Creative Computing Encouraging Artistry and the Freedom to Fail

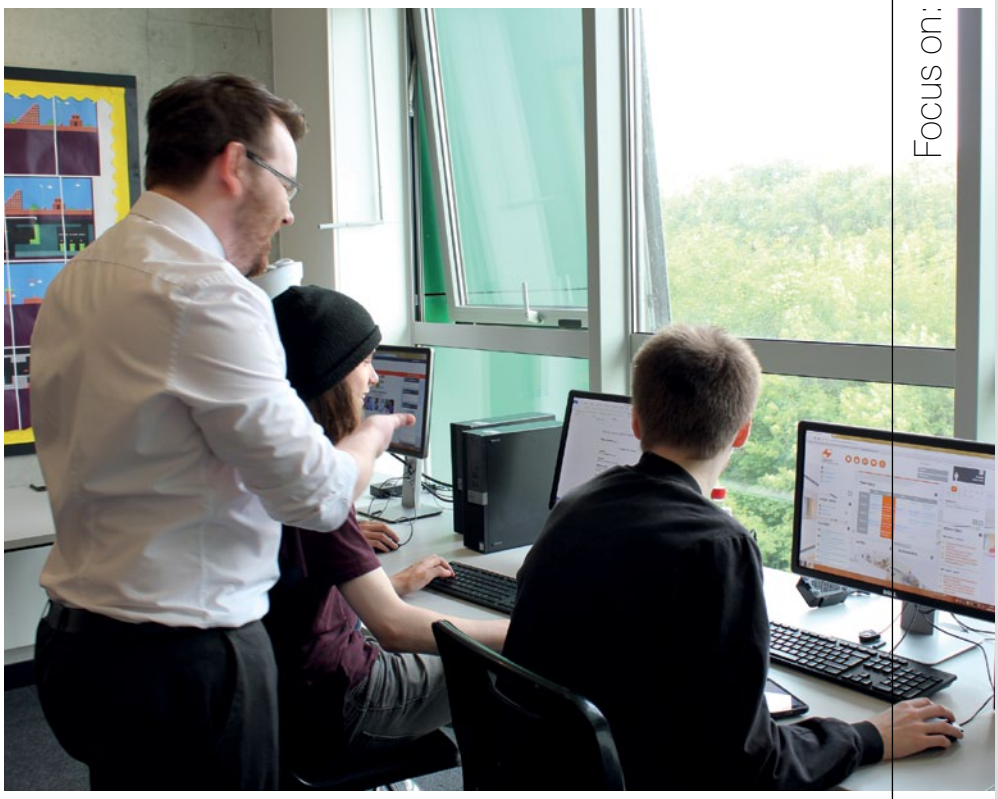
BAFTA Young Game Designer Mentor Award winner **Dave Chilver** shares his principles for how to keep creativity at the core of computer science classes. He highlights how leaving room for mistakes and 'failure' helps students access their creative impulses and overcome the technical challenges associated with computing.

The classroom opposite mine had a sticker in the window that the teacher had put there after a holiday to the USA. It was clearly visible from my room and was perhaps the most American thing I could imagine. It had a backdrop of a fluttering stars and stripes flag and a faded out bald eagle on the left with the statement "Failure is not an option" emblazoned across the centre. This quote from the movie Apollo 13 was often reused during exam season as a motivational tool. I'm not sure how successful it was but I'm sure many students found strength in the words and attitudes expressed by those pioneers of space travel. However, for my day-to-day lessons, I soon discovered that fear of failure had more of an impact on my students' attainment than pretty much any other factor you'd care to measure.

As a teacher of ICT, Computer Science and Game Design across Key Stage 3, 4 and 5, I have seen several trends emerge in recent years. Firstly, despite numerous attempts, the percentage of girls taking my subject has gradually fallen to embarrassingly low numbers. Secondly, my subject is increasingly seen as being technical rather than creative. I'm not senior enough to know why exactly these changes have occurred, but I do know that we need to try to reverse them. Computer Science and its 'cooler' cousin, Games Design, in my mind are much more 'arty' than some believe. Learning the language and syntax of code is very similar to the steps students take in Spanish and French or even in Music. A good coder or game designer will have much more in common with an artist approaching a blank canvas with just the beginnings of an idea than a mathematician searching for the correct outcome. I sketched out three rules to follow which have changed the way I teach my subject.[1]

Rule 1: Quick wins

A quick win is something a student should be able to learn and complete to a set level within the space of one lesson. It should also be something meaningful. A quick win gives a student a moment of satisfaction that they've achieved the impossible. Let's take the example of learning Python coding for the first time. For many students this can make or break their experience of Computer Science. Traditionally we should start with 'Hello World' and move through input/output tutorials to IF statements and beyond.



After about 5 to 6 relatively simple tutorials the students should be able to make some creative choices, but by that stage I often find that students have become frustrated at errors that have occurred or perhaps by the nature of the task. "When are we going to make a game?" they cry out.

Instead I start by asking the students to copy out some code from the board. No copy and paste – part of the challenge is in the typing.

```
def castle():
    print("You are outside a castle")
    choice = input("Type L to go left or type R to go right")
    if choice == "L":
        lake()
    elif choice =="R":
        forest()

def lake():
    print("You are next to a lake")
    choice = input("Type L to go left or type R to go right")
    if choice == "L":
        forest()
    elif choice =="R":
        castle()

def forest():
    print("You are in a forest")
    choice = input("Type L to go left or type R to go right")
    if choice == "L":
        castle()
    elif choice =="R":
        lake()


castle()
```

Run the code and you'll get the world's shortest text adventure game. Instant quick win. Then we have to figure out how to make more locations, and for the most part the students easily figure out that the 'blocks' of code under the def must be the locations. Very quickly I'll have students who will customise the names and directions, sometimes even adding multiple additional paths. The discovery of how the code works is now in the hands of the students and, in my opinion, this ownership empowers them to experiment for themselves. Coding errors can seem like a failure, but here it means they've tried to do something cool and missed. I'll ask the students to add a new location and perhaps only 10 per cent get it right the first time. As this task is very student-centred, I find that those who get it working first are quick to spread the news of their success and to inform those around them of the solution.

With every new development to their game they learn how to fail in new ways, but that it's ok because someone else in the room has their back. By the end of our 45-minute session every student has a simple text adventure game and their knowledge of Python syntax has increased from 'speaking' the language in context. When we teach French, we start by getting them to say "hello" and introduce themselves (a quick win) – we don't start with a lecture involving the rules for using apostrophes!

Rule 2: Hearts and minds

I'm told that Computer Science is for geeks (or worse) but I don't believe it. Computer Science is for dreamers who want solutions to problems. It's about facing limitations and

discovering workarounds. Many other subjects seem to have a better PR record than us and I work hard to change it. Our equivalent to sports days are Game Jams and I'm a big believer that they can change the way the subject is perceived.

A Game Jam is a short competition (usually 24 or 48 hours) where people work in groups to design a game using a theme or limitation. There are huge international competitions where people from all around the world do this at the same time and it's a great way to build links with local universities that might participate. Themes can be very clear, such as 'design and/or create a game that only uses two buttons to play'. They can also sometimes be more cryptic and hint at an emotional subtext, such as the theme for Ludum Dare #20 "It's dangerous to go alone! Take this!". I love the idea of Game Jams because they push students to think creatively rather than simply reinventing what already exists.

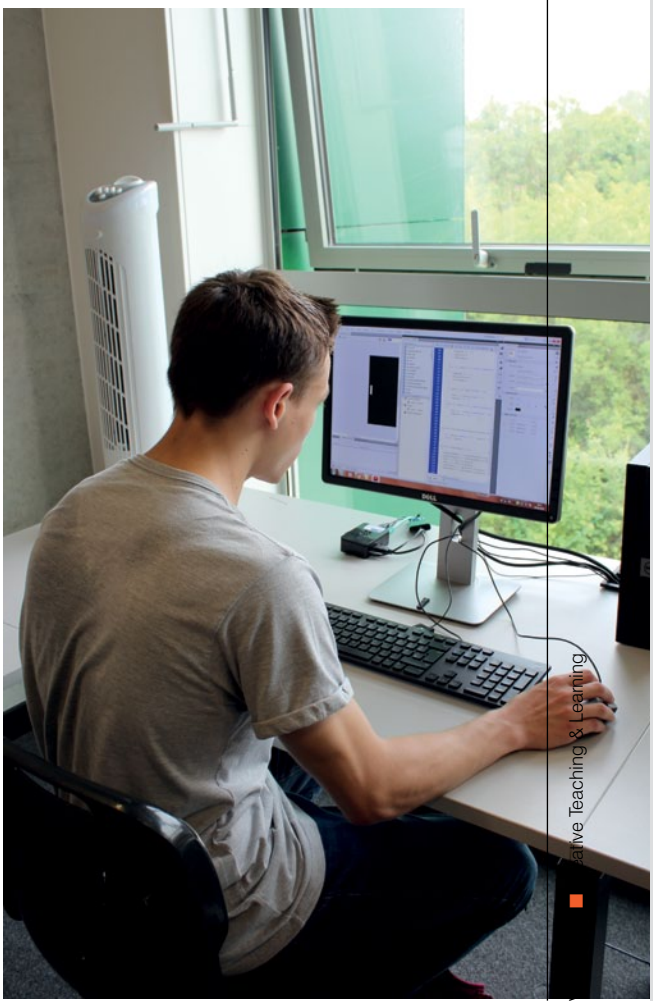
A take I've found popular with my students is a one-hour Game Jam. For this task, I come

up with four to five categories (e.g. emotion, weather, smell and taste) and then I write six examples for each category (e.g. fear, happiness, sadness, determination, resignation and ambivalence). I ask each student in the class to pick a number for a category and then reveal the word that they have chosen. With their 4 or 5 words they must design a story for a game, along with several characters and a goal for the player to achieve. The creative interpretations of words are fantastic, and as long as you spin the order of the words or add some extra ones/categories it's a very easy task to repeat throughout the year. The best examples are illustrated to make great displays and I have a competition for the best interpretation each time. It produces a great buzz around the subject and is a welcome change from the diagrams of keyboards I previously had up on my walls. Students who 'can't' code suddenly want to know if their steampunk arcade game in which they must protect their scampering puppy from 'fire rain' using a metal umbrella is easy enough to make in Scratch.

Rule 3: Don't forget your Magnum Opus

Big coding challenges always appeal to my students, especially the younger ones. Every time we start a new coding topic I'm asked if I can teach them how to make Mario/FIFA/Fortnite. The answer is, of course, no, and it's demoralising for them to hear it and for me to say it. Instead we need to harness that drive. For my students we have a long-term goal, something to aim for in the future. Perhaps, by the end of this year, they will be able to make a Fortnite clone (replace with current trend) in Scratch/Unity/etc. We may spend a lesson discussing it and breaking it down into simple ideas. "I need a way of getting 100 players on a map", or "I need a way to switch weapons", etc. After that first discussion, we leave it to one side and as the year progresses, every time we learn a new skill that might help us complete our Magnum Opus we make a quick note of it.

By the end of the year some students will be ready to have a go, some will still need more time, others may even be able to reflect on



what their idea was and realise that it needs to change. Student choice is an important aspect of Computer Science. It's very easy to opt out of the challenging aspects if they have no direct personal relevance. My Magnum Opus still isn't complete almost six years after I first laid down the initial plans, but I'm working on it and it inspires me to improve my skills when marking levels allow it. For students it's about making the subject more creative. The GCSE can be very challenging for students who haven't completely bought into the subject. KS3 for me is an incredibly important time where we as educators have the chance to guide students toward making the right choices. That Fortnite clone my Year 7 class are close to finishing may not change the world, but it has certainly inspired many of them to put in countless extra hours.

Failure is an option

I believe that Computer Science and IT are at an important junction and the destruction of most of the IT courses with the last curriculum was clearly a case of throwing the baby out with the bathwater. Much of the wonderful creative aspects have been lost in the search for a Holy Grail of academic vigour. That said, Computer Science doesn't need to be scary, and in fact it can be just as creative with the right approach from an early age. My steps won't save a Year 11 class only weeks away from their exams but I'm certain they'll have a massive impact on that Year 7 group who are just finding their feet.

Every year that we face the same challenges in the same way is another year where a sizable number of the cohort are dissuaded from taking up a fantastic subject with huge potential. All we need is to let the students fail a little bit more and help them develop their skills to cope. Each one of my rules will result in students failing something quickly. That's ok, because we remember our failures and we find creative ways to adapt and improve. Fearing failure holds us back, both staff and students, from achieving what we want. Failure ***is*** an option, and potentially it's the best one.

Dave Chilver is an inspirational game design and ICT teacher and was the 2017 winner of the BAFTA YGD Mentor Award.

NOTES:

1. These ideas have been adapted from the great work being done by the staff at the University of Suffolk (especially Rob Kurta) and the University of Lincoln. In addition, Mel Phillips at BAFTA YGD has been instrumental in developing an inspirational games program of study for Secondary and Sixth Form students. There's an obvious need to get creativity into computing and I'm glad that there are people working on it!