GENERAL

Use real tabs that equal 4 spaces.

Use typically trailing braces everywhere (if, else, functions, structures, typedefs, class definitions, etc.)

```
if ( x ) {
}
```

The else statement starts on the same line as the last closing brace.

```
if ( x ) {
} else {
}
```

Pad parenthesized expressions with spaces

```
if ( x ) {
}
```

Instead of

```
if (x) {
}
```

And

```
x = ( y * 0.5f );
```

Instead of

```
x = (y * 0.5f);
```

Use precision specification for floating point values unless there is an explicit need for a double.

```
float f = 0.5f;
```

Instead of

```
float f = 0.5;
```

And

```
float f = 1.0f;
```

Instead of

```
float f = 1.f;
```

Function names start with an upper case:

```
void Function( void );
```

In multi-word function names each word starts with an upper case:

```
void ThisFunctionDoesSomething( void );
```

The standard header for functions is:

```
/*
====================
FunctionName

  Description
====================
*/
```

Variable names start with a lower case character.

```
float x;
```

In multi-word variable names the first word starts with a lower case characte and each successive word starts with an upper case.

```
float maxDistanceFromPlane;
```

Typedef names use the same naming convention as variables, however they alway end with "_t".

```
typedef int fileHandle_t;
```

Struct names use the same naming convention as variables, however they always end with "_t".

```
struct renderEntity_t;
```

Enum names use the same naming convention as variables, however they always e with "_t". The enum constants use all upper case characters. Multiple words are separated with an underscore.

```
enum contact_t {
     CONTACT_NONE,
     CONTACT_EDGE,
     CONTACT_MODELVERTEX,
     CONTACT_TRMVERTEX
};
```

Names of recursive functions end with "_r"

```
void WalkBSP_r( int node );
```

Defined names use all upper case characters. Multiple words are separated wit an underscore.

```
#define SIDE_FRONT       0
```

Use ‘const’ as much as possible.

Use:

```
const int *p;            // pointer to const int
int * const p;           // const pointer to int
const int * const p;     // const pointer to const int
```

Don’t use:

```
int const *p;
```