```
PISTS - Coding Style Gudie

Author: MIN, 14.3.02
Changes: MIN Prefinal Version, 16.4.02


Intro
  Das Projekt soll Veröffentlicht werden können (Opensource).
  Deshalb achten wir darauf, das es einigermassen den geltenden
  Standards im Netz entspricht. Unteranderem das der Code und
  Doku im Code (Komentare) für jederman lesbar sind.


1 Sprache

  1.1 Source
    Der Code ist Englisch. gibResultat() -> getResult()
    Ebenso Dateinamen (Konsistenz zum Code).

  1.2 Kommentare und Text in Sourcefiles
    Kommentare zum Code sowie Texte in Sourcefiles sind in
    Englisch Die wird auch für die I18n mit qt so benötigt.


2 Dateien
  Unter UNIX ist das Dateisystem Case Sentsitive. Daher müssen
  wir auch unter Windows die Konventionen beachten.

  2.1 Dateinamen
    Der erste Buchstabe wird generell klein geschrieben.
    Bei zusammengestzten Wörtern jeweils den ersten Buchstaben
    gross. Wir verwenden keine Umlaute und Leerschläge im
    Dateisystem ('ä'-> 'ae' etc, "my House.bsp" -> "myHouse.bsp").
    Dateinamen bei Codefiles auf Englisch, dadurch Konsitent mit
    Code und gar keine Umlaute nötig.

    Bspiele:  aStupidName
              UFO (reine Abkürzung)
              aUFOPilot (erster Bstb. klein)

  2.2 Dateierweiterungen

    Datei Typ      |   Erweiterung
    ---------------+-------------
    C++ Source     |   cpp
    C++ Header     |   h
    C Source       |   c
    C Header       |   h
    ACSI Text      |   txt

    (zu erweitern...)


3 Coding

  3.1 Tabulator
    Standard Tabulatorweite ist 2 Spaces. Tabulatoren werden
    nicht durch Spaces ersetzt. (Dadurch kann es jeder so
    Einstellen wie er es will).

  3.2 Blöcke
```



```
    3.2.1 if etc.

      if(true)
      {
        cout<<"hello";
        cout<<" world!"<<endl;
      }
      else
      {
        cout<<"bye";
        cout<<" world!"<<endl;
      }

  3.3 Klassen
    Nur eine Klasse pro Datei. Funktionsbeschreibung in
    Headerfile (Siehe Header Template). Bei der Implemetation
    zwischen zwei Methoden 3 Leerzeilen.

    Declaration (h):

    <HEADER>


    #ifndef POCKETBOMB_H
    #define POCKETBOMB_H


    #include<math.h>        // "system" includes
    #include<iostream.h>

    #include"bomb.h"      // own includes
    #include"Object.h"


    class PocketBomb : public Bomb, public Object {
    public:
      PocketBomb(void* Parent,
                 const int* Size= 10,
                 Coordinate position= Coordinate(0,0,0),
                 Object* dedicatedTo= 0);
      ~PocketBomb();

      void* getParent(void);  /* returns a pointer the current
          Operator of the device */
      void setParent(void* newParent);  /* Assigns a new parent */


      Coordinate getPosition(void); /* returns the actual position
          of the device. */
      void setPosition(Coordinate newPosition); /* assignes (moves) it
          to a new Position. */

      ...etc....

    private:
      Object* theTarget;  /* Keeps track of the Target */
      Coordinate  bombPosition; /* Keeps track of itself */
      int bombSize; /* stores the bombsize */
      void* blaParent;  /* points to the parent */
    };

    #endif
```

```
   <FOOTER>

   ----

   Implementation (cpp):

   <HEADER>

   #include<xy.h>        // "system" includes

   #include"ingniter.h"  // own includes


   PocketBomb::PocketBomb(void* Parent,
                const int* Size= 10,
                Coordinate position= Coordinate(0,0,0),
                Object* dedicatedTo= 0)
                : blaParent(Parent),
                  bombPosition(position),
                  theTarget(dedicatedTo)
   {

   ...some code...
   } /* PocketBomb::PocketBomb */



   PocketBomb::~PocketBomb()
   {
     this->explode();  // we dont let the bomb be defused.
   } /* PocketBomb::~PocketBomb */


   /*------------------------------------------------------------
   /      THE METHODES/CALLBACKS/WHAT EVER...
   /-----------------------------------------------------------*/


   void* PocketBomb::getParent(void)
   {
     return(blaParent);
   } /* PocketBomb::getParent */

   ...etc...

   <FOOTER>
```

3.4 Kommentare
Sollen die Verständlichkeit für den Programmierer fördern. Dazu gehört Erläuterung der Funktion (was tut das) und Beschreibung des zugrundeliegenden Gedankens (warum so und nicht anders).

3.5 Operatoren
Um Zuweisungen und Vergleiche besser unterschieden und schneller erkennen zu können gelten folgende Regelen:

Zuweisungen
Bei Zuweisungen wird rechts des Operators kein Leerzeichen eingefügt:


```
       int a= 0;
       bla+= 5;

   Vergleiche:
     Beiderseits des Operators ein Leerzeichen:

       if(a == b)
       while(n < i)
```

4 Header, Footer

4.1 Header
Enthält Informationen zu:
- Author, Copyright: Name, Mail
  (c) by John Random Hacker, j.r.hacker@geek.net
- Contributers/Coauthors
  defusing protection added by Henry B. Dynamite, hbd@xplode.net
- Erstellt: Datum dd.Month.YYYY
  3. Feb. 2002
- Modifiziert: Datum dd.Month.YYYY
  4. Mar. 2002
- Dateiname
  source.h
- Projektname und URL
  PISTS, http://pists.sourceforge.net/
- Beschreibung der Aufgabe/Zweck. 1 bis 3 Zeilen
  The class PocketBomb is intended to provide a simple solution for safe bomb handling.
- Lizenz (ie. GPL)