COMP110: Principles of Computing

1: Computing Foundations

Learning outcomes

By the end of today's session, you will be able to:

- **Recall** the historical context of computing and gaming technology
- **Explain** the basic architecture of a computer
- **Distinguish** the most common programming languages and paradigms in use today

Today's agenda

- COMP110 course outline
- History of computing
- Computer architecture
- Programming languages and paradigms

Course introduction

From the module guide

This module is designed to introduce you to the basic principles of computing and programming in the context of digital games. It is designed to complement the other modules through providing a broad foundation on the different methods and techniques which will help you to be able to construct computer programs and able to use relevant scholarly sources. You will gain an understanding of software development and the various roles, pipelines, and terminology used within game development.

Topic schedule

On LearningSpace...

Timetable

http://mytimetable.falmouth.ac.uk

Assignments

- Assignment 1: worksheet tasks
  - **Five** worksheets throughout the study block
  - **Five** online quizzes throughout the study block
- Assignment 2: research journal
- See LearningSpace for assignment briefs, worksheets, quizzes
- See MyFalmouth for deadlines

Worksheet A

- SpaceChem
- Quiz: Pythagoras' Theorem
- Due in class on **Friday 6th October** (next week)

Personal tutor meetings

`http://learningspace.falmouth.ac.uk`

- You **must** meet your personal tutor at least **twice** per study block
- If you haven't booked a meeting yet, do it **now**!

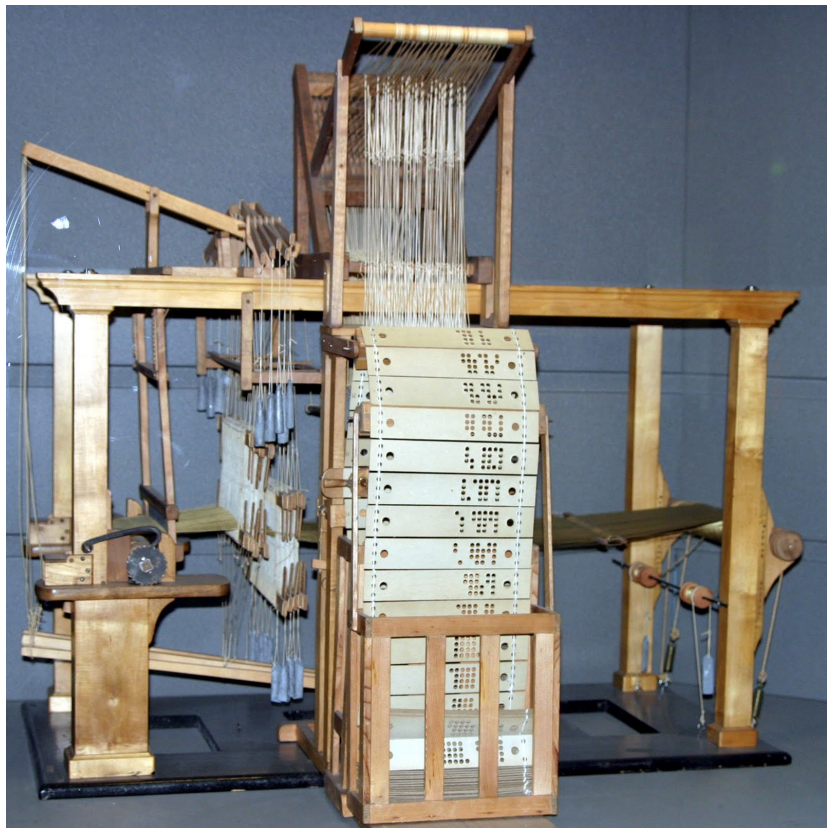
What was the first computer?

Antikythera Mechanism ($\sim$150 BC)

First mechanical computer?

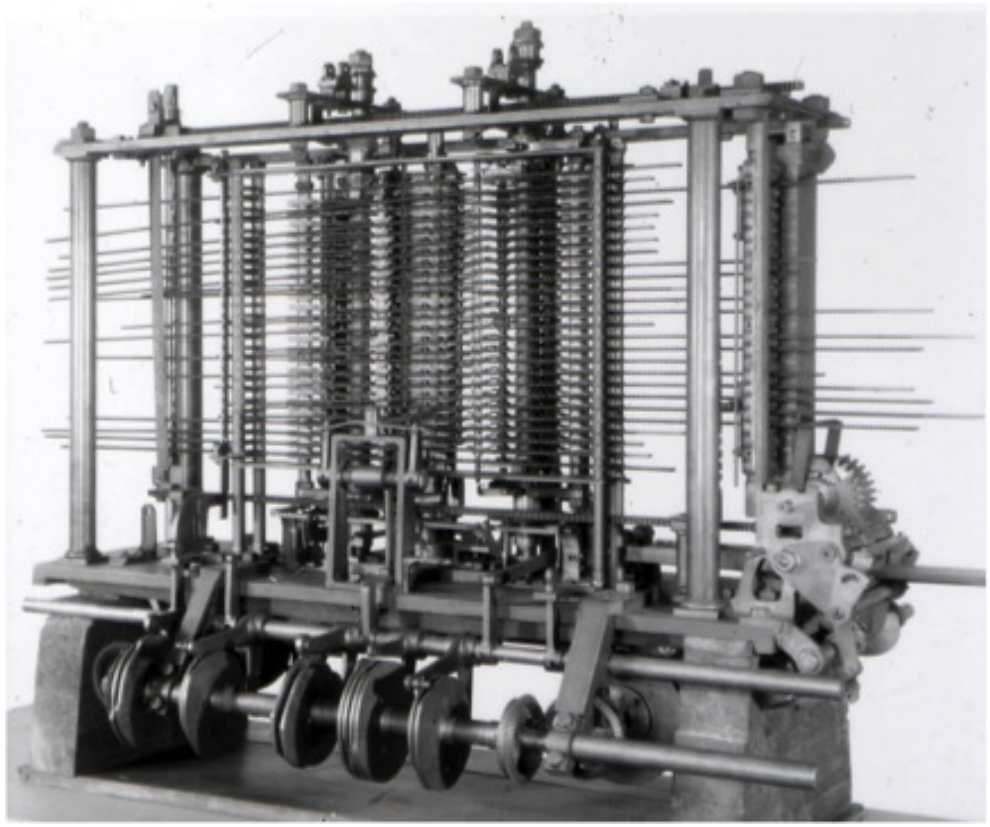
Jacquard Loom (1804)

First programmable machine in modern age



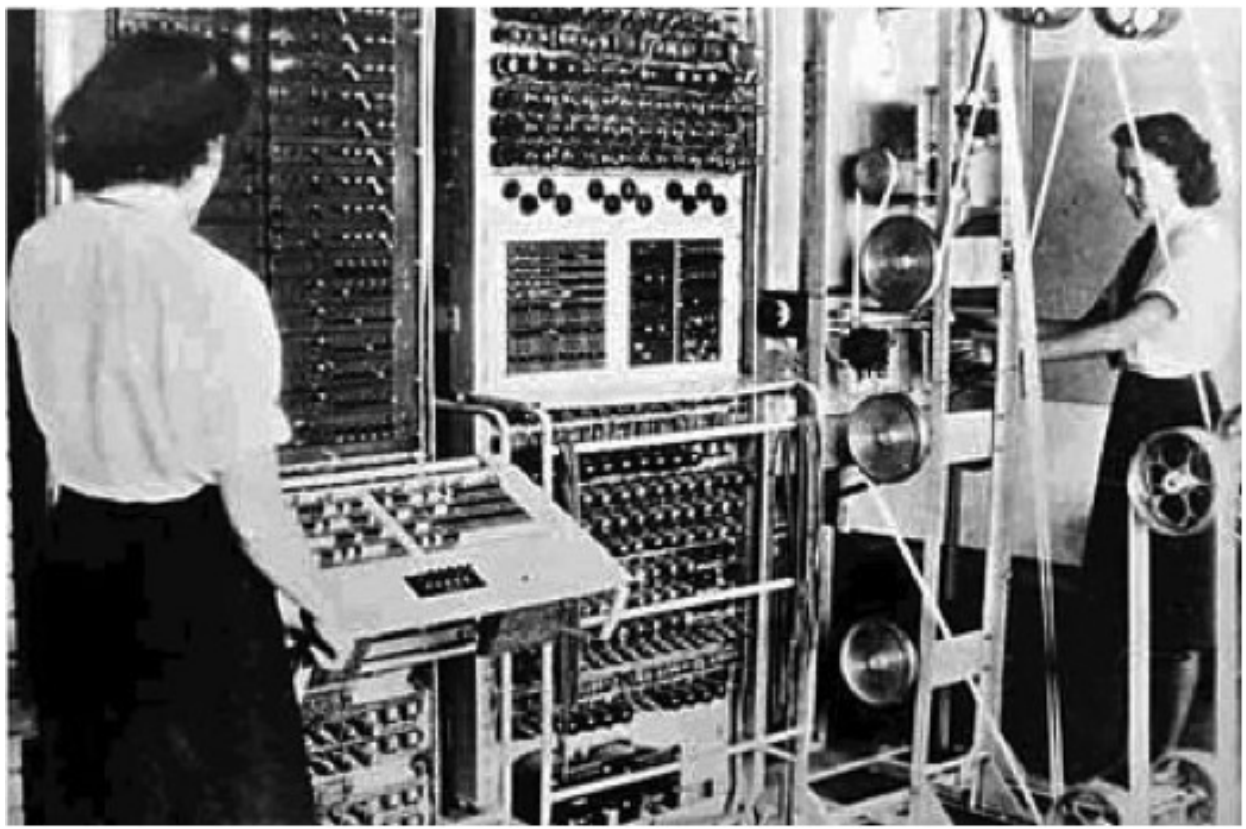
Babbage's Difference and Analytical Engines (1837)

First mechanical computer in modern age



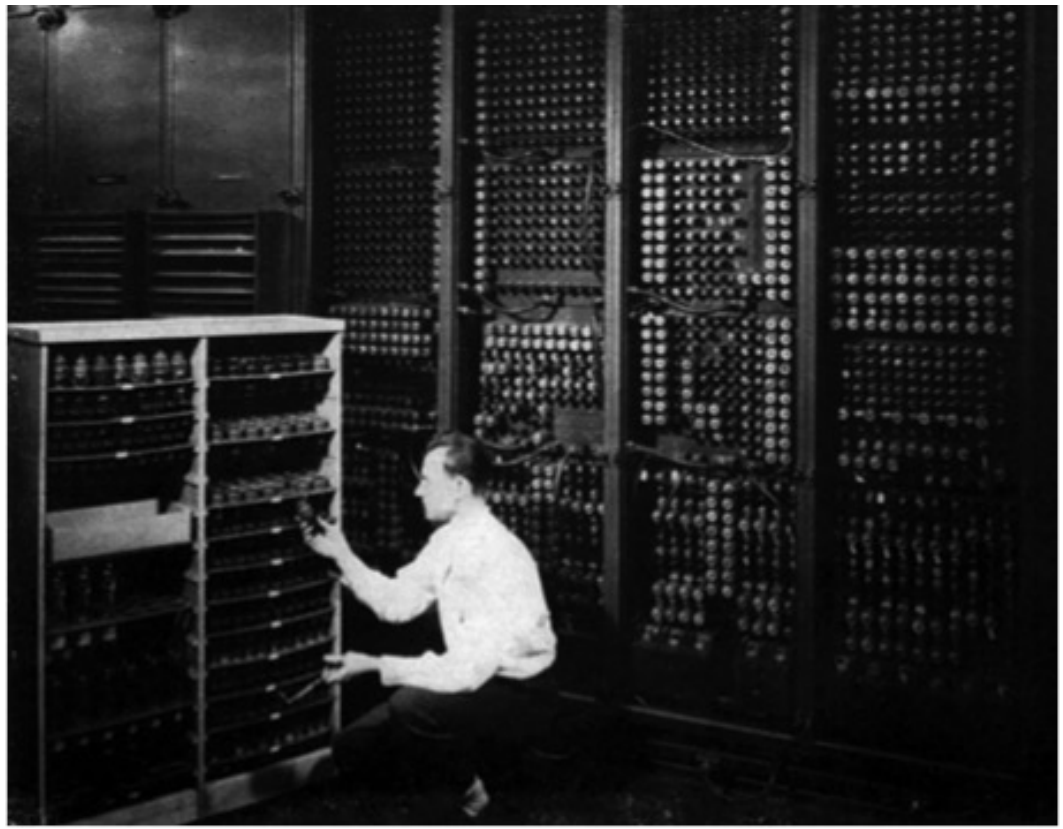
Colossus (1943)

First programmable electronic computer



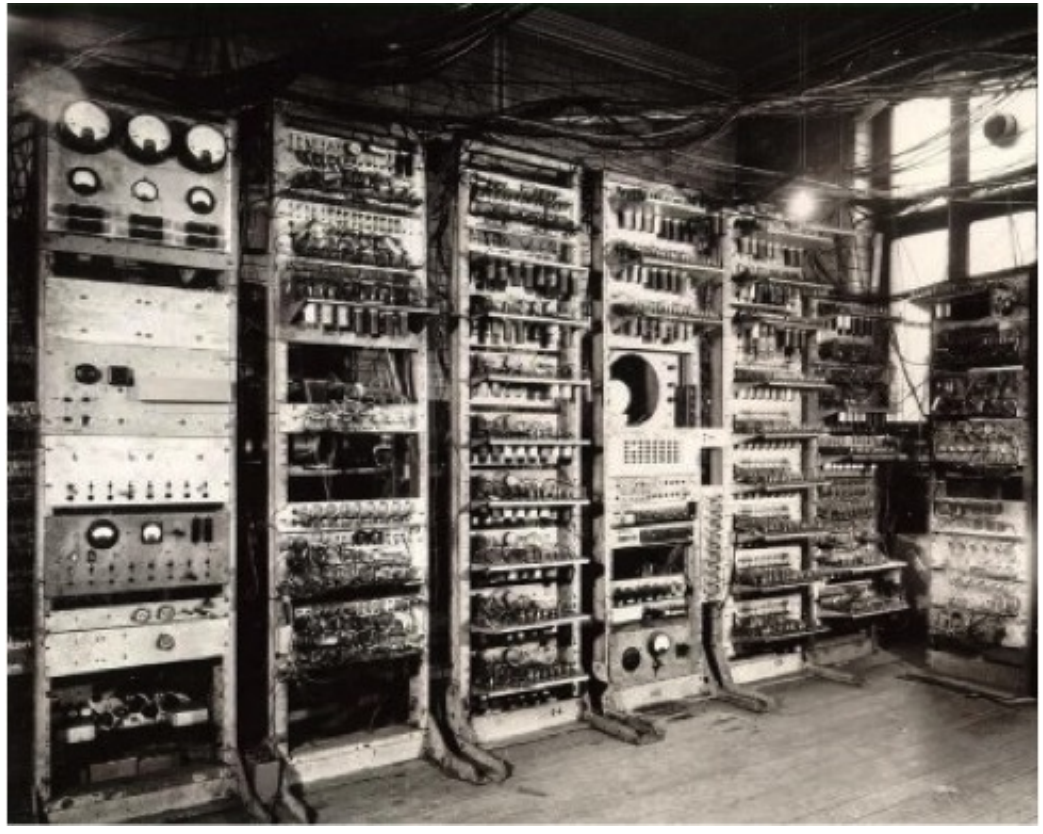
ENIAC (1946)

First general-purpose computer



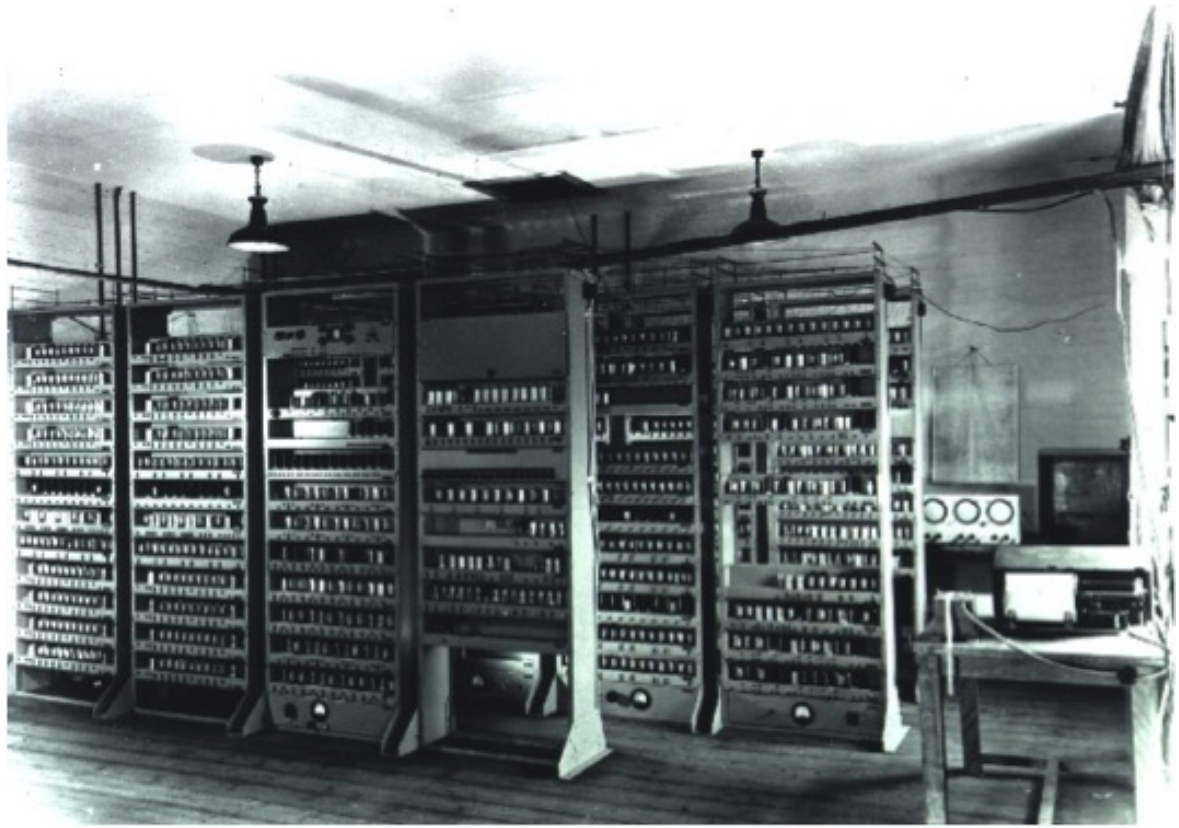
Manchester Small-Scale Experimental Machine (1948)

First stored program computer



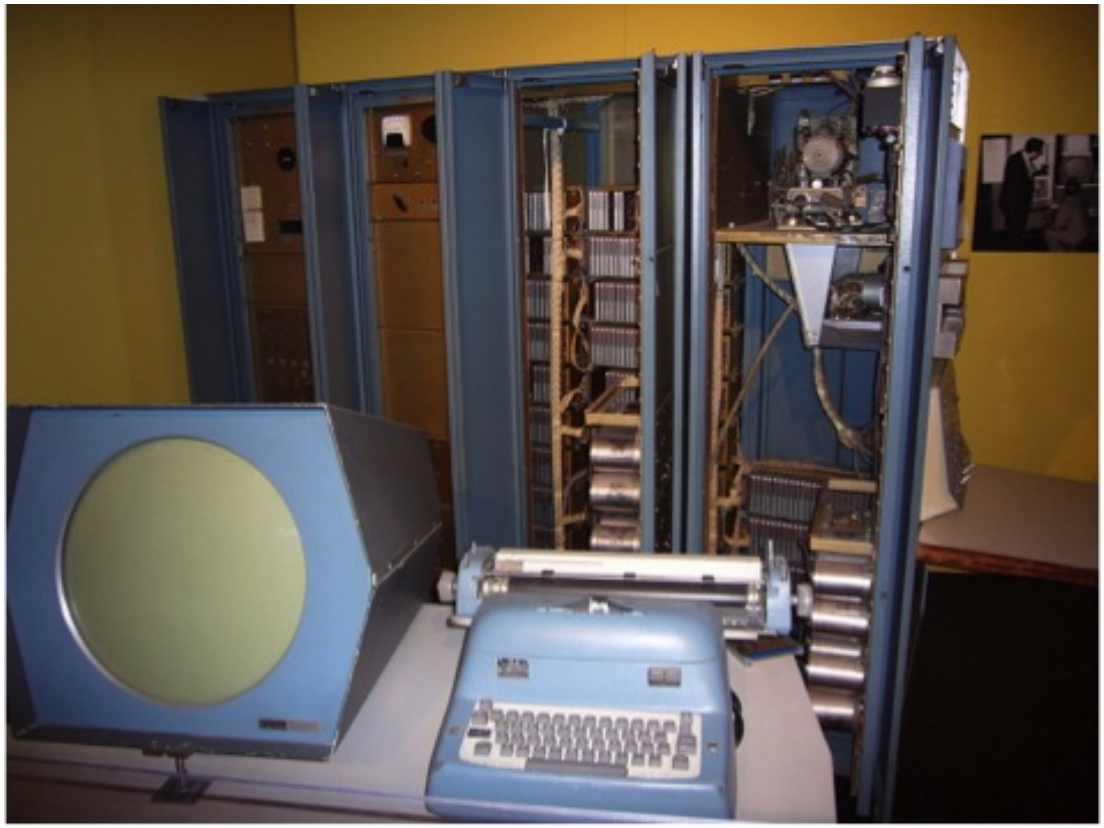
EDSAC (1949)

Many firsts in mathematics and science



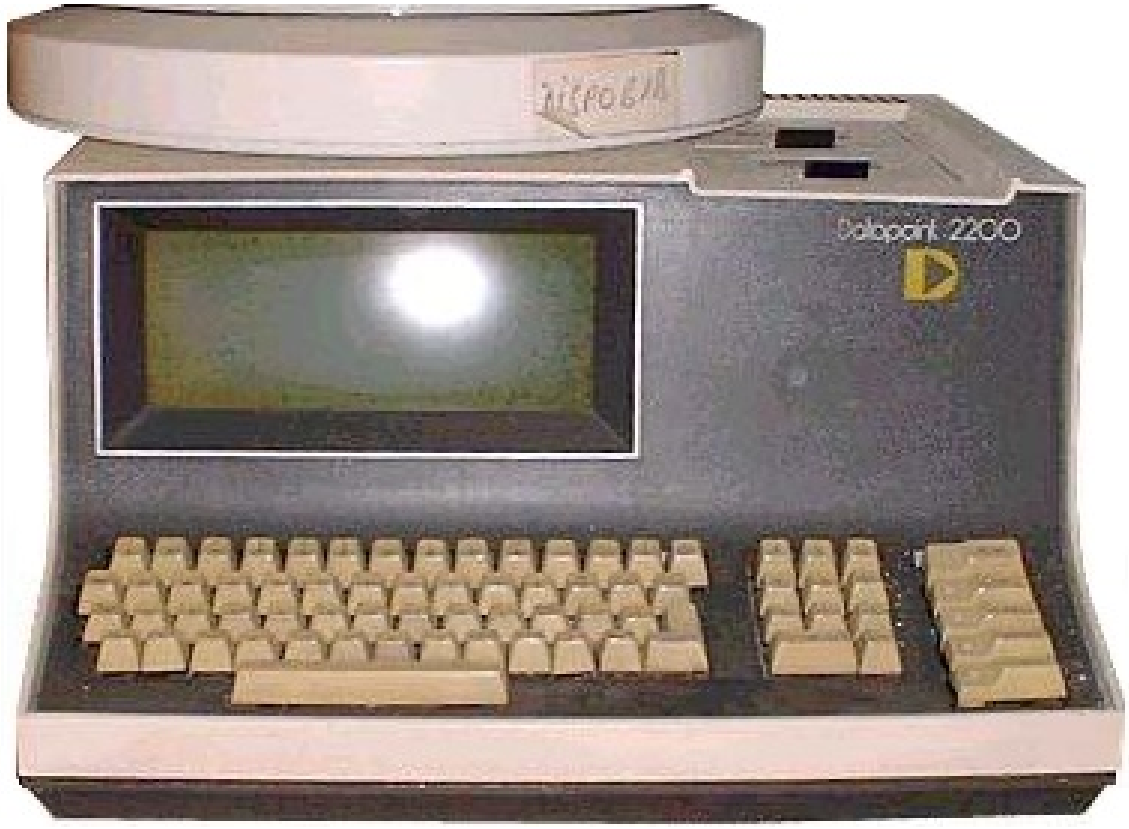
PDP-1 (1959)

Influenced “hacker culture”



Datapoint 2200 (1970)

First microcomputer



Commodore VIC 20 (1980)

First computer to sell 1 million units

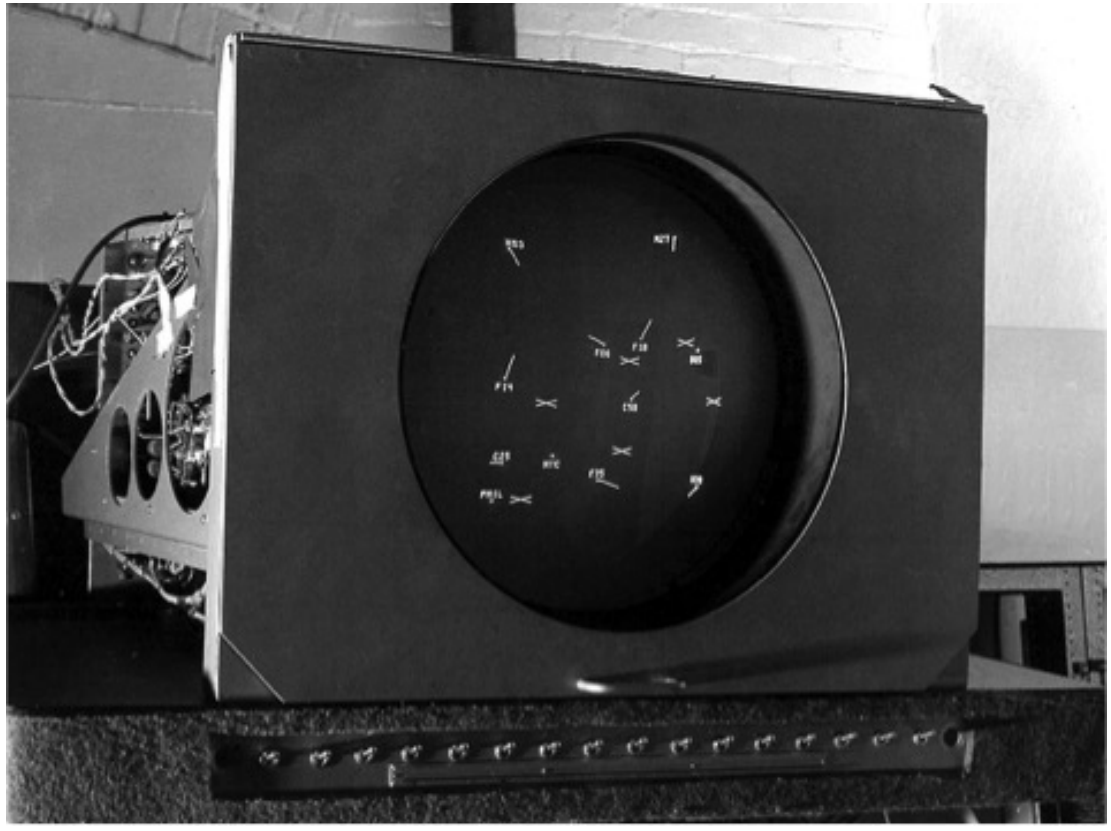
IBM Personal Computer Model 5150 (1981)

Precursor to the modern PC

What was the first computer game?

Cathode Ray Tube Amusement Device (1948)

First interactive electronic game



Chess AI on the Ferranti Mark I (1951)

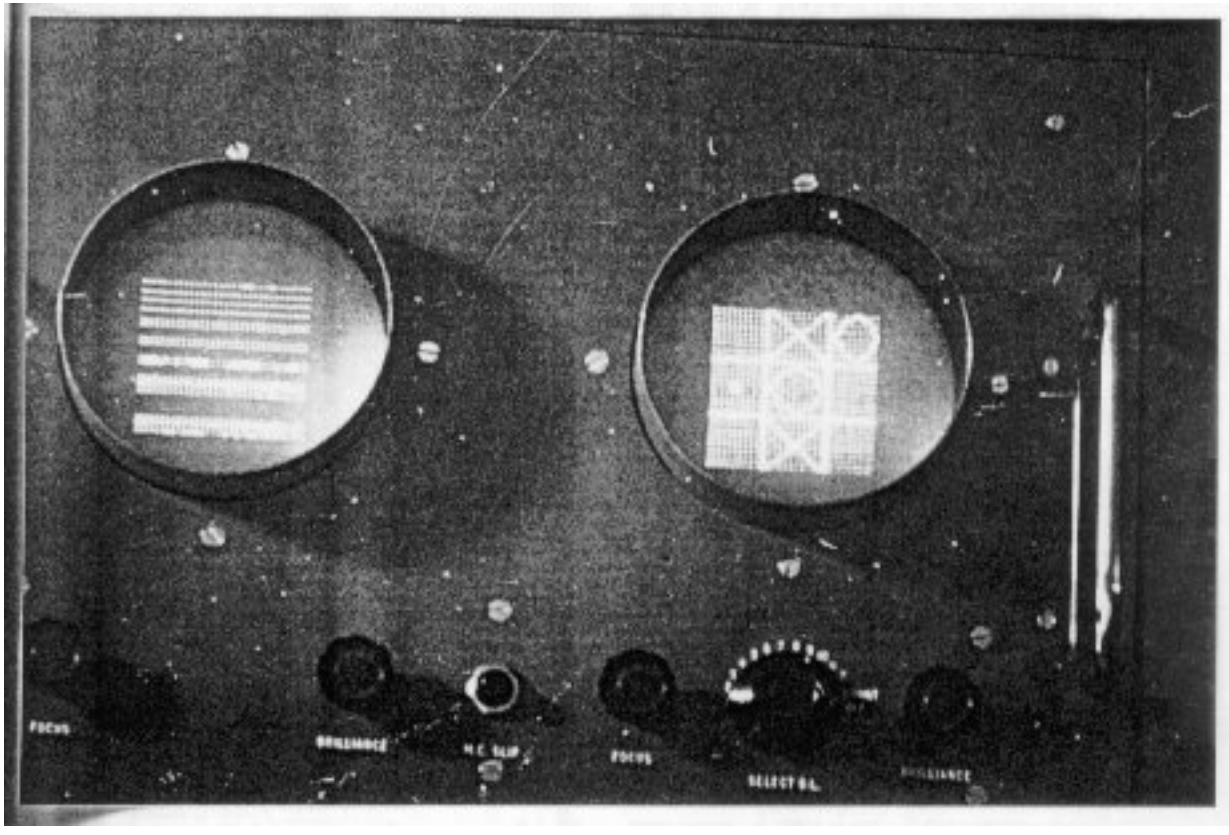
First chess program

Bertie the Brain (1950)

First computer game with a visual display

OXO (1951)

First game with visuals on a general-purpose computer



Tennis for Two (1959)

First to be created purely for entertainment

SpaceWar! (1962)

First widely available game, inspired first arcade games

Pong (1972)

First commercially successful game

What was the first games console?

The Brown Box (1967)

First prototype console

Magnavox Odyssey (1972)

First commercial console

Game console timeline

http://www.onlineeducation.net/videogame_timeline/video-game-timeline.jpg

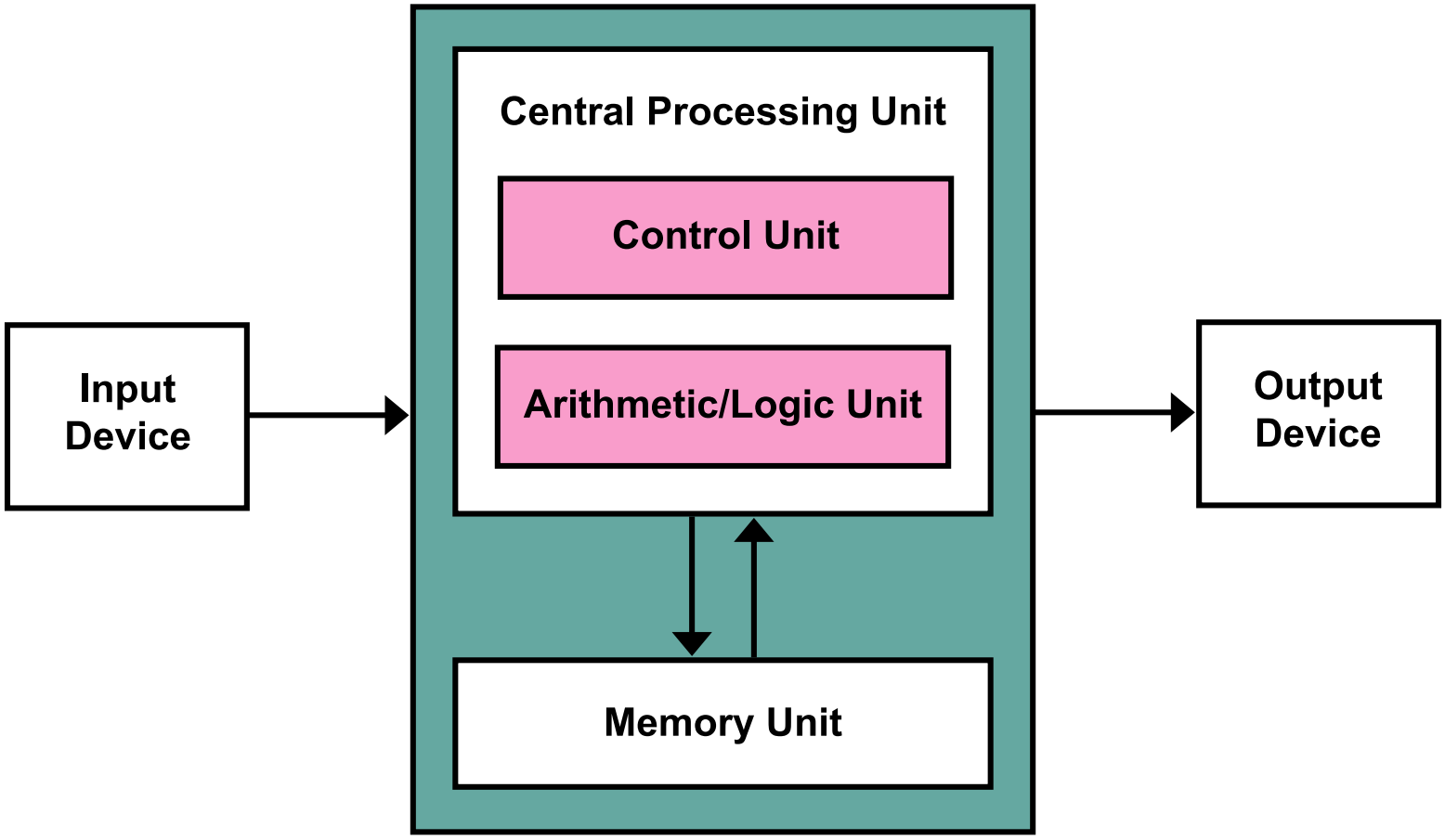
(A little out of date!)

Basic computer architecture

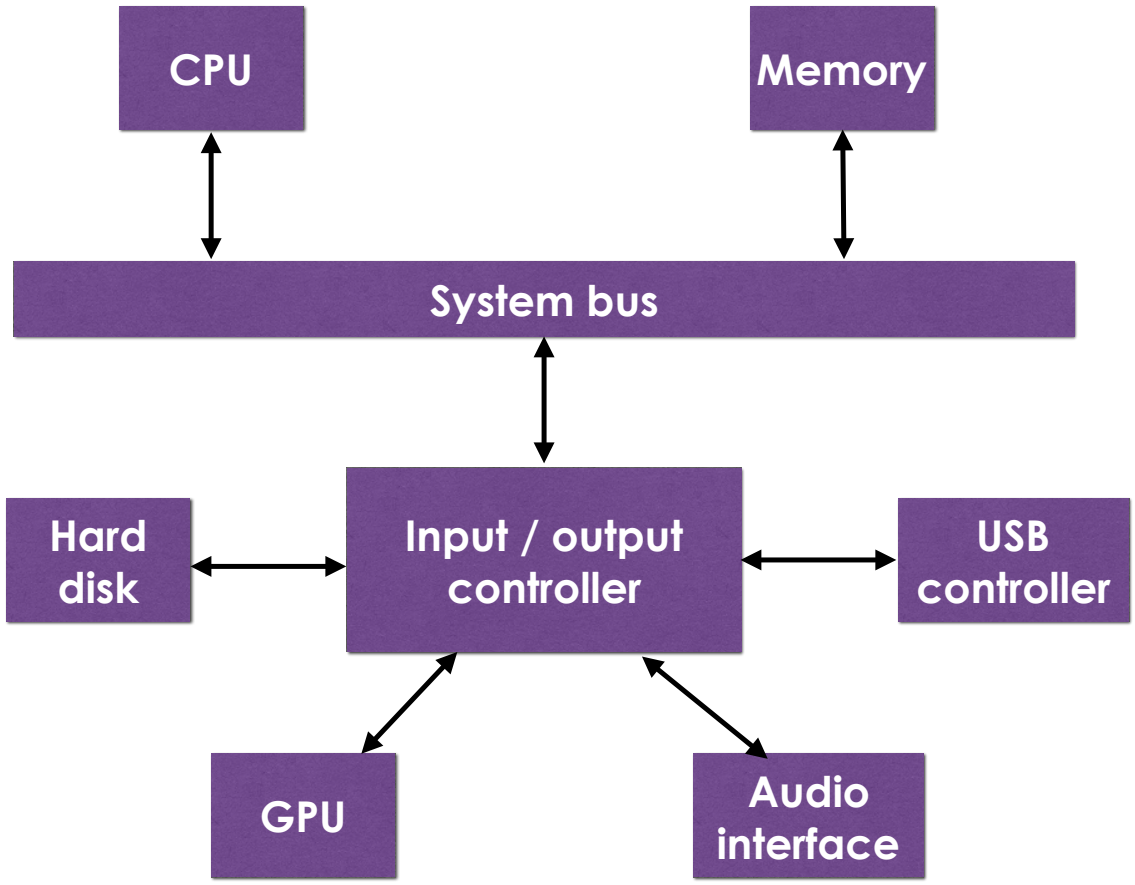
What is a computer?

- In **groups of 2-3**
- Discuss for **10 minutes**
- Go to `www.socrative.com` (or open the Socrative app) and enter room code `FALCOMPED`
- **Individually**, suggest a **one sentence** definition for a computer

The Von Neumann model



Modern PC architecture



Central processing unit (CPU)

Carries out

- Arithmetic operations
- Logic operations
- Control operations

Storage

- Primary storage
  - Directly accessible by the CPU
  - Random access memory (RAM)
  - Volatile — loses its contents when switched off
- Secondary storage
  - E.g. hard disk, SSD, USB flash drive, DVD
  - Non-volatile — keeps its contents when switched off

Graphics processing unit (GPU)

- Responsible for displaying images on screen
- Traditionally, one of many input/output devices
- Nowadays, essentially a highly specialised CPU with its own primary storage

The stored program architecture

- A **computer program** is a sequence of instructions for the CPU
  - (Note: it's spelled "program", not "programme")
- The **programmable computer** — can carry out different tasks depending on what program it is given
- Most modern computers use the **same** memory to store the program and the data it uses

Programming languages and paradigms

What is a programming language?

- A **program** is a sequence of instructions for a computer to perform a specific task
- A **programming language** is a formal language for communicating these sequences of instructions

Which is the best programming language?

- There is no "best" programming language
- There are hundreds of programming languages, each better suited to some tasks than others
- Sometimes your choice is dictated by your choice of platform, framework, game engine etc.
- To become a better programmer (and maximise your employability) you should learn several languages (but one at a time!)

Low vs high level

- **Low level languages** give the programmer direct control over the hardware
- **High level languages** give the programmer **abstraction**, hiding the details of the hardware
- High level languages trade efficiency for ease of programming
- Lower level languages were once the choice of game programmers, but advances in hardware mean that higher level languages are often a better choice

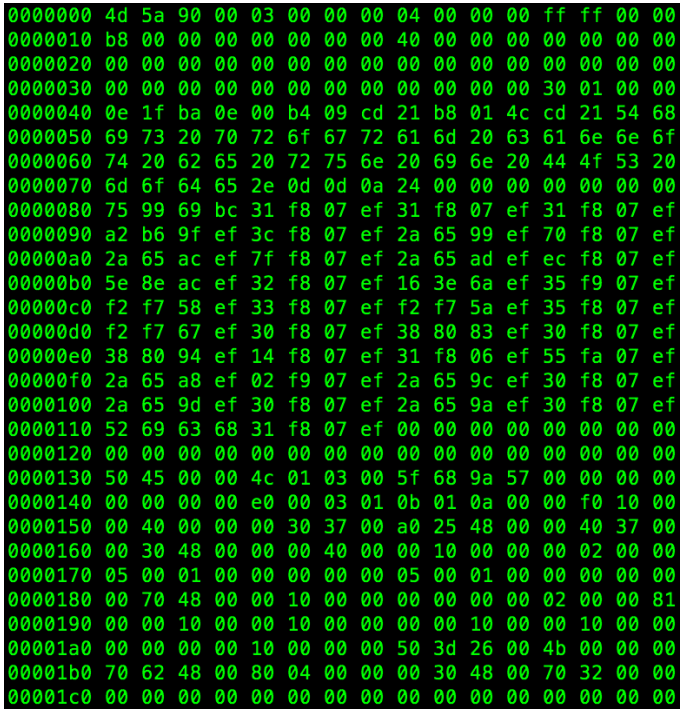
Programming paradigms

- **Imperative**: program is a simple sequence of instructions, with **goto** instructions for program flow
- **Structured**: like imperative, but with **control structures** (loops, conditionals etc.)
- **Procedural**: structured program is broken down into **procedures**
- **Object-oriented**: related procedures and data are grouped into **objects**
- **Functional**: procedures are treated as mathematical objects that can be passed around and manipulated
- **Declarative**: does not define the control flow of a program, but rather defines logical relations

Which paradigm?

- **Imperative** and **structured** languages are mainly of historical interest
- Most commonly used languages today are a mixture of **procedural** and **object-oriented** paradigms, with many also incorporating ideas from **functional** programming
- Purely **functional** languages are mainly used in academia, but favoured by some programmers
- Purely **declarative** languages have uses in academia and some special-purpose languages

Machine code



- Programs are represented as sequences of **numbers** specifying **machine instructions**
- More on this later in the module
- Nobody has actually written programs in machine code since the 1960s...

Assembly language

```
section     .text
global      _start

_start:

    mov     edx,len
    mov     ecx,msg
    mov     ebx,1
    mov     eax,4
    int     0x80

    mov     eax,1
    int     0x80

section     .data

msg     db  'Hello, world!',0xa
len     equ $ - msg
```

- Each line of assembly code translates **directly** to an instruction of machine code
- Commonly used for games in the 70s/80s/90s, but hardly ever used now
- Allows very fine control over the hardware...
- ... but difficult to use as there is no **abstraction**
- Also not portable between CPU architectures

C++

```cpp
#include "stdafx.h"
#include "GameObject.h"
#include "CoinGame.h"

GameObject::GameObject(CoinGame* game, Texture* sprite)
	: game(game), sprite(sprite), isDead(false)
{
	x = rand() % CoinGame::WINDOW_WIDTH;
	y = rand() % CoinGame::WINDOW_HEIGHT;
}


GameObject::~GameObject()
{
}

void GameObject::render(SDL_Renderer* renderer)
{
	sprite->render(renderer, x, y, CoinGame::SPRITE_SIZE, CoinGame::SPRITE_SIZE);
}

bool GameObject::checkCollision(int otherX, int otherY)
{
	double distance = sqrt(pow(otherX - x, 2) + pow(otherY - y, 2));
	return (distance < CoinGame::SPRITE_SIZE / 2);
}
```

- Initially an object-oriented extension for the procedural language C
- Low level (though higher level than assembly)
- Used by developers of game engines, and games using many popular “AAA” engines (Unreal, Source, CryEngine, …)
- Also used by developers of operating systems and embedded systems, but falling out of favour with other software developers

High level languages

Often favoured by smaller indie teams for rapid development

- C# (XNA, Unity)
- Python (EVE Online, Pygame, Ren'py)
- JavaScript (HTML5 browser games)
- ActionScript (Flash games)
- Objective-C, Swift (iOS games)
- Java (Minecraft, Android games)

There are many others, but these are the most commonly used in game development

Scripting languages

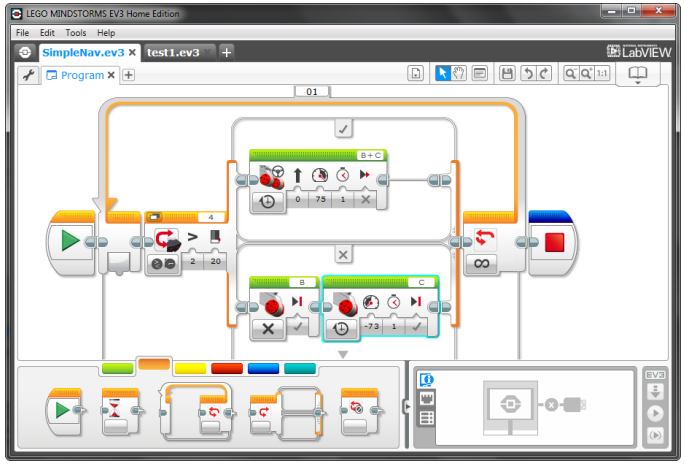
Many games use scripting languages in addition to their main development language

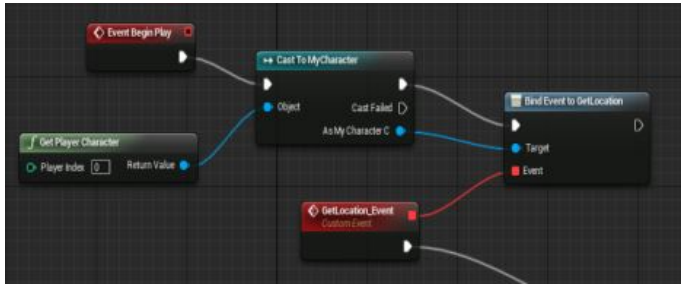
- Lua (many AAA games)
- Bespoke languages (many AAA games)

Some game engines have their own scripting language

- UnrealScript, Blueprint (Unreal Engine)
- GML (GameMaker)

Visual programming languages





Based on connecting graphical blocks rather than writing code as text

- Scratch (used for teaching in school)
- Lego Mindstorms
- Blueprint (Unreal)

Note: despite the name, Microsoft Visual Studio is **not** a visual programming environment!

Special purpose languages

- SQL (database queries)
- GLSL, HLSL (GPU shader programs)
- LEX, YACC (script interpreters)

Markup languages

Not to be confused with programming languages...

- HTML, CSS (web pages)
- LaTeX, Markdown (documentation)
- XML, JSON (data storage)

Which programming language is most popular?

`http://githut.info`

“Family tree” of programming languages

https://www.levenez.com/lang/lang.pdf

Debrief

You should now be able to:

- **Recall** the historical context of computing and gaming technology
- **Explain** the basic architecture of a computer
- **Distinguish** the most common programming languages and paradigms in use today

Remember: Worksheet A is due **this time next week**!