

# OXO Continues

COMSM0086

Dr Simon Lock & Dr Sion Hannuna

# Before We Begin

We are almost halfway through the teaching block  
Useful to get your feedback on how things are going

Could you please visit the "Blue" website:

<https://evaluation.bristol.ac.uk/home/>

And complete the questionnaire for this unit

Simon's Monitoring Link:

# Extra OXO Features

This week we will continue with the OXO exercise

We will add in some error handling mechanisms

Also add extensions to make game more interesting

Introduce a new process to our development work

Let's look at each of these in turn...

# Error Handling

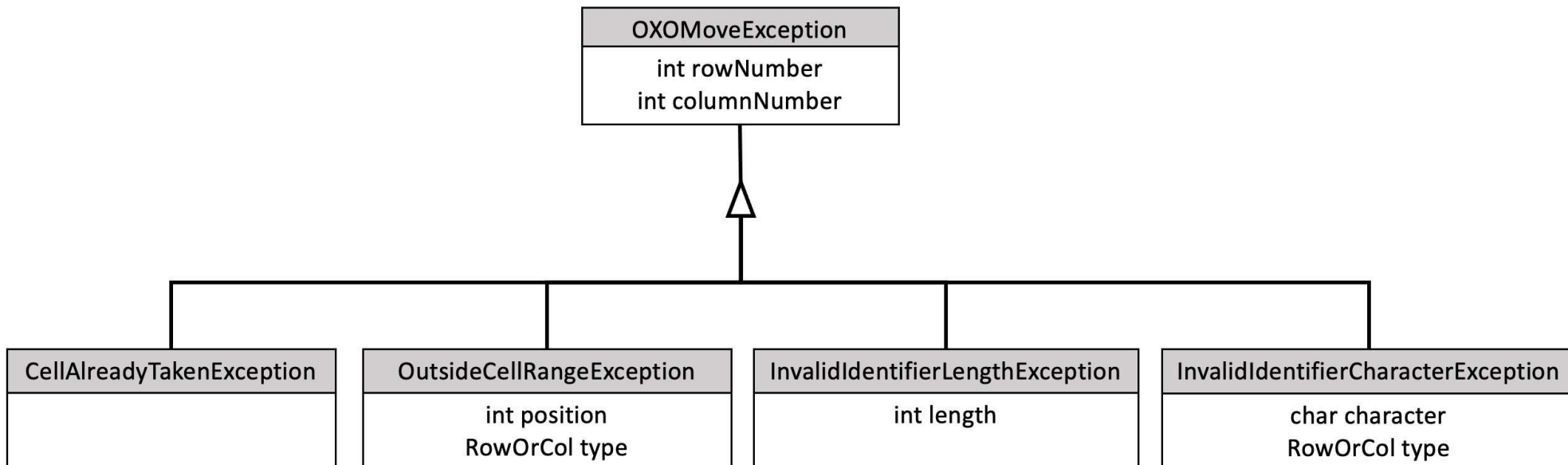
It's likely users will make mistakes during gameplay  
Entering 'invalid' cell identifiers into the GUI:

- Invalid Identifier Length: Command is not 2 chars
- Invalid Identifier Character: Row character is not a letter or column character not a numerical digit
- Outside Range: Valid characters, but identifier values are out of range (i.e. too big or too small)
- Already Taken: Cell has previously been claimed

In Java we handle run-time errors using 'Exceptions'

# Exceptions Hierarchy

We've provided you with a hierarchy of exceptions  
Commonalities are factored out into the superclass  
See workbook for examples of how to use these



# Adjustable Win Threshold

Win Threshold is the number of cells required to win  
More interesting if we can alter this threshold !

OXOGame allows users to set the win threshold  
Altered by pressing the `+` and `-` keys  
(actually the `=` and `-` keys for convenience)

Controller is then notified through two methods:

`increaseWinThreshold()`    `decreaseWinThreshold()`

You should update threshold in the model state  
Then use this value when performing win detection

# Greater Number of Players

Traditional number of players in an OXO game is 2  
Additional players makes game more interesting !

Add features to support any number of players  
(data structures, turn taking, win detection etc.)

Number of players can't be changed using GUI  
This can only be done programatically  
(A good opportunity for automated testing !)

# Source Code Management

Focus of unit is on broader 'Software Development'  
As part of this, we CARE about source management

So much so, that it is part of assessed exercises !  
It would be a good idea to start practicing this now  
(get used to the process before it really counts !)



# Links with other units

In OTHER units you learn the THEORY of using Git  
In THIS unit you get the chance to use it for REAL

Git is of MOST use in collaborative group projects  
(will be invaluable for group projects over summer)

That said...

Git still offers benefits to individual developers  
(As we shall see in the rest of this unit !)

# DON'T PANIC

Git is a complex beast...  
...but there is no need to use all of it !

clone / add / commit / push / pull

Is all you are really going to need

# Creating Repositories

I find the easiest way to create a repo

Is to do it via the GitHub website...

Click on the green 'new' button

Name the repo JAVA-CW-2023

Make it private

Add a README file

Then get a local copy using:

```
git clone <url>
```

# Adding Projects Resources

The next step is to drag in the maven project  
Each project should be kept inside it's own folder  
Add the whole folder so that it is tracked by git:

```
git add cw-oxo
```

Make sure you also ADD any new files you create !

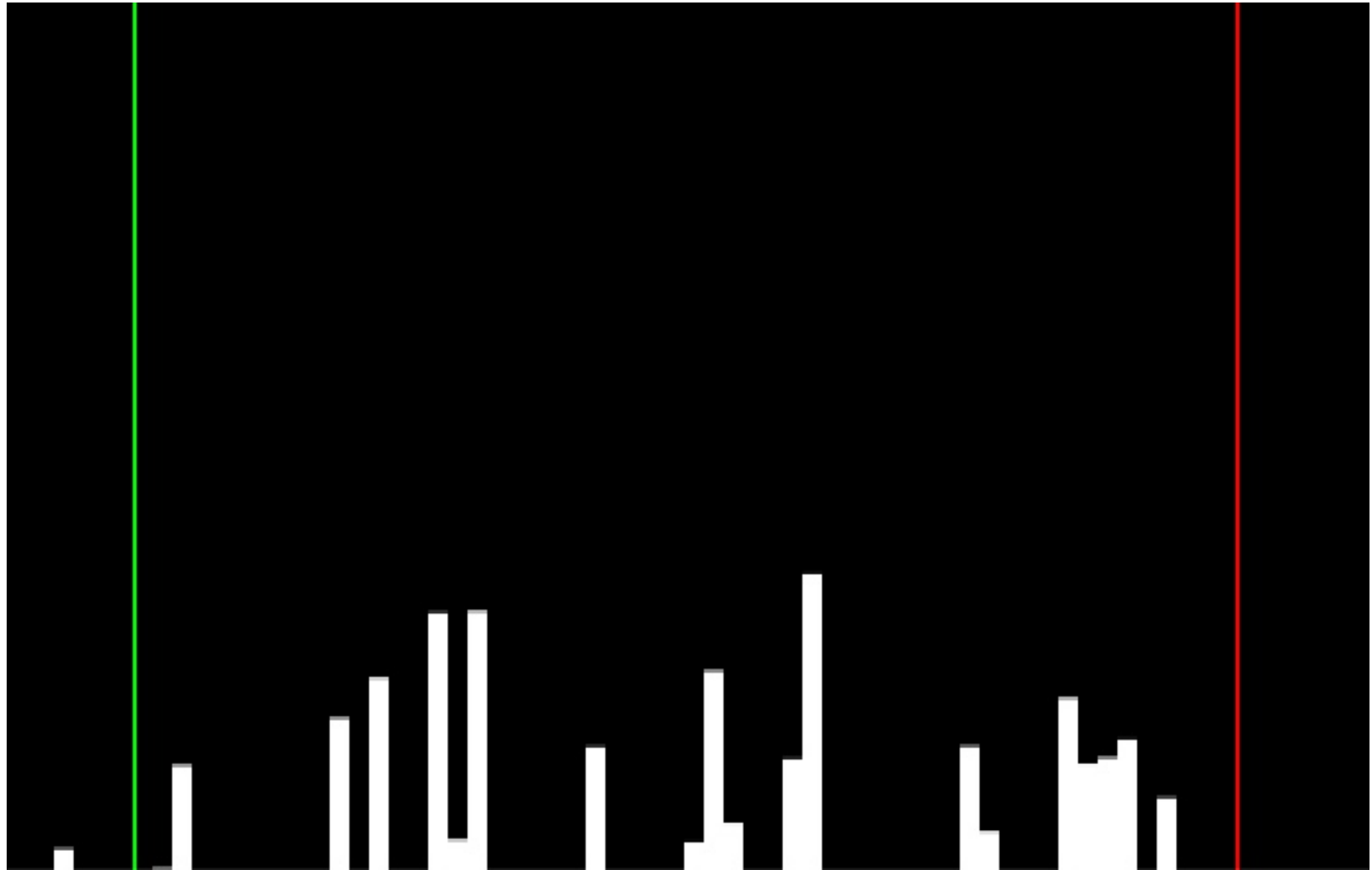
It is important that you only track "suitable" files  
(more guidance is provided later in this session)

# Committing and Pushing

You should COMMIT on regular basis  
(every time you feel a sense of achievement)  
e.g. when you fix a bug or implement a feature  
Makes it easier to write short & simple comments !

Be sure you PUSH your code to GitHub regularly  
Once per day should be a bare minimum  
At the end of each coding "session" is sensible  
"Before you eat or sleep" is a good principle !

# What your commit history should look like



# What you should definitely avoid



Fix bug on login screen

Add graph to dashboard



# Demonstration

Use a GUI client if you want, but...  
It is often simpler to just use the command line

Terminal  
ShowRepoFolder

```
nano app.js  
git commit -am "Update goodbye message"  
git push  
nano some-new-file.java  
git commit -am "Try a test"  
git add some-new-file.java
```

<https://github.com/drslock/test/tree/master>

# Be careful what you push to GitHub

Keep all your files in the correct places...

Source code goes in "src/main" tests in "src/test"

Don't commit `*built*` or `*generated*` files

There is no need: they can be re-generated again !

For example `.class` files and the `.idea` folder

Make use of `.gitignore` files to prevent upload...

# .gitignore

\*.class

.idea

.DS\_Store

passwords.txt

.gitignore file should be in project root folder  
Don't expect to see it - the dot makes it hidden !

# Trust GitHub to do the job

Whole point of GitHub is it maintains a full history  
It handles all versioning for you - and does it well !  
You don't need hacky ad-hoc version management:

- There should be NO duplicated files in your repo !
- Don't push backups or zip archives to your repo
- Don't create weekly snapshots in your repo structure
- Don't maintain different variants of the same file

Commit history and branches do these all for you !

Things that I have seen...

▼ Weekly Briefings

Week-01.pdf

Week-02.pdf

Week-03.pdf

▼ Weekly Workbooks

> 01 Introduction

> 02 Language and Libraries

> 03 Using Your Own Computer

> 04 Using Your Own Computer

> 05 Using the Lab Machines

> 06 The Template Project

> 07 Understanding the Template

> 08 GitHub Repository

▼ extras

▼ RedNoise

▼ RedNoise

> .vscode

> Background

▼ build

CanvasPoint.o

CanvasTriangle.o

▼ libs

> glm-0.9.7.2

> sdw

> src

58.ppm

Makefile

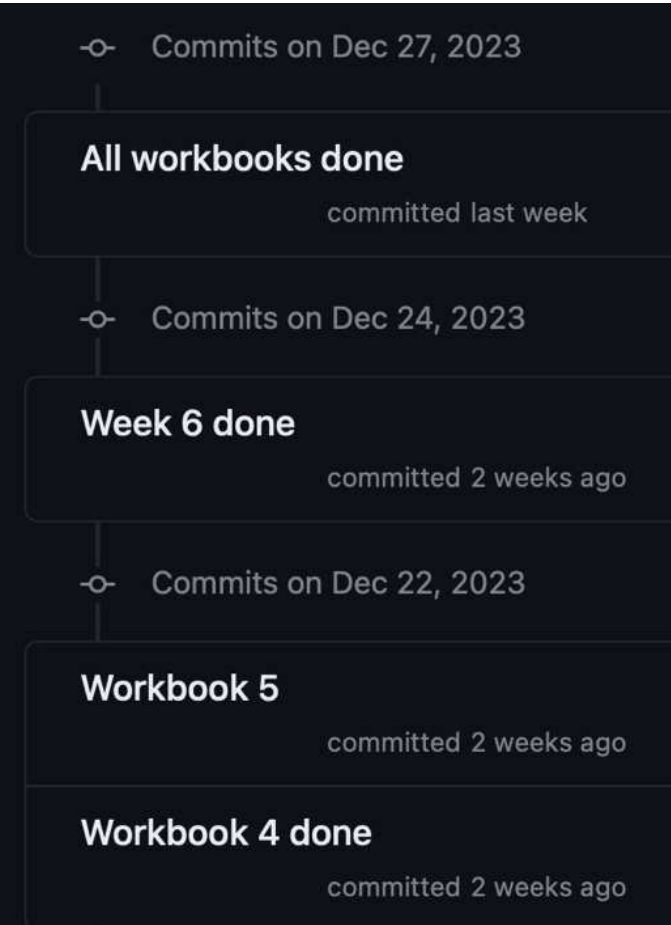
README.md

W4T7.obj

a.cpp

# This is just one 7 week project

1938ed0 · last week 🕒 13 Commits		
📁 Week 1/RedNoise	Small changes. T...	3 months ago
📁 Week 3/TraingleLi...	Obj and mtl file p...	2 months ago
📁 Week 4/threeD	Workbook 4 done	2 weeks ago
📁 Week 5/camTrans	Week 6 done	2 weeks ago
📁 Week 6/rayTrace	Week 6 done	2 weeks ago
📁 Week 7/lighting	All workbooks do...	last week
📄 README.md	Update README....	4 months ago



# Branches

You are welcome to use branches if you wish...  
But you don't have to for this particular unit

They are primarily intended for parallel collaboration  
But can be useful for individual experimentation  
Create branch if you're not sure if the code will work

They shouldn't exist on their own for too long  
Merge with master if/when you get features working

Branches are a bit like...



# Branch Metaphor

A bit like an operating theatre in a hospital

A place to open up your code & tinker with internals

NOT a place to keep the patient in the long term

You'll want to stitch them up as soon as possible

Then return them to the main ward (master branch)

In collaborative development...

there is another department

waiting to work on the patient

# Privacy of Repositories

It's ESSENTIAL you keep your repository private  
If you make it public, others will be able to read it  
Which is equivalent to publishing it online !  
Effectively this would be encouraging plagiarism

Bottom line: Keep your repository private  
(we'll ask you to invite us so we can see inside)

During the coursework assignments  
You will be assessed not only on functionality  
But also on the evolution of your repo codebase

Might be an idea to get practicing now !