**Code Style**

1. Prefix

|  |  |  |
| --- | --- | --- |
| Prefix | Purpose | Example |
| G | Global Variables | g\_counter |
| M | Member Variables | m\_counter |
| P | Point Variables | m\_pCounter |
| V | Virtual Functions | VDraw() |
| I | Interface Classes | IDrawable |

1. Every single if/else/for/while block should have its own braces.

**ex) 2-1**

bool m\_found;

if (m\_found)

{

bool = false;

}

else

{

bool = true;

}

**ex) 2-2**

int m\_int = 0;

if (m\_int <=0)

{

m\_found = false;

}

else

{

m\_found = true;

}

## Initialize your private variables.

|  |
| --- |
| **private int m\_myVar1,m\_myVar2,m\_myVar3;** |

This is not good.

|  |
| --- |
| **private int m\_myVar1 = 0;**  **private int m\_myVar2 = 0;**  **private int m\_myVar3 = 0;** |

This is better.

## Use whitespace and newlines.

## Each set of braces should ideally be on its own line. There are exceptions. But basic if/else blocks should look like the “This is good.” example in 1 above. Single lines of code don’t have to have whitespace around them, but complex blocks should. For example:

|  |
| --- |
| **if(condition)**  **{**  **Foo();**  **}**  **else**  **{**  **float myVariable = 0.0f;**  **for(int count = 0; count < SomeObject.Instance.numberOfThings; count++){**  **myVariable += Bar();**  **}**  **m\_someOtherObject.SomeCoolFunction(myVariable);**  **m\_someOtherObject.AnotherCoolFunction(myVariable);**  **BarTwo();**  **}** |

This example feels a bit crowded, it adds difficulty for other developers to parse.

|  |
| --- |
| **if(condition)**  **{**  **Foo();**  **}**  **else**  **{**  **float myVariable = 0.0f;**  **for(int count = 0; count < SomeObject.Instance.numberOfThings; count++)**  **{**  **myVariable += Bar();**  **}**  **m\_someOtherObject.SomeCoolFunction(myVariable);**  **m\_someOtherObject.AnotherCoolFunction(myVariable);**  **BarTwo();**  **}** |

Here we use whitespace to separate different "ideas" in the code, while still grouping actions that go together.

## Use understandable and sensible variable, class, and function names.

Don’t call something MyFunctionThatHandlesPlayerHealthUnlessTheyAreInGodMode(), but also don’t call it Handler(). If someone else can’t understand the high level purpose of something by its name alone, it should probably be renamed.

Our project follows the following patterns for naming:

|  |  |
| --- | --- |
| Member Variable | m\_variableName; |
| Local Variable | variableName; |
| Function | FunctionName(); |
| Constant | CONSTANT\_NAME = "value"; |
| Class | ClassName  {  } |
| Class File | ClassName.cs |

## 