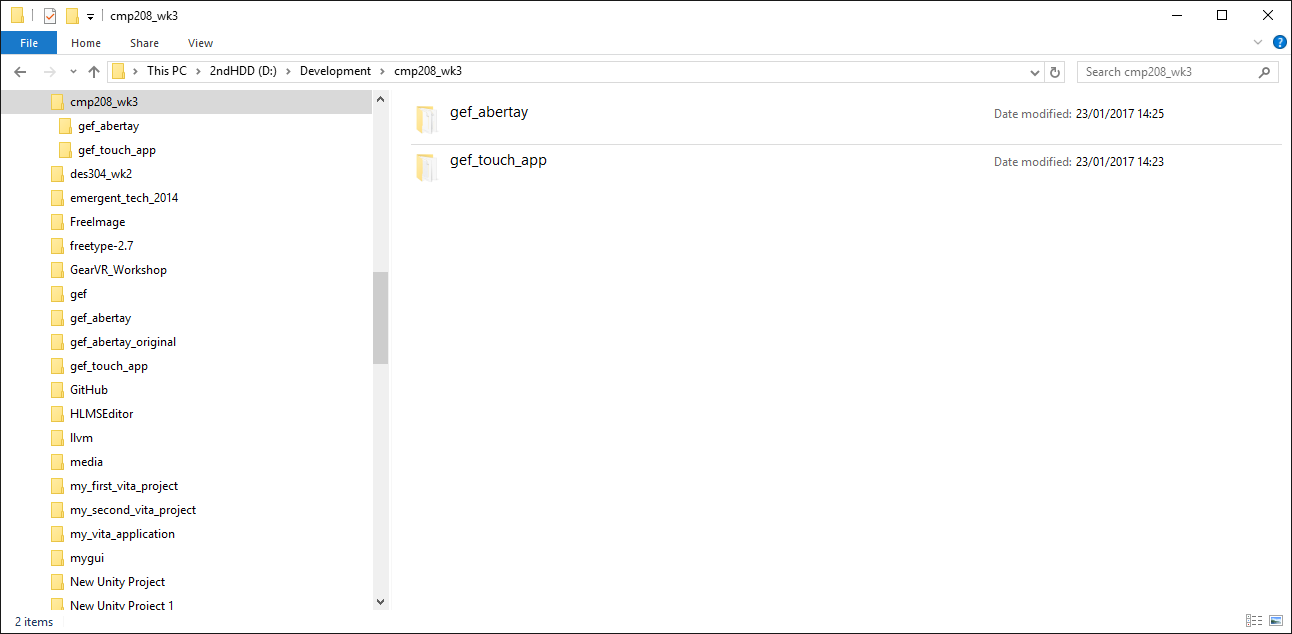
# Version Control and Touch Input

## Version Control

1. Download gef\_touch\_app.zip from blackboard and extract it to a folder (e.g. gef\_touch\_app) in the same folder that that the gef\_abertay framework has been extracted to.



1. Open up a Git Bash from the Windows Start Menu and change to the gef\_touch\_app folder using the cd command.

cd – change directory

e.g.

cd /m/cmp208\_wk3/gef\_touch\_app

1. Look at the tutorial below to carry out the following stages to create your repository.

<http://kbroman.org/github_tutorial/pages/init.html>

* Initialise the repository.
* Add all the files in the gef\_touch\_app folder to the staging area.
* Commit all the files in the staging area to the local repository. Don’t forget to set the commit message.

1. The next steps you will need a free github account from github.com.
2. Create a new repository (e.g., cmp208\_wk3) there and copy the repository URL to the clipboard.

It will look something like this:

<https://github.com/grantclarke-abertay/cmp208_wk3.git>

1. Go back to the Git Bash window and create a remote for the repository using the command

git remote add origin <https://github.com/grantclarke-abertay/cmp208_wk3.git>

1. Push the changes made to the local repository to the remote repository using the command

git push -u origin master

Check that the files have been added to the remote repository by logging into you github account and looking at your repository.

1. Try cloning your remote repository to a completely new folder.
2. Read other git tutorials and the help reference to make sure you understand how the git commands we have covered today work.
3. Try using the Git GUI to do the same kind of operations.
4. Experiment a lot with making changes to your repository before you put any valuable work in it.

## Touch Input

1. Add a sprite to your application. [Refer to previous weeks if you’re not sure how to do this]
2. Using the type variable of the Touch object, make the sprite perform the following actions:

* Move to the position of a NEW touch.
* Change colour to red while the touch is ACTIVE, but do not update its position.
* When the touch is RELEASED, change the colour to green and move the sprite gradually to the position of the touch when it was released.

Hint: You’ll need to calculate a vector between the touch start position and end position and then divide that vector into a smaller vector that can be used to update the position each frame.

e.g.

Start position = (0, 0)

End position = (120, 0)

Vector between points is therefore (120, 0)

If we wanted the movement to happen over 60 frames then you divide the vector by 60 giving a vector of (2,0)

Add this vector to the position of the game object for 60 frames and it’ll reach the final destination.

1. Write a function with the following prototype:

bool IsInside(const gef::Sprite& sprite, const gef::Vector2& point);

Write the body of the function to test if the coordinates of the point passed in are within the sprite boundaries. Remember the position of the sprite specifies the centre of the sprite and the width and height will give you the size.

It might be useful to visualise the sprite by drawing it on paper to help calculate the position of each of the sides.

Change your application so the sprite follows your touch, but only if you are actually touching the sprite.