COMS30020 - Computer Graphics

Week 2 Briefing

Dr Simon Lock

Where are we ?

Hope that you all got the RedNoise project working ?
If not, be sure to do so as quickly as possible !

Aim of this session: to explore this week's workbook
Practical on Friday, but do start the workbook now !

In this workbook we will actually do some drawing !
(but only very basic stuff - in 2D)

Later we'll be doing more sophisticated stuff in 3D
But we must learn to walk before we can run

This Week's Workbook Tasks

Interpolation a key technique in Computer Graphics

`"Evenly filling-in the gaps between known values"`

We will need to apply it to a wide range of entities:

`Lines, Colours, Positions, Textures, Shading, Shadows`

We first start with simple floating point values:

`1.1   ?   ?   ?   5.5`

We then move on to 3-element numerical values:

`[1,5,0]  [?,?,?]  [?,?,?]  [?,?,?]  [5,1,8]`

So we can interpolate RGB colours, 3D positions etc

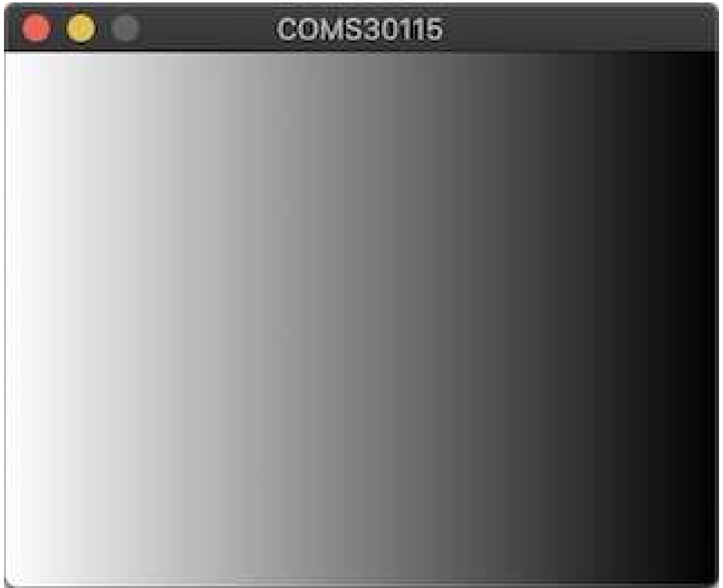
Linear Grayscale Interpolation

This is our very first GRAPHICAL task !

Draw a grayscale linear gradient

Just in one dimension (from left to right)

Repeat on each row (because it's only 1D)



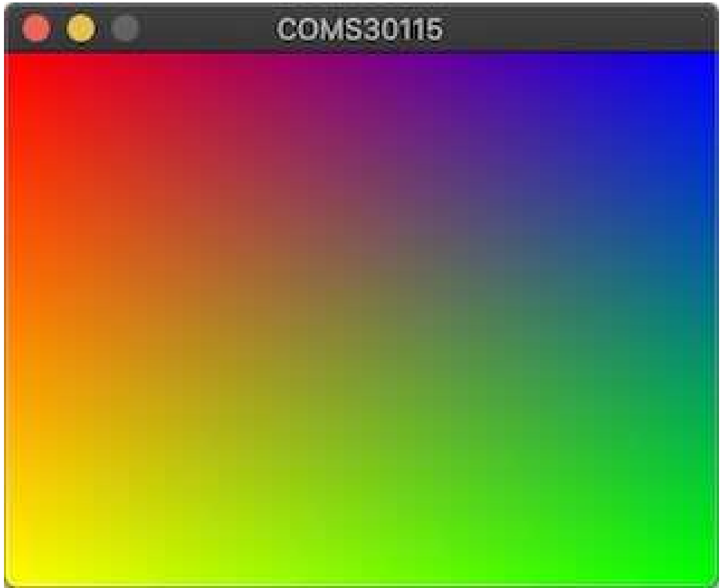
Rectangular Colour Interpolation

Next stage is a 2D colour interpolation

Start with "pure" colours in the four corners

Interpolate down to calculate the first & last columns

Then interpolate across to calculate pixels on each row



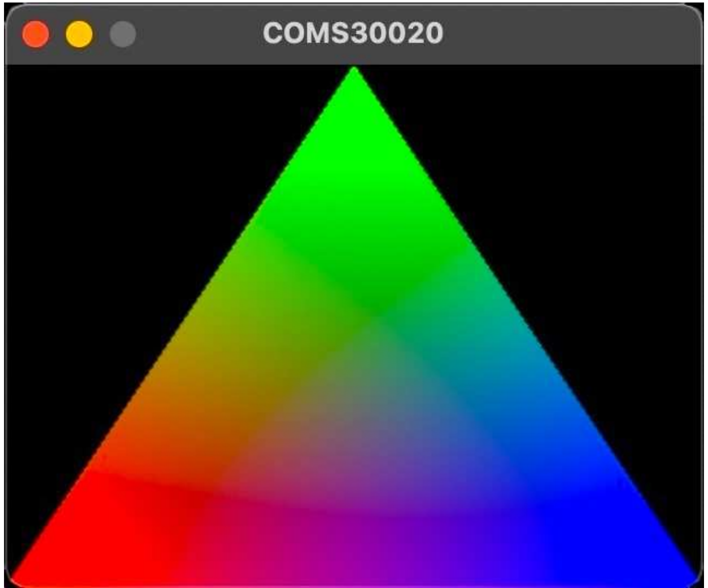
Triangular Colour Interpolation

If we want primary colour corners, we need a triangle

With a rectangular shape, the sides were perpendicular

A triangle has 3 non-perpendicular sides, so requires...

BarycentricWeighting



Warning !

Your colour spectrum will look a little bit "murky"
This is because it's not actually a full colour spectrum

As you move away from a particular vertex...
the value of that primary colour's channel will drop

You'll never get [255,255,0] [255,255,255] [0,0,0]
SUM of ALL channels will be always be exactly 255

`This is neither additive nor subtractive colour mixing`

What's your Vector Victor ?

This week we'll encounter a slightly annoying issue
We will be using 2 different things called "vector":

- "vector" (from std) used as a dynamic list structure
- "vec3" (from glm) used to store 3 element entities

Need to be careful - think about what you are doing
Ensure you use right structure for a given situation

In reality these are BOTH inappropriate names :o(
We don't get to choose names, so what can you do !

Prevalence of AI tools

Let's spend a bit of time talking about the use of AI

AI tools to support development are everywhere:

- Plugins embedded within IDEs
- Services integrated into code repositories
- "Freely" accessible web-based tools
  (although many require a paid subscription !)

The lure to use AI in development is hard to resist !

Acceptance of AI tools

Rather than trying to ban the use of AI on this unit
(which would probably be impossible to enforce)

A more pragmatic approach is to permit such tools

In many ways, use of AI support tools is a key skill
Chances are you'll be using them when you graduate

Note that this isn't a "learning how to code" unit
So *careful* use of AI shouldn't conflict with ILOs

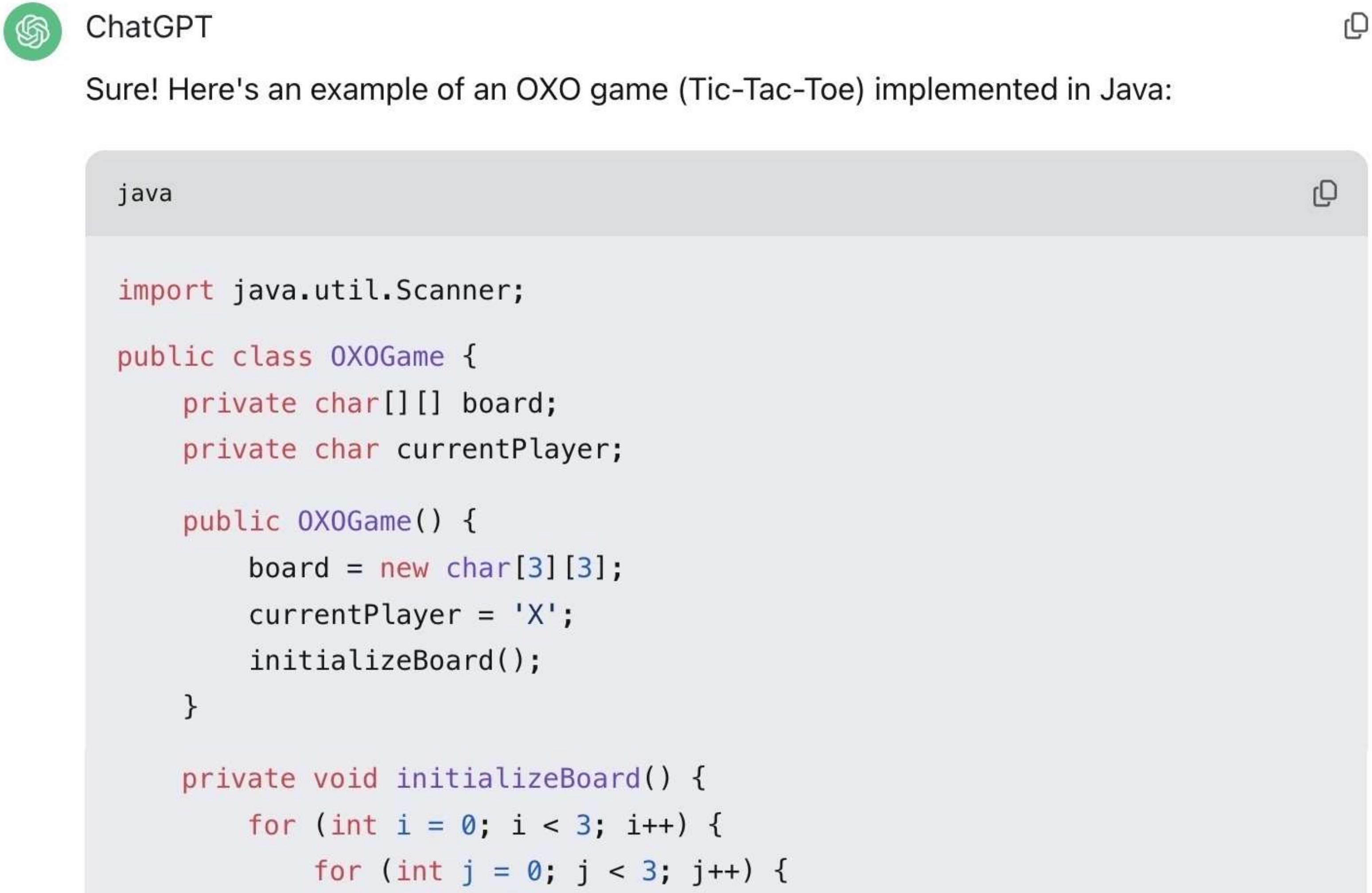
"Generative" AI

There is a reason why I put "Generative" in quotes
It's not really generating new material, but rather...
just recycling/reusing/recombining existing work
More like a "mashup" than creating something new

Just consider the recent upset of Artists with AI
Tools "trained" on their work, used to "synthesise"

Highlighted by legal action by publishing companies:
https://www.bbc.co.uk/news/articles/ckrrr8yelzvo

"Write me an OXO game in Java"



ChatGPT

Sure! Here's an example of an OXO game (Tic-Tac-Toe) implemented in Java:

```java
import java.util.Scanner;

public class OXOGame {
    private char[][] board;
    private char currentPlayer;

    public OXOGame() {
        board = new char[3][3];
        currentPlayer = 'X';
        initializeBoard();
    }

    private void initializeBoard() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
```

Mashup Coursework Submissions

The notion of a "mashup" codebase is nothing new
It does not rely on availability of effective AI tools
Students have been creating mashup work for years
`borrow a bit of code from here, borrow a bit from there`

Modern AI tools just make process quicker & easier
Students often don't even realise they are doing it !
But that's what you get when you Gen AI some code

Problem

How can we deal fairly with all following situations:

- Students who hand-craft ALL code themselves
- Students who "borrow" online code fragments
- Students who generate whole functions using AI

Typically people borrow/use AI when they get stuck

`I couldn't get my parser to work, so copied some code`

How can we let people do this (in order to progress)
But differentiate them from students who "did it all"

Concept Solution

If we could detect code that has been "derived"
We could factor it out during the marking process
(you don't get marks for code that you didn't write)

This would allow us to give credit where credit's due
But allow people to fairly use AI when they need to

This would provide flexibility in development process
As well as encouraging people to think critically...
about material produced using "generative" AI tools

Implementing this Approach

We have a library of 300 solutions to the assignment
ALL submissions of ALL students from previous years

Many are available online (as part of code portfolios)
These will be part of the training set used by AI tools

These 300 solutions also borrow from online examples
(which provides us with an even broader sampling)

All we do is compare YOUR code with this training set
The more similarities that are found...
The more "derivative" your submission has become

I know what you are thinking...

There is BOUND to be some level of similarly
Some standard boilerplate code that everyone has
Or some convergent algorithm that everyone uses

You'd be surprised how much variability there is !

We will be using a variety of filters and thresholds
The aim being to factor out "natural" commonality

Also human-in-the-loop to check for false-positives

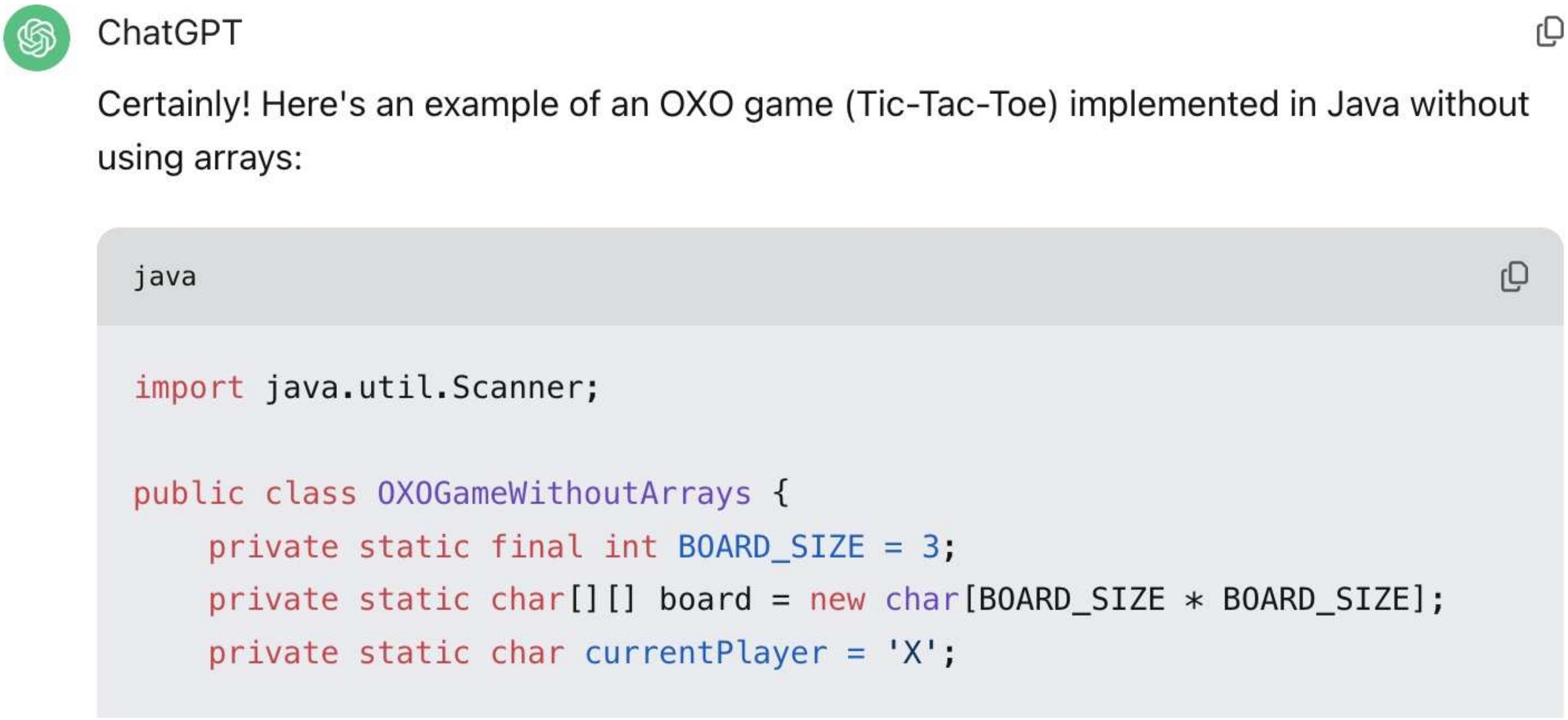
The Bottom Line

You *may* use AI tools to support your development
However, avoid extensive use of AI code "generation"

AI tools should be there to ASSIST you in development
NOT to do ALL the work for you !

Be critical/analytical/reflective about the output of AI
It is not always correct, appropriate and useful

"rewrite OXO so that it doesn't use arrays"



Certainly! Here's an example of an OXO game (Tic-Tac-Toe) implemented in Java without using arrays:

```java
import java.util.Scanner;

public class OXOGameWithoutArrays {
    private static final int BOARD_SIZE = 3;
    private static char[][] board = new char[BOARD_SIZE * BOARD_SIZE];
    private static char currentPlayer = 'X';
```

The SAFE use of AI

AI tools *can* provide some really good prompts

IDE code completion tools can often be very helpful
(avoid following advice blindly - try to understand !)

Chat AI tools can be useful for understanding errors
Paste obscure compiler error into chat for insight

Ask AI what is wrong with specific lines of code
But avoid dropping in large fragments of code...
Remember: these AIs tools are always learning !

Some important questions

Who "owns" the code that you feed into AI tools ?

Have you given the AI permission to use that code ?

Will the AI keep a copy of that code somewhere ?

Will AI "generate" fragments of your code for others ?

Did you check the terms and conditions of the tools ?

REMEMBER

You will only be rewarded for code that YOU write

Any "derived" code ("found" items or IA generated)

Will be discounted during the marking process

(resulting in a reduced final mark)

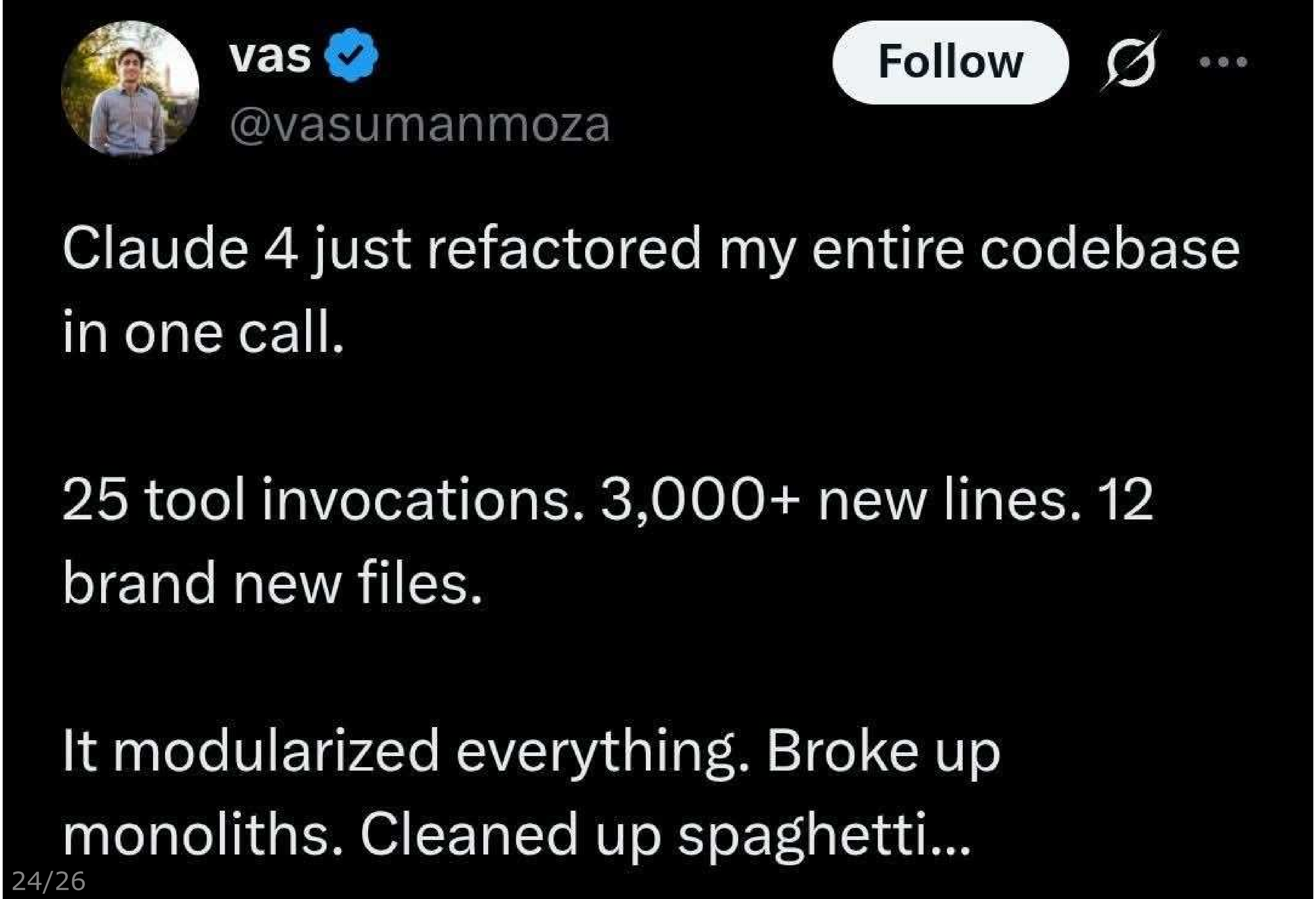
Why are we telling you this NOW ?

COURSEWORK UNIT

You will need the codebase you write in weeks 1-8
As a foundation to complete coursework assignment
So the "derivation" rules apply for the entire unit !

EXAMINATION UNIT

If you are taking the exam variant, same rules apply
You need to experiment hands-on with practical tasks
If you just generate your code, you won't learn much



vas

@vasumanmoza

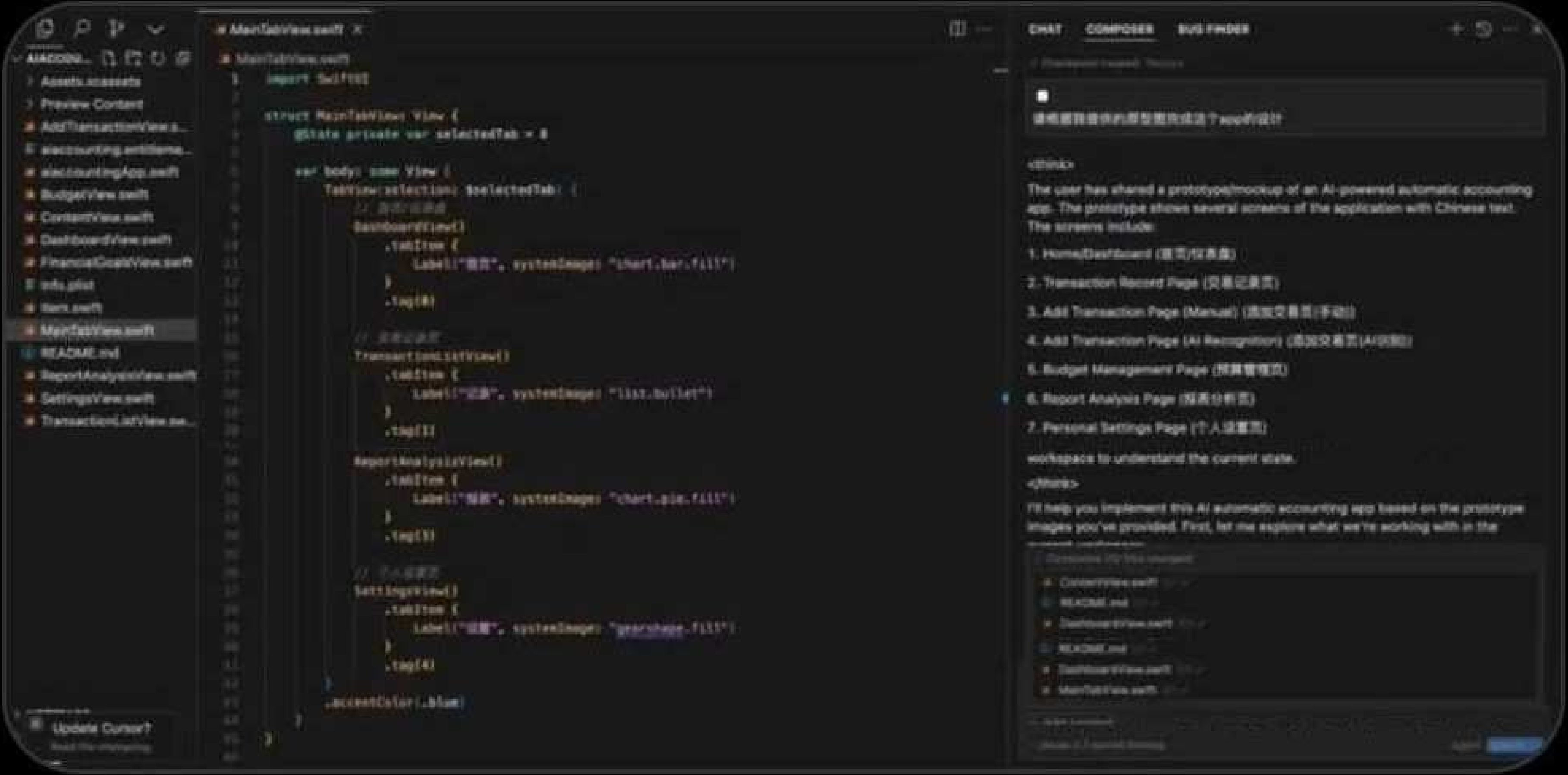
Claude 4 just refactored my entire codebase in one call.

25 tool invocations. 3,000+ new lines. 12 brand new files.

It modularized everything. Broke up monoliths. Cleaned up spaghetti...

None of it worked.

But boy was it beautiful.



```
import SwiftUI

struct MainTabView: View {
    @State private var selectedTab = 0

    var body: some View {
        TabView(selection: $selectedTab) {
            // [illegible]
            DashboardView()
                .tabItem {
                    Label("[illegible]", systemImage: "chart.bar.fill")
                }
                .tag(0)

            // [illegible]
            TransactionListView()
                .tabItem {
                    Label("[illegible]", systemImage: "list.bullet")
                }
                .tag(1)

            ReportAnalysisView()
                .tabItem {
                    Label("[illegible]", systemImage: "chart.pie.fill")
                }
                .tag(3)

            // [illegible]
            SettingsView()
                .tabItem {
                    Label("[illegible]", systemImage: "gearshape.fill")
                }
                .tag(4)
        }
        .accentColor(.blue)
    }
}
```

[illegible]

<think>

The user has shared a prototype/mockup of an AI-powered automatic accounting app. The prototype shows several screens of the application with Chinese text. The screens include:

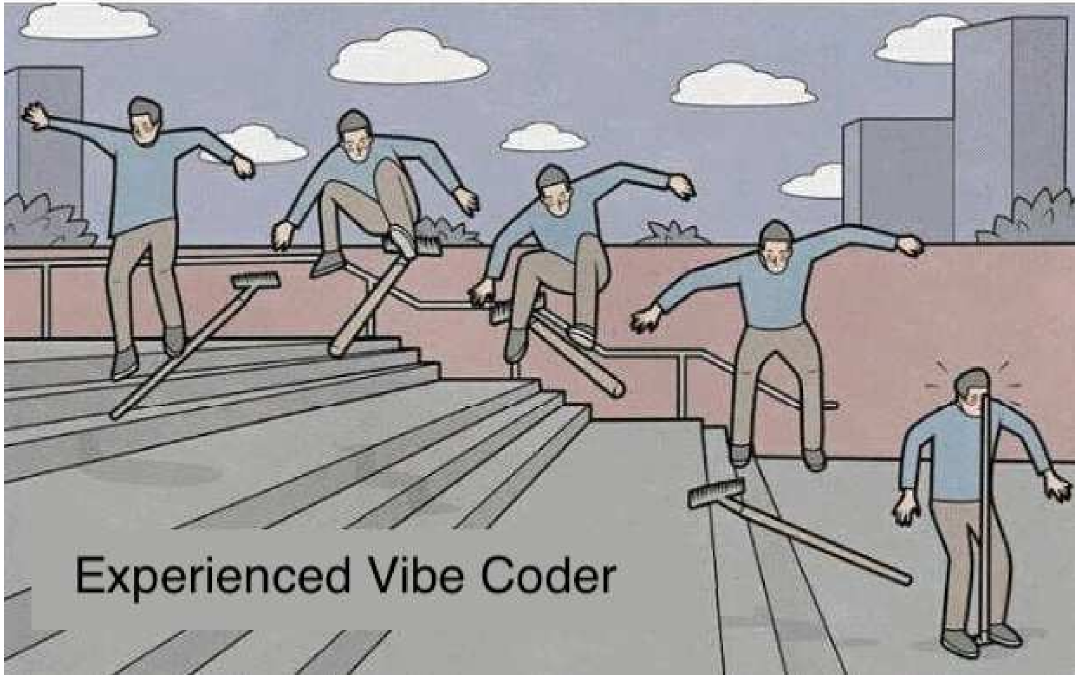
1. Home/Dashboard ([illegible])
2. Transaction Record Page ([illegible])
3. Add Transaction Page (Manual) ([illegible])
4. Add Transaction Page (AI Recognition) ([illegible])
5. Budget Management Page ([illegible])
6. Report Analysis Page ([illegible])
7. Personal Settings Page ([illegible])

workspace to understand the current state.

</think>

I'll help you implement this AI automatic accounting app based on the prototype images you've provided. First, let me explore what we're working with in the

Novice Vibe Coder



Experienced Vibe Coder