Sandbach School Computer Science Curriculum

Intent

Year 12 and 13

Learners will begin their studies by taking on aboard some new languages: C#, PHP and JavaScript. This means that the fundamentals learned at GCSE are applied to 6th form study in a new and interesting way, and then advanced towards professional development environments. Units have been ordered so that, for each area of study, learning is recapped for GCSE and/or Year 12 and then progressed until final mastery is attained in Year 13.
High expectations are surrounding the final programming project with learners being given a free reign on project choice, development environment and language.

Year 9, 10 and 11

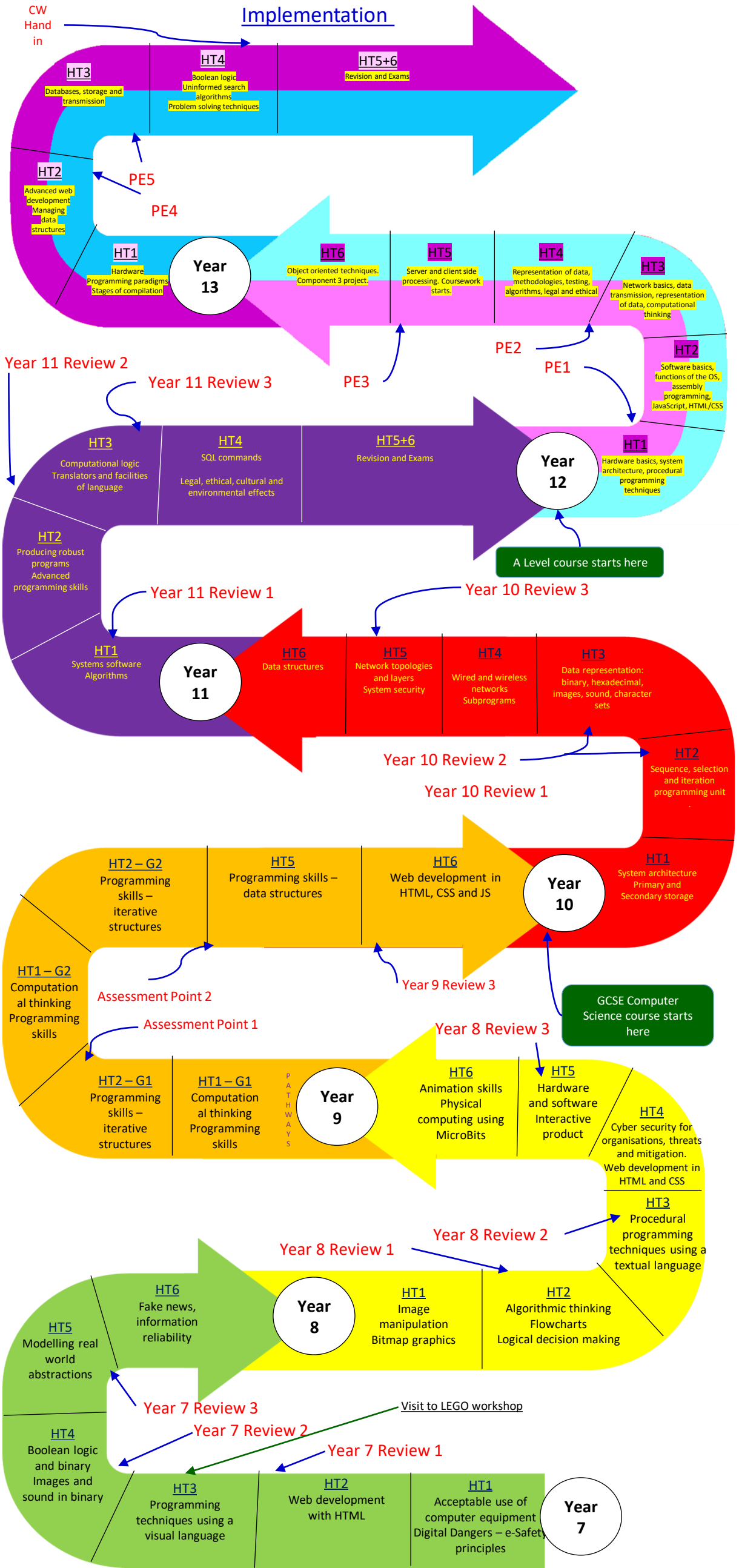
The Year 9 curriculum has been carefully chosen to firstly give pupils a flavour of the style of learning that takes place in the GCSE course, and then to give enriching experiences surrounding the area of computer science that are beyond, yet supportive of, the taught GCSE specification.
The GCSE course is taught in specification order in terms of the theoretical sections, but programming is interleaved in between each unit so that learners are constantly revisiting the more practical natures of the course. The programming curriculum has been carefully sequenced across years 7 to 11 to be progressive: each year begins with recapping previous skills before moving on to advance skills and knowledge with more challenging learning.

.

Year 7 and 8

The computing curriculum in the transition and induction phase aims to give learners an in depth journey through the National Curriculum content so that they are able to access digital technology successfully and safely both in school and in the outside world.
In addition, core skills from the two taught qualifications in the bridging and qualifications phase (computer science and creative imedia) are interleaved so that a firm grounding of both disciplines is evident. This sets learners up for success should they make those course choices and also enables them to make informed curriculum decisions when choosing future courses and careers.

Implementation



Impact

By the end of year 13 students will:

- Have near professional grade programming skills in a range of languages, including, but not necessarily limited to: Python, C#, JavaScript, PHP, SQL
- Understand how problems can be decomposed, abstracted and solved using a wide range of problem-solving techniques that can be fully analysed and evaluated
- Have detailed knowledge of system architecture including the evolution of CPU design and the classic Von Neumann architecture in precise detail
- Have embarked on an ambitious and significant software development project, fully understanding the entire software development life cycle
- Appreciate how networks, such as the internet, work in terms of layers, transmission techniques and encoding
- Be able to use object-oriented techniques to advance programming and software engineering design principles
- Know laws and regulations surrounding the use of IT

By the end of year 11 students will:

- Have developed excellent problem-solving skills
- Be able to solve problems using sequence, selection and iteration
- Be able to write modular code using sub programs
- Have a firm understanding of data structures such as arrays, 2D arrays and records
- Understand PC hardware including Von Neumann architecture and the fetch execute cycle
- Be confident at discussing issues surrounding cyber security: threats to networks and ways to minimise threats
- Be confident at discussing legal, ethical, cultural and environmental concern surrounding computer science
- Have the ability to think algorithmically and know standard algorithms

By the end of year 8 pupils will:

- Understand binary and how many real world aspects of computing are represented in binary
- Have experience of programming techniques in a visual and textual programming language plus scripting in HTML
- Understand how to use computers safely and responsibly plus some of the wider threats to computer systems
- Have experience of algorithmic thinking and have learned some standard algorithms
- Be able to use multiple software packages to create digital artefacts for a given purpose and audience
- Understand some simple hardware and software components
- Understand simple Boolean logic operations and gates
- Have designed, used and evaluated computational abstractions that model the state and behaviour of real-world problems and physical systems