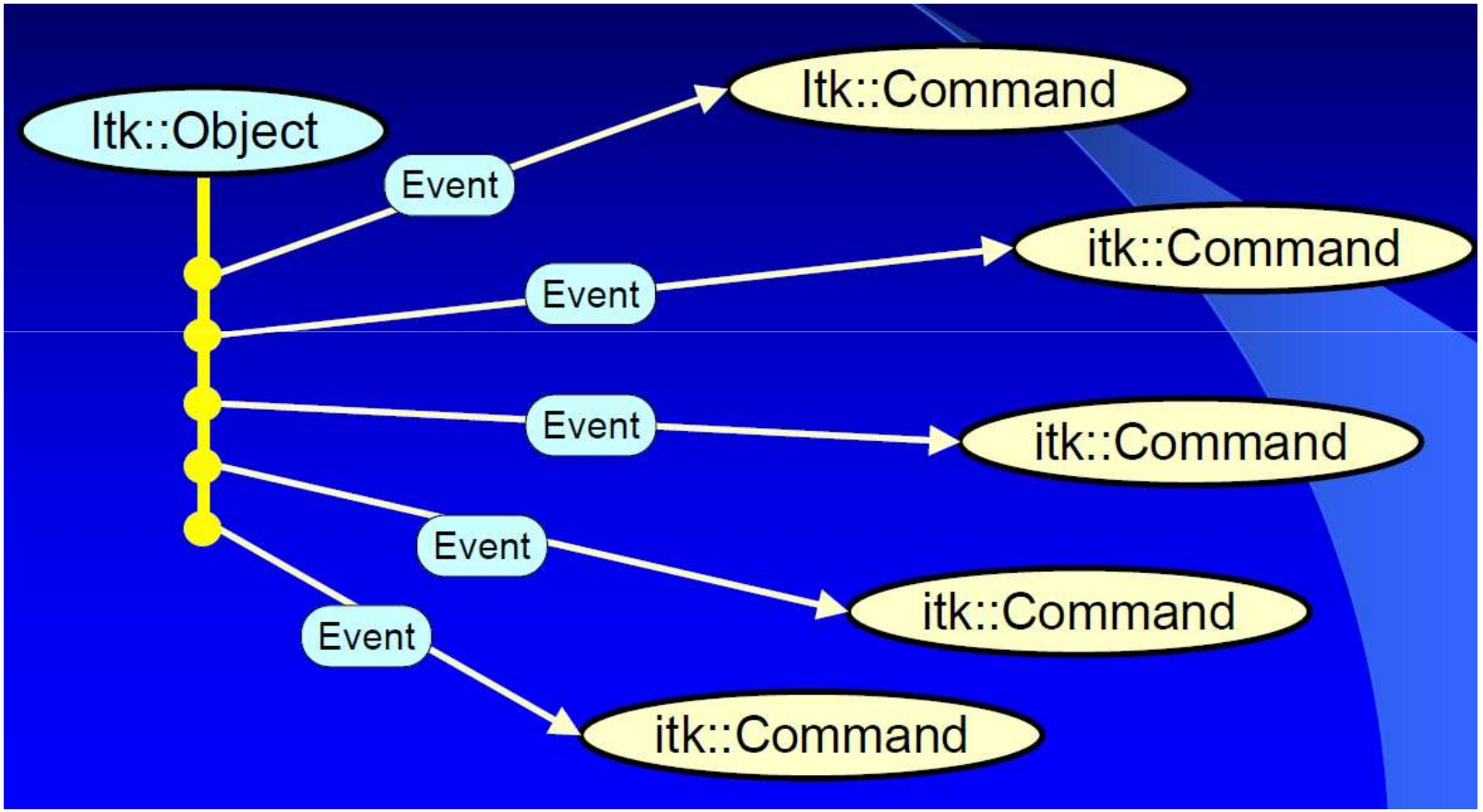
ITK Events

Michael Müller

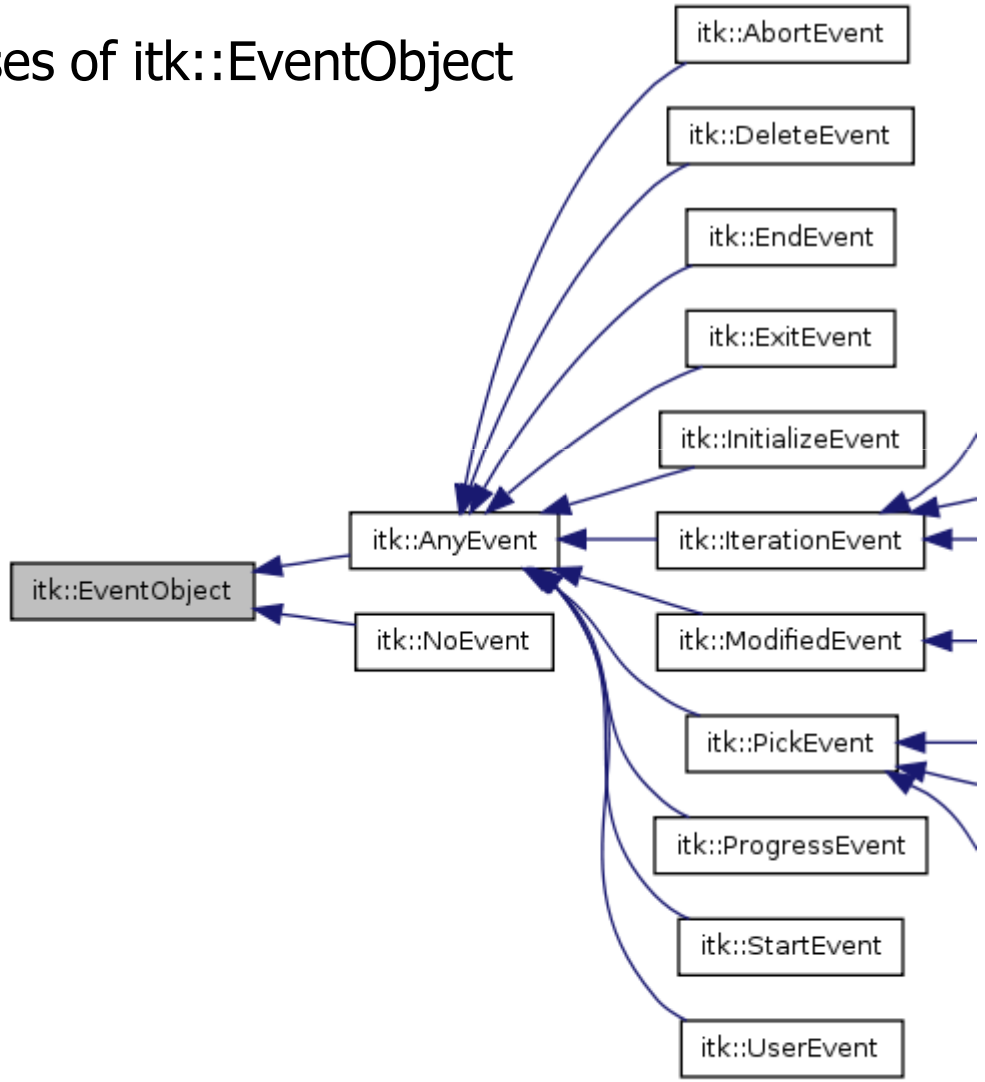
DEUTSCHES
KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

Overview


Itk::Object
Event
Itk::Command
Event
itk::Command
Event
itk::Command
Event
itk::Command
Event
itk::Command

Types of events

- Events are subclasses of itk::EventObject



Register and unregister event observer

```
class MyClass
{
public:
        itk::Object::Pointer m_itkObject;
        unsigned long m_ObserverTag;

        MyClass:: OnObjectModified(const itk::Object* caller, const itk::EventObject &event)
        {
          MITK_INFO << „Object was modified";
        }

        MyClass::AddObserver()
        {
           itk::MemberCommand<MyClass>::Pointer command = itk::MemberCommand<MyClass>::New();
           command->SetCallbackFunction(this, &MyClass::OnObjectModified);
           m_ObserverTag = m_itkObject->AddObserver(itk::ModifiedEvent(), command);
        }

        MyClass::RemoveObserver()
        {
          m_itkObject->RemoveObserver(m_ObserverTag );
        }
};
```

Register for AnyEvent

```
class MyClass
{
public:
        itk::Object::Pointer m_itkObject;
        unsigned long m_ObserverTag;

        MyClass:: OnEventOccured (const itk::Object* caller, const itk::EventObject &event))
        {
          try
          {
            dynamic_cast< itk::DeleteEvent& >( event  );
            MITK_INFO << „Object is about to be deleted“;
          }
          catch(…) {}
        }

        MyClass::AddObserver()
        {
           itk::MemberCommand<MyClass>::Pointer command = itk::MemberCommand<MyClass>::New();
           command->SetCallbackFunction(this, &MyClass::OnEventOccured );
           m_ObserverTag = m_itkObject->AddObserver(itk::AnyEvent(),  command);
        }
};
```

Invoke events

```
#include <itkProgressEvent.h>
#include <itkEndEvent.h>

class MyFilter: public itk::Object
{
public:
        MyFilter :: Execute()
        {
          while ( morePixelAvailable )
          {
             this->InvokeEvent( itk::ProgressEvent() );
             …
          }

          this->InvokeEvent( itk::EndEvent()  );
        }

};
```

The modified event

Use this->Modified() in your subclasses to …

1. mark your object as modified and

2. send out a modified event to observer

```
#include <itkObject.h>

class MyFilter: public itk::Object
{
public:
    void SetParameter( double param )
    {
      MITK_INFO << this->GetMTime(); // e.g. „1000“

      m_Param = param;
      this->Modified();

      MITK_INFO << this->GetMTime(); // „1001“
    }

    double m_Param;

};
```

Pitfalls

1. Do not unregister observer in your event processing method:

```
class MyClass
{
public:
…
        MyClass:: OnObjectModified(const itk::Object* caller, const itk::EventObject &event)
        {
          MITK_INFO << „Object was modified";
          m_itkObject->RemoveObserver(m_ObserverTag );
        }
};
```

2. Always unregister your observer to avoid null pointer access

```
class MyClass
{
public:
…
        MyClass:: SetObject( itk::Object* object )
        {
           this->RemoveObserver();
           m_itkObject = object;
           this->AddObserver();
        }
};
```