Cat Mouse

High Distinction Project for HIT2302 Object Oriented Programming

Generated by Doxygen 1.8.5

Sat Nov 9 2013 22:15:20

Contents

1 C++ Decoupled Implementation

Version

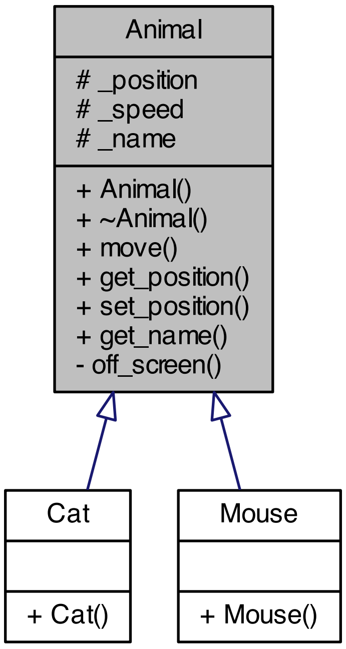
1

2 Class Documentation

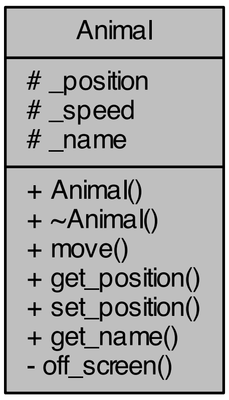
2.1 Animal Class Reference

```
#include <Animal.h>
```

Inheritance diagram for Animal:



Collaboration diagram for Animal:



Public Member Functions

- **Animal** ()
- virtual ∼**Animal** ()=0

- void **move** (**dirs** dir)
- point2d ∗const **get_position** ()
- void **set_position** (point2d ∗newPos)
- string const **get_name** ()

Protected Attributes

- point2d ∗ **_position**
- const int **_speed** = 3
- string **_name**

Private Member Functions

- void **off_screen** ()

2.1.1 Detailed Description

Defines an abstract, base class for a playable 'thing' on the screen which can move around etc.

Defines class for the general 'game' of the cat and mice.

Author

Alex Cummaudo

Date

16 Oct 2013

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Animal::Animal ( )

Default constructor for initialising _position and _speed for all new Animals.

2.1.2.2 Animal::∼Animal ( ) [pure virtual]

Destructor reliquishes resources created in this class.

2.1.3 Member Function Documentation

2.1.3.1 void Animal::move (dirs *dir*)

Move implementation for a **Animal** (p. 1) to move an animal in a direction at its speed.

Parameters

dir	Direction the animal is told to move in (alters x and y axis position of poisition accordingly)

2.1.3.2 point2d∗ const Animal::get_position ( )

Read property to get the position of the **Animal** (p. 1)

Returns

Position of the animal

2.1.3.3 void Animal::set_position (point2d ∗ *newPos*)

Write property to set the position of the **Animal** (p. 1)

Parameters

newPos	New position to set the animal at

2.1.3.4 string const Animal::get_name ( )

Readonly property to get the name of the animal

Returns

Name of the animal

2.1.3.5 void Animal::off_screen ( ) `[private]`

Off screen check that prevents any **Animal** (p. 1) from going outside the borders of the screen.

2.1.4 Member Data Documentation

2.1.4.1 point2d∗ Animal::_position `[protected]`

Centrepoint position of the animal.

2.1.4.2 const int Animal::_speed = 3 `[protected]`

Speed at which animals move at, set to a value of 3.

2.1.4.3 string Animal::_name `[protected]`

Name of animals, overriden by children (i.e. 'Cat' or 'Mouse')

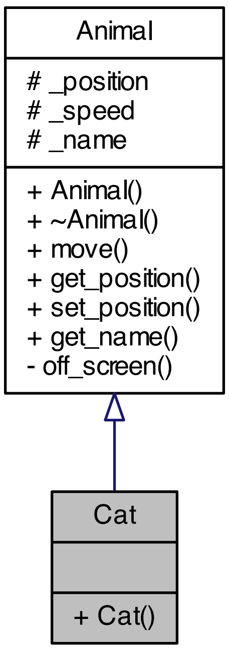
The documentation for this class was generated from the following files:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Animal.h**
- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Animal.cpp**

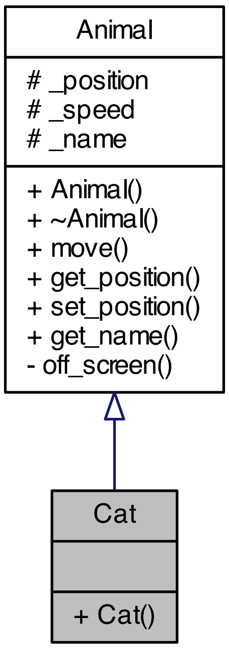
2.2 Cat Class Reference

```
#include <CatMouse.hpp>
```

Inheritance diagram for Cat:



Collaboration diagram for Cat:



Public Member Functions

- **Cat** ()

Additional Inherited Members

2.2.1 Detailed Description

Defines an class for a playable chaser (i.e. the chasing cat)

Author

Alex Cummaudo

Date

18 Oct 2013

2.2.2 Constructor & Destructor Documentation

2.2.2.1 Cat::Cat ( )

The default constructor for the cat constructs parent and sets position on lefthand-side of screen.

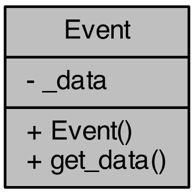
The documentation for this class was generated from the following file:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**CatMouse.hpp**

2.3 Event Class Reference

`#include <Event.h>`

Collaboration diagram for Event:



Public Member Functions

- **Event** (map< string, string > data)
- map< string, string > const **get_data** ()

Private Attributes

- map< string, string > **_data**

2.3.1 Detailed Description

Defines event class of what to pass a view, thereby allowing a link between each view and each model.

Author

Alex Cummaudo

Date

7 Oct 2013

2.3.2 Constructor & Destructor Documentation

2.3.2.1 Event::Event (map< string, string > *data*)

Constructor for new event object to initialise fields.

Parameters

data	Textual data to insert as data to this event

2.3.3 Member Function Documentation

2.3.3.1 map<string, string> const Event::get_data ( )

Readonly property to data.

2.3.4 Member Data Documentation

2.3.4.1 map<string, string> Event::_data `[private]`

Textual data contained within the **Event** (p. 8).

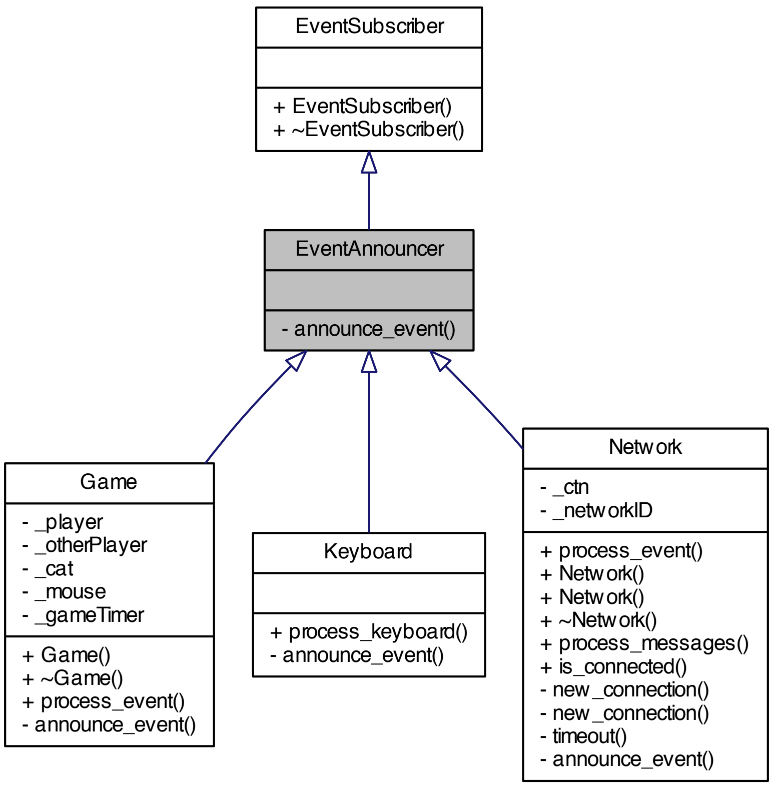
The documentation for this class was generated from the following files:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Event.h**
- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Event.cpp**

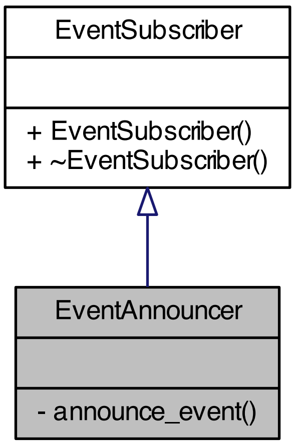
2.4 EventAnnouncer Interface Reference

```
#include <EventAnnouncer.h>
```

Inheritance diagram for EventAnnouncer:



Collaboration diagram for EventAnnouncer:



Private Member Functions

- virtual void **announce_event** (string msg)=0

Additional Inherited Members

2.4.1 Detailed Description

An pure abstract class that defines all the methods that each announcer of events must implement.

Author

Alex Cummaudo

Date

7 Oct 2013

Note

Inherits as virtual so that any users of BOTH EventProcessors and EventSubscribers will avoid diamond inheritance issues.

2.4.2 Member Function Documentation

2.4.2.1 virtual void EventAnnouncer::announce_event (string *msg*) `[private]`, `[pure virtual]`

Defines that whoever uses this interface must announce events with given a message to the **EventManager** (p. 11).

Parameters

msg	Message to announce when creating an **Event** (p. 8)

Implemented in **Network** (p. 31), **Game** (p. 18), and **Keyboard** (p. 24).

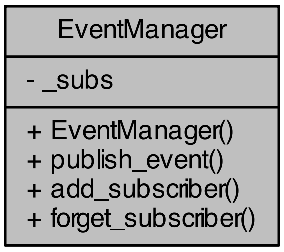
The documentation for this interface was generated from the following file:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**EventAnnouncer.h**

2.5 EventManager Class Reference

```
#include <EventManager.h>
```

Collaboration diagram for EventManager:



Public Member Functions

- **EventManager** ()

Static Public Member Functions

- static void **publish_event** (**Event** $*$eData)
- static void **add_subscriber** (**EventSubscriber** $*$sub)
- static void **forget_subscriber** (**EventSubscriber** $*$sub)

Static Private Attributes

- static vector< **EventSubscriber** $*$ > $*$ **_subs**

2.5.1 Detailed Description

Defines **EventManager** (p. 11) class which processes each event to each kind of **EventProcessor** (p. 12).

Author

Alex Cummaudo

Date

7 Oct 2013

Note

This is a static member class; so that clients do not need to make an instance of an **EventManager** (p. 11) (since there's only ever going to be one processor). Therefore we invoke **EventManager** (p. 11) by calling directly on the class (i.e. EventManager::notify_subscribers(event))

2.5.2 Constructor & Destructor Documentation

2.5.2.1 EventManager::EventManager ( )

Constructor initialises _subs vector.

2.5.3 Member Function Documentation

2.5.3.1 void EventManager::publish_event (Event ∗ *eData*) `[static]`

Processes the event for each kind subscriber who publishes events (i.e. EventProcessors ONLY!)

Parameters

eData	**Event** (p. 8) to publish to all EventProcessors

2.5.3.2 void EventManager::add_subscriber (EventSubscriber ∗ *sub*) `[static]`

Adds a subscriber to the _subs vector.

Parameters

sub	Subscriber to manage

2.5.3.3 void EventManager::forget_subscriber (EventSubscriber ∗ *sub*) `[static]`

Removes a subscriber to the _subs vector.

Parameters

sub	Subscriber to forget

2.5.4 Member Data Documentation

2.5.4.1 vector< EventSubscriber ∗ > ∗ EventManager::_subs `[static]`, `[private]`

Declare subscribers (who process or announce events) vector.

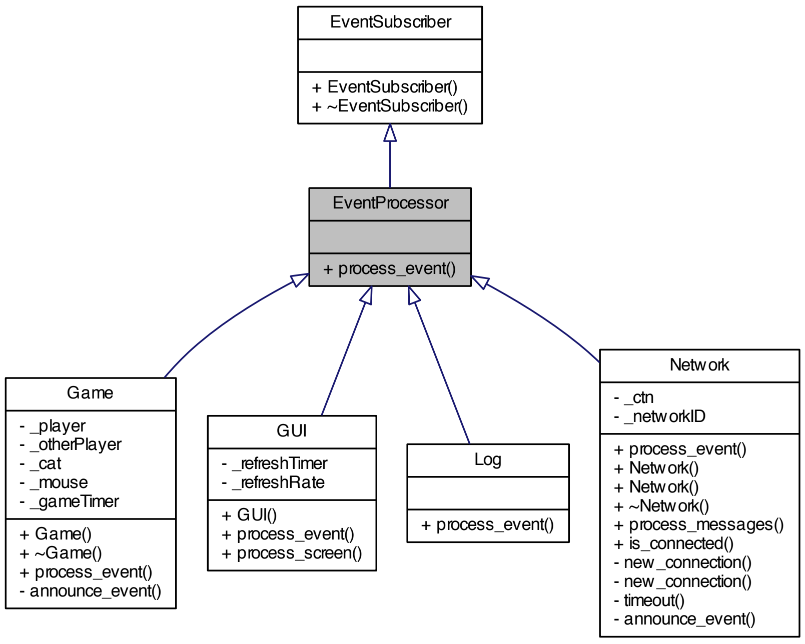
The documentation for this class was generated from the following files:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**EventManager.h**
- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**EventManager.cpp**

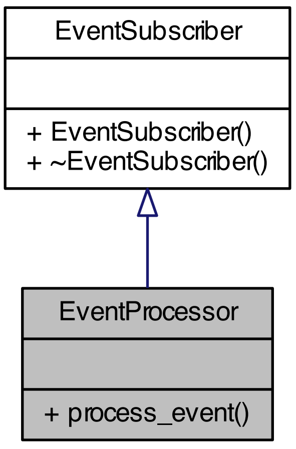
2.6 EventProcessor Interface Reference

`#include <EventProcessor.h>`

Inheritance diagram for EventProcessor:



Collaboration diagram for EventProcessor:



Public Member Functions

- virtual void **process_event** (**Event** ∗eData)=0

2.6.1 Detailed Description

An pure abstract class that defines all the methods that each processor of events must implement.

Author

Alex Cummaudo

Date

7 Oct 2013

Note

Inherits as virtual so that any users of BOTH EventProcessors and EventSubscribers will avoid diamond inheritance issues.

2.6.2 Member Function Documentation

2.6.2.1 virtual void EventProcessor::process_event (**Event** ∗ *eData*) `[pure virtual]`

Defines that whoever uses this interface must process an event in anyway with the given **Event** (p. 8).

Parameters

eData	**Event** (p. 8) Data to process

Implemented in **Game** (p. 18), **GUI** (p. 21), **Network** (p. 30), and **Log** (p. 25).

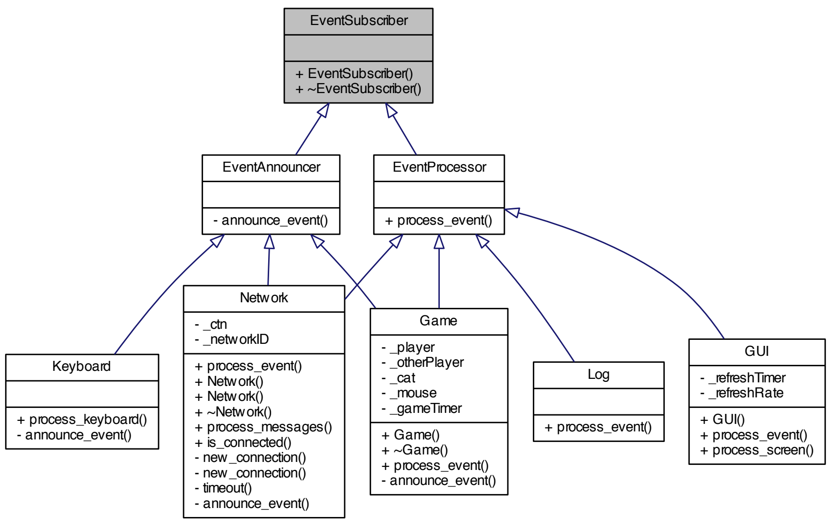
The documentation for this interface was generated from the following file:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**EventProcessor.h**

2.7 EventSubscriber Interface Reference

`#include <EventSubscriber.h>`

Inheritance diagram for EventSubscriber:



Collaboration diagram for EventSubscriber:

Public Member Functions

- **EventSubscriber** ()
- virtual ∼**EventSubscriber** ()

2.7.1 Detailed Description

Acts as a parent to subscribers and announcers so that the **Event** (p. 8) manager knows what to manager (i.e. both Announcers and Processors, and this allows this relationship to occur via inheritance)

Author

Alex Cummaudo

Date

7 Oct 2013

2.7.2 Constructor & Destructor Documentation

2.7.2.1 EventSubscriber::EventSubscriber ( )

To dynamically add event subscribers to the **EventManager** (p. 11) on creation, the **EventSubscriber** (p. 14) constructor does this for us.

2.7.2.2 virtual EventSubscriber::∼EventSubscriber ( ) `[virtual]`

To make **EventSubscriber** (p. 14) polymorphic, make a virtual destructor—this will allow for dynamic casting in the **EventManager** (p. 11). On invocation, the **EventManager** (p. 11) will forget about this subscriber.

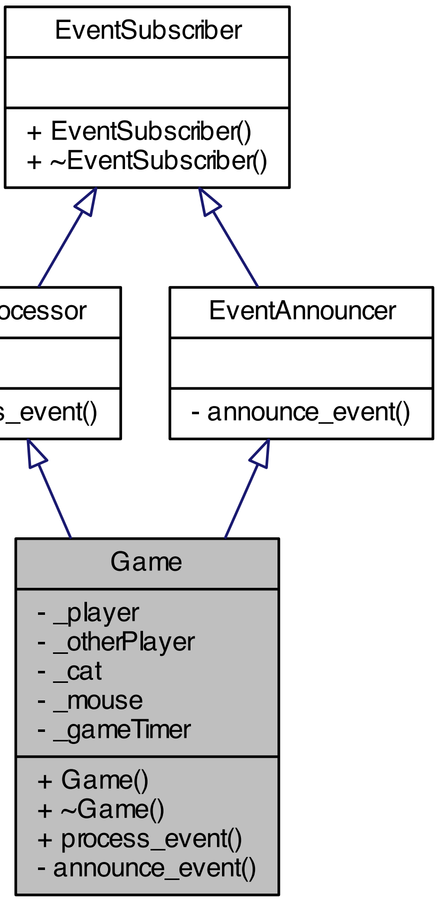
The documentation for this interface was generated from the following file:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**EventSubscriber.h**

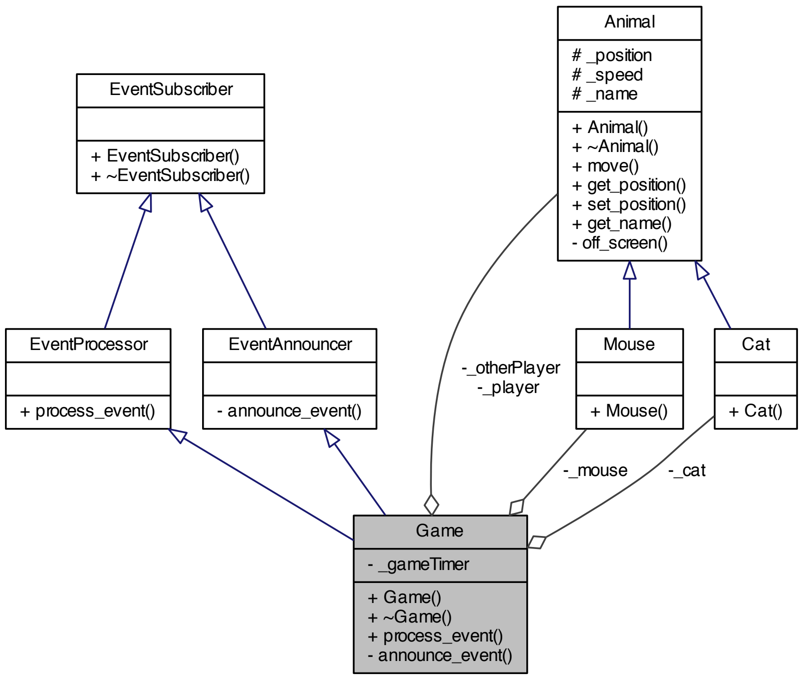
2.8 Game Class Reference

```
#include <Game.h>
```

Inheritance diagram for Game:



Collaboration diagram for Game:



Public Member Functions

- **Game** (bool asCat)
- ∼**Game** ()
- virtual void **process_event** (**Event** ∗eData)

Private Member Functions

- virtual void **announce_event** (string msg)

Private Attributes

- **Animal** ∗ **_player**
- **Animal** ∗ **_otherPlayer**
- **Cat** ∗ **_cat**
- **Mouse** ∗ **_mouse**
- timer **_gameTimer**

2.8.1 Detailed Description

Defines class for the general 'game' of the cat and mice.

Author

Alex Cummaudo

Date

16 Oct 2013

2.8.2 Constructor & Destructor Documentation

2.8.2.1 Game::Game (bool *asCat*)

On construction of a game, a cat and a mouse will be created—if the asCat is true, player references the cat, else it will reference the mouse.

Parameters

asCat	Where true, the game initialises the player as the cat, with the other player as the mouse.

2.8.2.2 Game::∼Game ( )

Destructor reliquishes resources created in this class.

2.8.3 Member Function Documentation

2.8.3.1 void Game::process_event (Event ∗ *eData*) `[virtual]`

Recieves events from the **EventManager** (p. 11) to set the coordinates of the players to either a specified location (_otherPlayer) or to move the _player according to key events.

Parameters

eData	**Event** (p. 8) Data to process

Implements **EventProcessor** (p. 14).

2.8.3.2 void Game::announce_event (string *msg*) `[private]`, `[virtual]`

Announces that the player moved (called by key_check on a key move) to all of **EventManager** (p. 11)'s Event-Processors—i.e. to announce updates of the model.

Parameters

msg	Message to announce

Implements **EventAnnouncer** (p. 10).

2.8.4 Member Data Documentation

2.8.4.1 Animal∗ Game::_player `[private]`

Player of the game (person controlling the game)

2.8.4.2 Animal∗ Game::_otherPlayer `[private]`

Other player in the game (other person controlling enemy)

2.8.4.3 Cat∗ Game::_cat `[private]`

Cat (p. 5) (chaser) of the game.

2.8.4.4 Mouse∗ Game::_mouse `[private]`

Mouse (p. 26) (chasee) of the game.

2.8.4.5 timer Game::_gameTimer `[private]`

Ingame-timer to keep track of how long game has gone for.

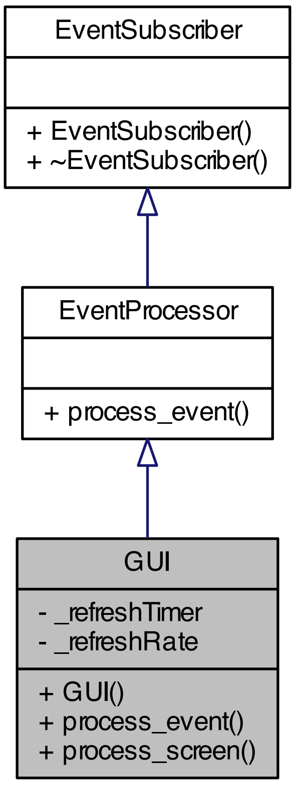
The documentation for this class was generated from the following files:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Game.h**
- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Game.cpp**

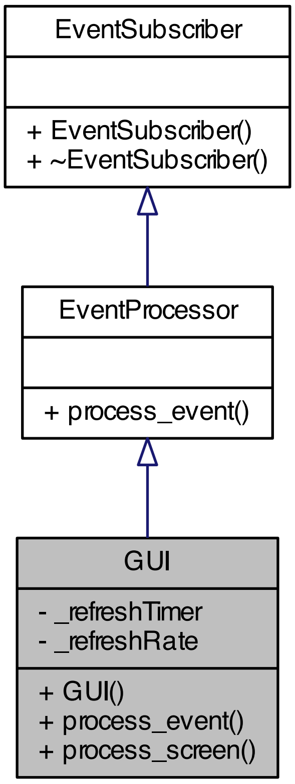
2.9 GUI Class Reference

`#include <GUI.h>`

Inheritance diagram for GUI:



Collaboration diagram for GUI:



Public Member Functions

- **GUI** (float refRate)
- virtual void **process_event** (**Event** ∗eData)
- void **process_screen** ()

Private Attributes

- timer **_refreshTimer**
- float **_refreshRate**

2.9.1 Detailed Description

Provides **GUI** (p. 19) View for the game to display the game on in a graphics window.

Author

Alex Cummaudo

Date

19 Oct 2013

2.9.2 Constructor & Destructor Documentation

2.9.2.1 GUI::GUI (float *refRate*)

Constructor for **GUI** (p. 19) view creates a graphics window for SwinGame.

2.9.3 Member Function Documentation

2.9.3.1 void GUI::process_event (**Event** ∗ *eData*) `[virtual]`

Processes events by drawing only **Game** (p. 15) events onto the screen given the **Game** (p. 15) event is about an **Animal** (p. 1).

Parameters

eData	**Event** (p. 8) Data to process

Implements **EventProcessor** (p. 14).

2.9.3.2 void GUI::process_screen ( )

Clears and refreshes the screen by the time given in reset timer.

2.9.4 Member Data Documentation

2.9.4.1 timer GUI::_refreshTimer `[private]`

Timer used to refresh the screen at the by clearing the screen and resetting at refreshRate given

2.9.4.2 float GUI::_refreshRate `[private]`

Seconds to refresh the screen at.

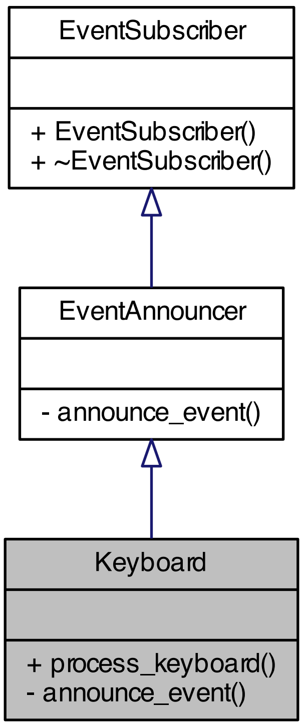
The documentation for this class was generated from the following files:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**GUI.h**
- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**GUI.cpp**

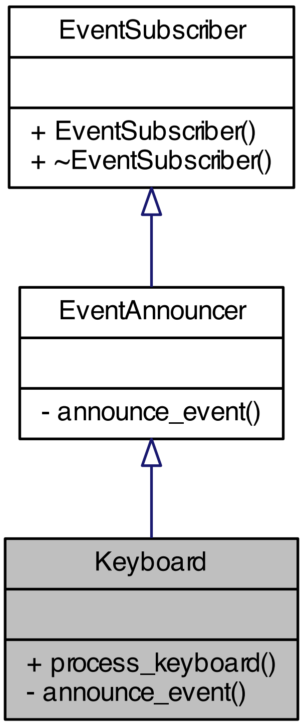
2.10 Keyboard Class Reference

`#include <Keyboard.h>`

Inheritance diagram for Keyboard:



Collaboration diagram for Keyboard:



Public Member Functions

- void **process_keyboard** ()

Private Member Functions

- virtual void **announce_event** (string key)

2.10.1 Detailed Description

Defines class to capture keyboard events and pass them to the processor.

Author

Alex Cummaudo

Date

19 Oct 2013

2.10.2 Member Function Documentation

2.10.2.1 void Keyboard::process_keyboard ( )

Captures keydown events and processes them by passing them to the **EventManager** (p. 11).

2.10.2.2 void Keyboard::announce_event (string *key*) [private], [virtual]

Sends an event to all subscribers with the given key.

Parameters

msg	Message to announce when creating an **Event** (p. 8)

Implements **EventAnnouncer** (p. 10).

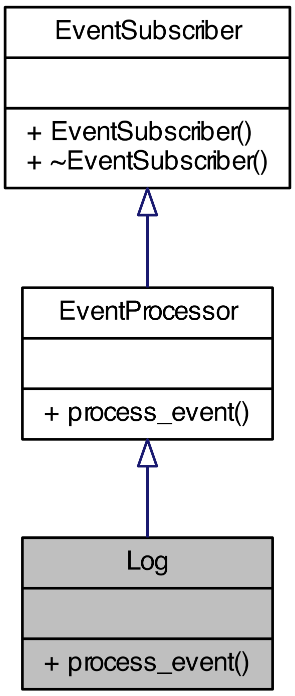
The documentation for this class was generated from the following files:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Keyboard.h**
- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Keyboard.cpp**

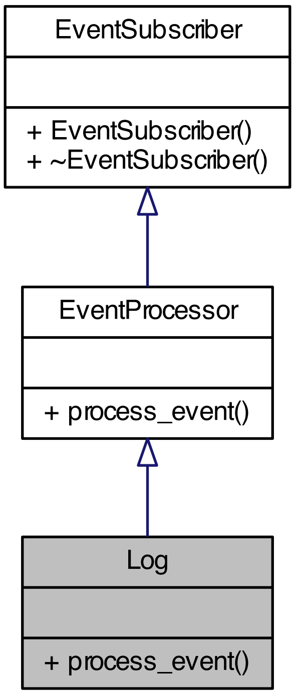
2.11 Log Class Reference

```
#include <Log.h>
```

Inheritance diagram for Log:



Collaboration diagram for Log:



Public Member Functions

- virtual void **process_event** (**Event** ∗eData)

2.11.1 Detailed Description

CLI view to game.

Author

Alex Cummaudo

Date

7 Oct 2013

2.11.2 Member Function Documentation

2.11.2.1 void Log::process_event (**Event** ∗ ***eData***) `[virtual]`

Processes a data depending on the data passed by printing each key/value pair in the event data's map, priting lines to cout.

Parameters

eData	**Event** (p. 8) Data to process

Implements **EventProcessor** (p. 14).

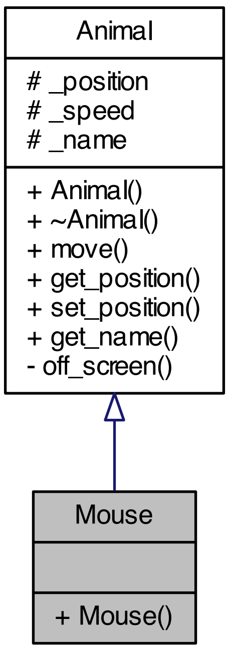
The documentation for this class was generated from the following files:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Log.h**
- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Log.cpp**

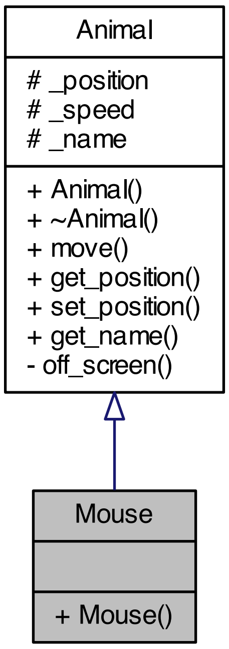
2.12 Mouse Class Reference

```
#include <CatMouse.hpp>
```

Inheritance diagram for Mouse:



Collaboration diagram for Mouse:



Public Member Functions

- **Mouse** ()

Additional Inherited Members

2.12.1 Detailed Description

Defines an class for a playable chasee (i.e. the hunted mouse)

Author

Alex Cummaudo

Date

18 Oct 2013

2.12.2 Constructor & Destructor Documentation

2.12.2.1 Mouse::Mouse ( )

The default constructor for the mouse constructs parent and sets position on righthand-side of screen.

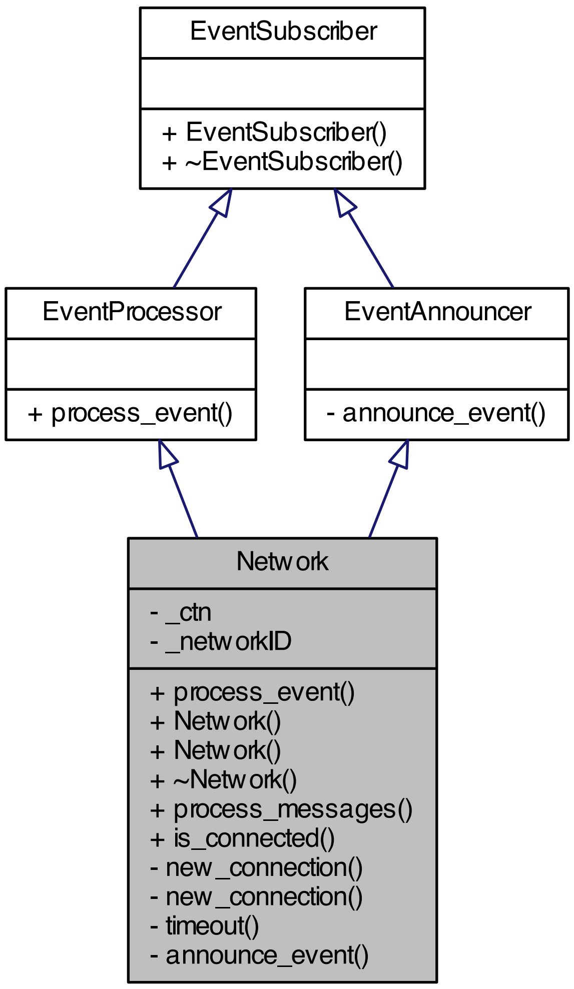
The documentation for this class was generated from the following file:

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**CatMouse.hpp**

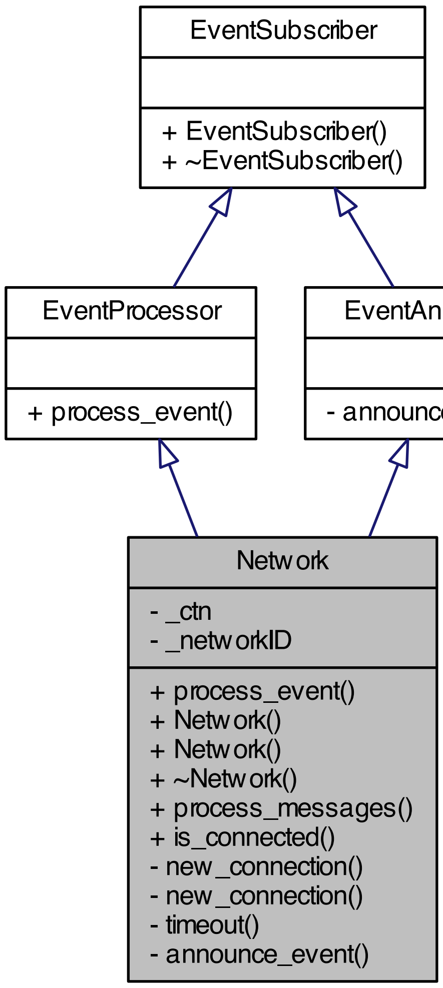
2.13 Network Class Reference

```
#include <Network.h>
```

Inheritance diagram for Network:



Collaboration diagram for Network:



Public Member Functions

- virtual void **process_event** (**Event** ∗eData)
- **Network** ()
- **Network** (string ipAddr)
- ∼**Network** ()
- void **process_messages** ()
- bool **is_connected** ()

Private Member Functions

- connection **new_connection** ()
- connection **new_connection** (string ipAddr)
- void **timeout** (string msgPrompt, string msgSucc, string msgFail, int timeoutSecs, function< void(void)> countdownBody, function< bool(void)> breakCondition)
- virtual void **announce_event** (string msg)

Private Attributes

- connection **_ctn**
- string **_networkID**

2.13.1 Detailed Description

Packages up data recieved from the controller and passes it to a given network.

Author

Alex Cummaudo

Date

7 Oct 2013

2.13.2 Constructor & Destructor Documentation

2.13.2.1 Network::Network ( )

Constructor initiates a connection as a host.

2.13.2.2 Network::Network (string *ipAddr*)

Constructor initiates a connection to a given ip address (as a client)

Parameters

ipAddr	The IP Address of the host this client will connect to

2.13.2.3 Network::∼Network ( )

Destructor closes all connections and announces Goodbye message.

2.13.3 Member Function Documentation

2.13.3.1 void Network::process_event (**Event** ∗ *eData*) `[virtual]`

Process the event data by packaging it and sending it over the network as a string (i.e. an outgoing event to a network string)

Parameters

eData	**Event** (p. 8) Data to process

Implements **EventProcessor** (p. 14).

2.13.3.2 void Network::process_messages ( )

Process messages that are being recieved (i.e. incoming network string to an outgoing event)

Note

Messages recieved in the format: key:value,key:value| etc. Hence we want to parse the msg back into its event kind

2.13.3.3 bool Network::is_connected ( )

Delcare is connected property

Returns

Boolean whether or not the network is connected

2.13.3.4 connection Network::new_connection ( ) [private]

Initiates the connection as a host, returning true or false on a success or error.

Returns

A new host connection to work with

Force break condition to be true

2.13.3.5 connection Network::new_connection (string *ip_addr*) [private]

Initiates the connection as a client, returning true or false on a success or error.

Parameters

ip_addr	IP Address that this client should connect to

Returns

A new client connection to work with

2.13.3.6 void Network::timeout (string *msgPrompt,* string *msgSucc,* string *msgFail,* int *timeoutSecs,* function< void(void)> *countdownBody,* function< bool(void)> *breakCondition*) [private]

The timeout connection; runs the passed function success on a success, and error function on error; allow passing of two functors so that lambda expressions can be passed in as parameters to the timeout.

Parameters

msgPrompt	Prompt message announced when timeout begins (i.e., why we're having a timeout).
msgSucc	Message announced when timeout did not run out and the break condition was met
msgFail	Message announced when timeout did ran out of timeoutSecs and the break condition was never met
timeoutSecs	How long to run timeout for
countdownBody	Function to run every second on timeout
breakCondition	Function that returns a bool to check whether or not the timeout should break

2.13.3.7 void Network::announce_event (string *msg*) [private], [virtual]

Sends an event to all subscribers with the given message.

Parameters

msg	Message to announce when creating an **Event** (p. 8)

Implements **EventAnnouncer** (p. 10).

2.13.4 Member Data Documentation

2.13.4.1 connection Network::_ctn [private]

Network (p. 28) connection controller between client and host.

2.13.4.2 string Network::_networkID [private]

Defines a unique address of this machine.

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

- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Network.h**
- /Users/Alex/Dropbox/Swinburne/HIT2302 - OOP/Projects/Cat and Mouse/#1_CatMouse_C++_DE-Coupled/src/**Network.cpp**