00048
00049 public:
00054     void SetTitle(const VString& Title);
00060     void Resize(const size_t Width, const size_t Height);
00066     void Move(const size_t& X, const size_t& Y);
00070     void Show();
00074     void Hide();
00078     void MoveCenter();
00083     inline const size_t GetWidth() const noexcept;
00088     inline const size_t GetHeight() const noexcept;
00089
00094     const VWindowHandle GetHandle() const noexcept;
00095
00096 private:
00097     VWindowHandle Handle;
00098 };
00099
00100 VQUARK_SPACE_END
```

Index