# C++

During sprint 1 I mainly focused on creating a playable character and learning how ogre works. The result of this are the creation of the Shipcharacter and Character classes. These classes extend on the object and moveable object classes for some basic properties.

The shipcharacter class can be called upon to create a playable ship character in the DrawScene method in the TutorialApplication.cpp.

The shipcharacter constructor looks like the following:

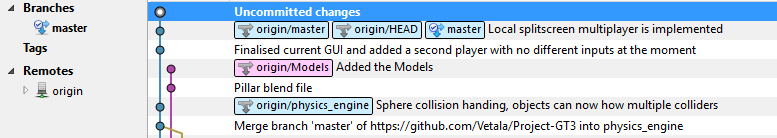
# http://puu.sh/nOmrO/8c5ad8c470.png

The constructor asks for a unique name, the scenemanager, the name of the shipmodel, a health amount, a starting position offset and a camera object which will follow the spaceship when it travels around.

# Tooling

For GIT management I use the Sourcetree plugin. This allows me to easily check everything happening in the GIT repository and also helps me resolve possible merge conflicts by showing me the differences in the Sourcetree program.

We create a branch for every new item that is on the scrum board when we start working on the item. The branch will be merged with the main branch when the code has been checked by a different team member and has been proven to work. The following is a screenshot from Sourcetree.



To easily import Ogre Meshes we create a .blend model in blender and drag this onto the OgreMeshy program which instantly converts the .blend model to a .mesh file which can be imported in Ogre.

We also use the Doxygen tool for automated documentation which can be found at the following link:

<https://oege.ie.hva.nl/~dikmanb001/>

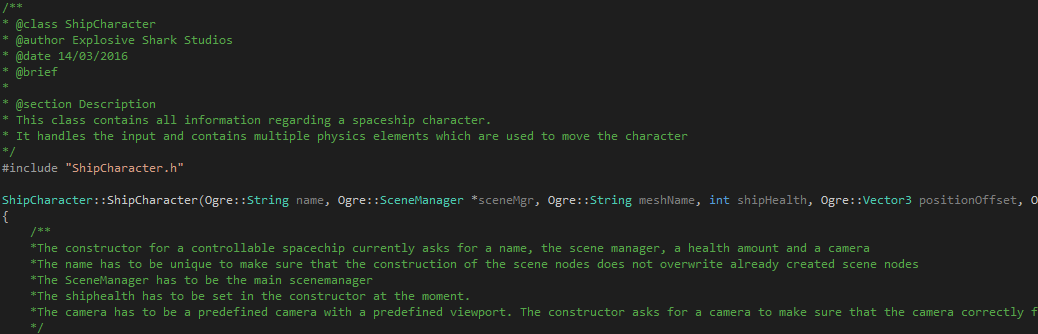
# Coding Standards

Initially we had set up a coding standard for our group which contained the method of how we'd write classes, functions and methods.

Including things such as having { and } on new lines,

In addition this coding standard strongly encouraged writing comments explaining all parts of a function right above and only adding any comment information inside when it was a very extensive or difficult function.

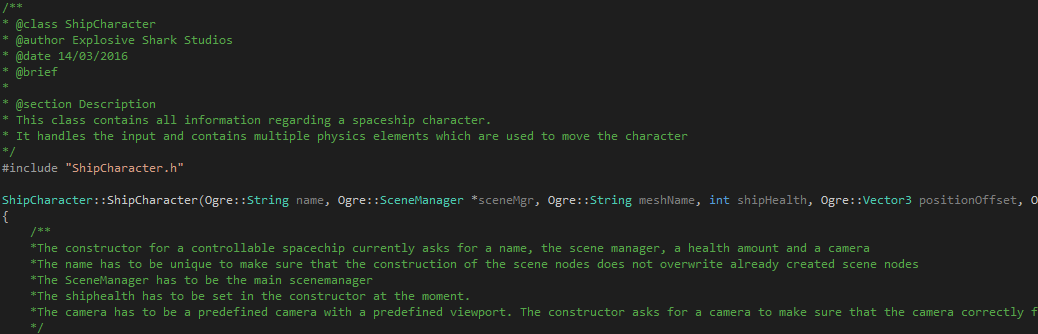
With the setup of our online documenting tool Doxygen, the standard changed. After the tool was implemented, classes needed to add information at the start which can be seen in the following screenshot.

Because these coding standards contribute to the usage of an automatically documented tool it's essential they be actively used. This motivation would qualify this chapter of Coding Standards to the Advanced 10 points

# Documentation

As mentioned earlier in this report we currently make use of an online documentation tool called Doxygen.

This tool takes comments like the following and converts them into an html page. It also creates class diagrams.



The online documentation can be seen at the following link:

<https://oege.ie.hva.nl/~dikmanb001/>

# Refactoring