Code Conventions

Basic

Language:

- Everything in English! Punctuation!
  We are writing everything (variable names, documentation) in English, because the C++ syntax is also in English.

Header files

Header:

- Header files have the same name as cpp files. (.h and .cpp files)
  We are doing this, because we know they are related to each other.
- Header guards:
  #ifndef EXAMPLEHEADER_H
  #def EXAMPLEHEADER_H
  // code
  #endif // EXAMPLEHEADER_H

Order of includes:

- First we use the OGRE includes, then the system includes and the others at last.

Scoping

Namespaces:

- Always use the scope resolution operator "::" to access a namespace.
- Never use "using Namespace". This prevents class names from overlapping and thus removes the possibility of accidentally creating multiple objects with the same name (since we use multiple libraries).

Comments

Comments style:

- Comments above the designated code.
  We are doing this, so we know what C++ code the comment is describing.
- If there is only up to two lines use // for comments. If there are more lines use /* */.
  We are doing this, to keep the comments readable and because the first two signs will let you know whether it is a long comment or not.
- Use capitalisation. End the comment with a "." (dot).
  We are doing this, because we want to have good grammar and punctuation.
- Explain the function parameters before they get used.
  // @param x - x does something.
  // @param y - This does something else.
  void function(int x, int y);
  We are doing this so we can all understand what the parameters are doing.

TODO comment:

- The first word in the comments is “TODO”. This will indicate that there is still a part to do.
- After the TODO there is a short explanation of what is left to be done. This way everybody who reads the comments knows was is left to be done.

Formatting

Brackets:

- The brackets are displayed the following way:

```
Class
{
        private:
        public:
                void function();

        protected:
};


void Class::function()
{
}
```

- With multiple lines of code if will be displayed like this:

```
if(true)
{
        Do something fun.
}
else
{
        Do something.
}
```

Single line of code:

```
if(true){ break; }
```

```
if (foo)
{
        doSomething();
        while (bar.hasMoreElements())
        {
                doSomethingElse();
        }
}
```

We do it like this because we believe this provides code that is easier to be read by humans.

Naming

General naming rules:

- Names should have something in it that is related to what it does.

Capitalisation:

- First letter of variables/functions aren't capitalised. Second and subsequent words are capitalised. Class names are capitalized.

Private/Protected variables start with an "_"(underscore)

- The private and protected variables start with an "_" example:

```
Class SomeClass
{
      private:
            bool _active;
      public:
            void setActive(bool active){_active = active;};
            bool isActive(){return _active};
}
```

This way we can always immediately see whether a variable is public or not, without having to check the header file.