Supporting Creativity and Motivation in Learning Programming: A Musical Treatment

Dr Chris Nash (chris.nash@uwe.ac.uk)
Senior Lecturer and Course Leader, Audio & Music Technology
Department of Computer Science & Creative Technologies, UWE Bristol
Researching **digital pedagogies and tools** for:
- Learning programming (Computer Science).
- Learning music (Theory and Composition).

- **BACKGROUND**
- **MANHATTAN PROJECT**
  - Software (with demo)
  - Digital Pedagogy
- **EMPIRICAL FINDINGS**
- **FUTURE DEVELOPMENT**
Creativity in Education

Creativity requires **expertise and motivation**.
(but learning is a form of creativity)

High-threshold, theory-heavy fields (e.g. programming, music)

Bloom’s Taxonomy (flipped)

- **Remembering**
  - Find or remember information
  - List, Find, Name, Identify, Locate, Describe, Memorize, Define

- **Understanding**
  - Understanding & making sense out of information
  - Interpret, Summarize, Explain, Infer, Paraphrase, Discuss

- **Applying**
  - Apply information in a new (but similar) situation
  - Use, Diagram, Make a Chart, Draw, Apply, Solve, Calculate

- **Analyzing**
  - Take information apart & explore relationships
  - Categorize, Examine, Compare, Contrast, Organize

- **Evaluating**
  - Critically examine information & make judgements
  - Judge, Test, Critique, Defend, Criticize

- **Creating**
  - Use information to create something new
  - Design, Build, Construct, Plan, Produce, Devise, Invent
Technological Context

• Lower **computer literacy**
  – 1980’s: micro-computers, programming languages, command line
  – 1990’s: GUIs and usability, “power users”, home computing
  – 2000’s: controlled ecosystems (“Smart” devices), Internet
  – Computer Science → ICT curricula

• Recent **digital learning** initiatives
  – ICT → Computer Science curricula
  – Novice programming languages (e.g. Scratch)
  – Raspberry Pi, Arduino, BBC micro:bit, etc.
  – “Institute of Coding” funding (2015/2017)

• Tools for **music programming**
  – Max/MSP, SuperCollider, OpenMusic, Sonic Pi
  – Programming for non-programmers (“end-user programming”)

• Traditional music pedagogies are front-loaded with theory, repertoire recital, and rote learning.

• More recent music pedagogies:
  – **Orff Schulwerk** (1920-)
    (understanding through creativity)
  – **Kodály Method** (1935-)
    (understanding through practice / doing)
  – **Gordon Music Learning Theory** (1975-)
    (understanding through audiation)
Learning Objectives in Music and Computing

• Develop analytical thinking styles.

  Recognise, use, and adapt patterns.

  Learn from and draw on extant works to innovate.

• Develop more abstract thinking styles.

  Novice ~ focus on low-level detail

  Expert ~ focus to high-level structure ("big picture").

• Use technology to enhance productivity and creativity.

Core learning objectives in all domains that are creative and technical, including music and computer science.
### Editing vs. Programming

<table>
<thead>
<tr>
<th>MUSIC EDITING</th>
<th>MUSIC PROGRAMMING</th>
</tr>
</thead>
<tbody>
<tr>
<td>• sequencers, score editors</td>
<td>• programming languages</td>
</tr>
<tr>
<td>• low-level editing</td>
<td>• high-level editing</td>
</tr>
<tr>
<td>• note/data-oriented</td>
<td>• process-oriented</td>
</tr>
<tr>
<td>• process is implicit / emergent</td>
<td>• notes / data hidden</td>
</tr>
<tr>
<td>• concrete, direct</td>
<td>• abstract, indirect</td>
</tr>
<tr>
<td>• static, fixed</td>
<td>• dynamic, generative</td>
</tr>
<tr>
<td>• software designed for user</td>
<td>• software requires literacy</td>
</tr>
<tr>
<td>• favours ‘musical’ minds</td>
<td>• favours technical minds</td>
</tr>
<tr>
<td>• mainstream aesthetic</td>
<td>• avant-garde aesthetic</td>
</tr>
<tr>
<td>• tools define abstractions</td>
<td>• users define abstractions based on breaking / creating new rules and practices</td>
</tr>
<tr>
<td>• based on well-established conventions and formalisms</td>
<td></td>
</tr>
</tbody>
</table>

Disjunctions between creative practices in digital music.
Programming Metaphors

Spreadsheets are widely recognised as one of the most successful paradigms in end-user programming. (Lewis & Olson, 1987; Myers & Pane, 1995)

More people have used spreadsheets than any other programming tool. (Hendry & Green, 1994; Abraham, Burnett, and Erwig, 2009)

(from Ko et al., 2004)
Programming Metaphors in Music

(from Ko et al., 2004 and Nash, 2014)
Musical Spreadsheets

Tracker-style music sequencers (1987-present; Amiga, Dos, Windows, Mac)
• Text-based digital music notation.
• End-user programming environment for music.
  – Music sequencer + programming language.
• Flexible **balance** of composing and coding.
• Developed on the principle of **Pattern + Process**
Example: Popular Music New Order’s Blue Monday
Pattern + Process

Example:
Process-based Music
Arvo Pärt’s Fratres
Notation for Music + Programming

Broad spectrum/continuum of programmable behaviour:

- sequenced
- dynamic
- algorithmic
- automation
- generative
- end-user
- programmer
Manhattan: Trinity (Client)

• Based on a subset of the tracker UX.
  – Single screen/UI; pattern editor (notation) only.
  – No song, sample, or instrument editing.

• Based on a single pattern.
  – Explicitly limited to one 4-16 bar phrase.
  – Played looped with variation supported through formulas that modify content.
  – Enough for simple musical interaction (for beginners).
  – Designed to elicit more innovative solutions (developing coding expertise).

• Integrated sample synthesiser and instrument library.
  – Modified GM voice list, based on 100 high-quality, multi-sampled instruments.
  – Avoids set-up time at start of creative process / no pre-requisites (e.g. samples).
  – Ensures music is portable, shared code can be played on any users’ system.

• Integrated support system, activities and learning materials.
  – Interactive tutorials, exercises, and activities.
    • Digital curriculum designed to explore musical and programming concepts.
  – Reference documentation (tracker and programming syntax).
Live Demo
Evaluation (UWE BSc(Hons) Music Technology, 2015-)

- UWE’s music technology programmes combine music practice, production, plus software and hardware development (C/C++).

- STUDENT PROFILE
  - Predominantly male, young adult (mean age, 18).
  - Has artistic/‘creative’ career aspirations (CMT).
    - Creative Music Technology students are trained musicians.
  - Has production career aspirations (AMT/BAMT).
    - Significant proportion of A-Level/BTEC Music Technology.
  - Limited technical proficiency; no experience of programming.

- HISTORICAL ISSUES
  - Students struggle with programming.
    - Highest number of resits/retakes for Level 1 and 2 modules.
  - Students are not motivated to learn programming.
    - Unconvinced of utility in professional practice.
Evaluation (UWE BSc(Hons) Music Technology, 2015-)

DEPLOYMENT & METHODOLOGY

One week of Manhattan-based teaching, initially integrated in 2015/16, with the Level 1 Audio Technology module:

• One 2-hour lecture, followed by one 2-hour practical workshop.
• Module covers theory, science, and practice of music technology (runs alongside the Level 1 introductory C/C++ programming)
• Expanded to four 2-hour Manhattan-based workshops in 2017/18, interwoven with core curriculum (at 4-week intervals), and used on other modules (including project work at Level 2).

All workshop learning materials and activities integrated with software:

• Designed to simulate and support auto-didactic learning beyond classroom.
• Supplemented with live demo/walkthrough of exercises. (can be substituted with online video walkthroughs).

Quantitive (Likert) and qualitative data from end-of-session surveys.
Results (UWE BSc(Hons) Music Technology, 2015-)

I was easily able to relate activities to specific programming concepts.
I was able to draw parallels between musical and programming concepts.
Formulas provide an accessible introduction to programming.
The software encourages a new approach to music composition.
Formulas are useful for editing music.
Formulas are useful for generative/algorithmic music.
The tutorials provided an accessible path to learning key concepts.
I was able to quickly put together new ideas using music and code.
I enjoyed exploring music theory concepts through programming.
I found it difficult to extract patterns from music and translate them to code.*
I found the composition process too abstract.*
The methods encouraged me to think differently about music.
I enjoyed the challenge of trying to encapsulate music in code.
Formulas support both simple use and more complex programming.
It was easy to get started and grasp the basics.
With time, I feel I could master the programming environment.
I can imagine further musical uses for formulas using the software.
I enjoyed using the software.

(* Polarity plotted in reverse.)
Variation between Creative Music Technology (CMT) vs. Audio & Music Technology (AMT) cohorts:

- AMT (more technology-oriented, computer-based) students better disposed to the general user experience (UX), engaging with coding concepts, and generating new ideas using code (ideation).
- CMT (more music-oriented, instrument-based) students better disposed to analytical applications and abstract modelling of music (using code).
**Results** (UWE BSc(Hons) Music Technology, 2015-)

**SUMMARY OF FINDINGS**

- **New approach** to music composition.
- Enjoyable user experience, **easy to grasp** basics, with scope for more advanced and challenging uses. But notation/UI **appears daunting at first**.
- **Undo** functionality supports ideation.
- Provides an “accessible” platform for **exploring both programming and music concepts** (and their relationships), in a practical creative context.
- Formulas support a **scalable** level of complexity, from simple music editing tasks (automation) to more complex generative applications.
- Appealed most to **analytical music listeners** and mathematical minds.

**STUDENT FEEDBACK** (selected comments)

- “Very hands on and easy to use, using programming language to change parameters is a good idea for learning programming.”
- “Easy to get to grips with the process timelines and what order you should do them in. You don't have to have lots of small processes to make one sound, like Xcode and Max.”
- “It was an interesting way of composing and opened the horizon to a new way of thinking.”
- “The use of formulas gave me a much different perspective on the composition and a great level of satisfaction when it worked as intended.”
- “The program looks intimidating, even though it isn't really!”
- “It's fun to add code to cells and just see what happens.”
- “This programme made me want to be able to use and master it properly. I could see a potentially addictive side to it.”
Manhattan: Trinity (Client)

- Based on a subset of the tracker UX.
  - Single screen/UI; pattern editor (notation) only.
  - No song, sample, or instrument editing.

- Based on a single pattern.
  - Explicitly limited to one 4-16 bar phrase (up to 256 rows).
  - Played looped with variation supported through formulas that modify content.
  - Enough for simple musical interaction (for beginners).
  - Designed to elicit more innovative solutions (developing coding expertise).

- Integrated sample synthesiser and instrument library.
  - Modified GM voice list, based on 100 high-quality, multi-sampled instruments.
  - Avoids set-up time at start of creative process / no pre-requisites (e.g. samples).
  - Ensures music is portable, shared code can be played on any users’ system.

- Integrated support system, activities and learning materials.
  - Interactive tutorials, exercises, and activities.
    - Digital curriculum designed to explore musical and programming concepts.
  - Reference documentation (tracker and programming syntax).
Online system for sharing and both online and offline music collaborations.

- **Users share and publish** their work.
  - Portable file format (compressed text only). Small, fast, and human-readable.

- **Open-source music** (with code)
  - Other users download complete source for any piece.
  - Invaluable resource for beginners and learning technique.
  - Opportunity to incorporate others’ work (e.g. remix, reversion, reprise).

- **Users audition and review** finished works.
  - Separate user scores for aesthetic and technical.
  - System-generated scores for music data and coding complexity (~ data size).
  - “Wow Factor” metric (~ user score ÷ complexity), designed to model virtuosity.
    - Simpler solutions that produce more impressive art score higher.

- **Competitions** and ranking to encourage **virtuosity**.
  - Based on votes or metric (see previous).
  - Akin to existing “demoscene compos” (competitions to write in 4 channels only).
Realtime **collaborative** and **performance** mechanisms for social interaction.

- **Group creativity** using multi-user interaction (shared remote workspaces).
  - Users share their pattern with other users in remote locations, and edit the notation synchronously.
  - Samples and playback are situated locally, but pattern edits are synchronised and propagate between remote systems.
  - Users see cursors of connected users and communicate via IM.

- **Dedicated features to facilitate live performance.**
  - Scratch areas to edit and trial code before committing it to the live document.
  - Headphone mix for auditioning code, apart from main (audience) mix.

- **Language support for communication** with other technologies:
  - Protocols: MIDI output, OSC, serial, TCP/UDP, OS notifications.
  - Applications: SuperCollider, Max/MSP/Jitter, Sonic Pi, DAWs, ...
Thank you.

Download Manhattan for Mac/PC from [http://revisit.info/trinity](http://revisit.info/trinity)

HEA STEM 2018 WORKSHOP

The Manhattan Project:
Creativity and Computing Synthesised

SESSION 10.5 | TOMORROW 13:15-14:25

Dr Chris Nash (chris.nash@uwe.ac.uk)
Senior Lecturer and Course Leader, Audio & Music Technology
Department of Computer Science & Creative Technologies, UWE Bristol